UNIVERSITÀ DEGLI STUDI DI FIRENZE
Facoltà di Ingegneria - Dipartimento di Sistemi e Informatica

Corso di Dottorato di Ricerca: Ingegneria Informatica e delle Telecomunicazioni
Curriculum: Architetture dei Sistemi di Elaborazione dell'Informazione

ANALISI DI IMMAGINI DI SPARTITI MUSICALI: METODI E STRUMENTI PER IL RICONOSCIMENTO E L'INDICIZZAZIONE AUTOMATICA.

Autore:

Ivan Bruno

Supervisori:

Prof. Paolo Nesi

Prof. Giacomo Bucci

Coordinatore:

Prof. Giacomo Bucci

Ciclo XV, 2000-2003

Ringraziamenti

Un sentito ringraziamento al Prof. Paolo Nesi, al Prof. Giacomo Bucci e all'Ing. Pierfrancesco Bellini per i loro suggerimenti e il costante supporto durante questa ricerca.

Firenze, Giugno 2003 Ivan Bruno

Indice

Prefazione

In questo documento sono riportati i risultati del lavoro di ricerca condotta durante il Dottorato e in seno ai progetti di ricerca WEDELMUSIC e IMUTUS, finanziati dalla Commissione Europea. Il principale obiettivo è stato lo studio delle tematiche di indicizzazione e riconoscimento ottico automatico della musica, ovvero la conversione di un'immagine di uno spartito musicale acquisita otticamente in una rappresentazione che possa essere convenientemente memorizzata o "letta" da un computer. Sono molte le similitudini con il problema del riconoscimento ottico dei caratteri (OCR), ma le soluzioni tecnologiche adottate per i sistemi OCR non possono essere utilizzate per il riconoscimento dei simboli musicali. La notazione musicale presenta una struttura bidimensionale: in un pentagramma la posizione orizzontale denota la durata delle note, mentre la posizione verticale ne definisce l'altezza. A questo si aggiunge la possibilità di combinare i simboli musicali (una nota e un accento), la presenza delle linee del pentagramma come immagine di fondo, la distorsione delle figure, la variabilità delle dimensioni, etc. La complessità del problema è, quindi, maggiore rispetto al riconoscimento dei caratteri alfanumerici e nonostante siano presenti numerosi software commerciali, l'efficienza dichiarata superiore all'80% è spesso relativa a spartiti musicali semplici e regolari. Le nuove soluzioni proposte al problema della codifica della notazione musicale, la necessità di velocizzare il processo di scrittura musicale avvalendosi di editor musicali, la richiesta degli editori di procedure automatiche di conversione dei contenuti musicali in un formato elettronico (ad oggi il repertorio classico e moderno è archiviato in gran parte su supporti cartacei) e le nuove tecnologie nel settore delle interfaccie utente e della distribuzione dei contenuti via Internet hanno contribuito allo sviluppo di nuove tecniche in un settore ancora ben lontano dall'essere considerato esaurito dal punto di vista della ricerca e giustificano il lavoro di ricerca intorno al problema in esame.

I sistemi di riconoscimento ottico della musica, più comunemente denominati OMR (Optical Music Recognition), sono stati classificati sulla base del livello di decomposizione scelto per il riconoscimento dei simboli musicali e sul criterio adottato per definire i *simboli di base* con i quali ricostruire l'informazione musicale. Sono possibili due principali approcci alla decomposizione: (i) la rimozione delle linee del pentagramma e la ricerca

delle componenti connesse (simboli musicali completi) e (ii) l'estrazione di simboli grafici elementari o di base come le teste delle note, le pause, gli uncini, i punti, che possono essere composti per costruire la notazione musicale. Il secondo criterio è stato considerato come punto di partenza per la conduzione della ricerca sulle tematiche del riconoscimento ottico della musica. Questa scelta ha condotto alla definizione di un processo di riconoscimento diviso in quattro fasi principali: (i) la *segmentazione* dello spartito, l'identificazione e l'estrazione dei simboli di base e delle informazioni topologiche e contestuali, (ii) il *riconoscimento* dei simboli di base, (iii) la *ricostruzione* dell'informazione musicale e (iv) la costruzione del modello della notazione musicale per rappresentare l'informazione.

Nella fase di segmentazione lo spartito è scomposto in simboli/immagini elementari o di base (teste di nota, uncini, travi, etc.) attraverso un procedimento di decomposizione gerarchica, realizzato mediante l'uso del metodo delle proiezioni. Durante la fase è condotta la ricerca delle informazioni topologiche (dimensioni, posizione) e contestuali (gruppo di note, nota isolata) da utilizzare successivamente nella fase di riconoscimento e di ricostruzione. Nella fase di riconoscimento, i simboli sono classificati da una rete neurale ed è associata l'informazione topologica e la percentuale di riconoscimento (confidenza). Nella fase di ricostruzione, ai simboli riconosciuti sono associati un insieme di componenti elementari rappresentativi dell'informazione che essi esprimono. L'assemblaggio dei componenti elementari è definita su base probabilistica e su un insieme di regole sulle disposizioni dei simboli musicali rispetto al pentagramma e di scrittura musicale, formalizzate in una *grammatica posizionale* ed una *grammatica musicale*. La procedura di assemblaggio (aggregazione) è eseguita sulla base della probabilità associata al componente elementare e della verifica della regola di ricostruzione. La fase di costruzione e visualizzazione dell'informazione musicale è stata realizzata utilizzando il modello ad oggetti associato al formato WEDELMUSIC.

Il presente documento di tesi è stato organizzato in capitoli e con la struttura di seguito descritta.

Nel Capitolo 1 è riportata la definizione di un sistema di riconoscimento ottico dei caratteri musicali, sono discusse le problematiche che caratterizzano il riconoscimento dei simboli musicali (l'inclinazione delle linee, le imperfezioni di stampa, etc.) e la rappresentazione dell'informazione musicale (modelli e formati per la notazione musicale).

Nel Capitolo 2 è presentata una panoramica dei sistemi fino ad oggi sviluppati. Sono evidenziati gli aspetti comuni e le innovazioni introdotte.

Nel Capitolo 3 sono discussi e riassunti alcuni aspetti della teoria musicale, con riferimento alla notazione musicale, alla definizione dei simboli musicali, alle regole di scrittura musicale e alla struttura di una spartito.

Nel Capitolo 4 è descritta l'architettura del sistema di riconoscimento. Sono presentati i moduli di segmentazione, di riconoscimento, di ricostruzione e di rappresentazione

dell'informazione musicale.

Nel Capitolo 5 sono descritte la fase di segmentazione dell'immagine musicale e l'estrazione degli elementi grafici di base. Sono riportate le fasi che costituiscono il processo di segmentazione, le tecniche e gli algoritmi messi a punto, le informazioni di contesto estraibili durante la fase di decomposizione.

Nel Capitolo 6 è descritta la fase del riconoscimento dei simboli basato su reti neurali. Sono analizzati il database delle immagini utilizzate per l'addestramento della rete neurale, la struttura adottata ed i risultati ottenuti con l'addestramento, la codifica dell'informazione musicale.

Nei Capitoli 7, 8 e 9 è descritta la fase di ricostruzione dello spartito musicale. Sono definite la grammatica posizionale e l'insieme delle regole alla base della ricostruzione dell'informazione musicale, il meccanismo di assemblaggio dell'informazione generata dalla fase di riconoscimento e la ricostruzione dei simboli musicali.

Nel Capitolo 10 è descritto un metodo di estrazione automatica e indicizzazione dei pentagrammi, sviluppato con l'ausilio delle tecniche di elaborazione delle immagini utilizzate nella fase di segmentazione.

Nel Capitolo 11 sono riportati i risultati del riconoscimento di spartiti musicali e una valutazione comparativa con due dei più potenti software presenti sul mercato.

Infine, nel Capitolo 12 si riportano alcune conclusioni sul lavoro svolto.

Capitolo 1

Introduzione al riconoscimento delle immagini musicali

1.1 Introduzione

L'utilità dell'estrazione di informazioni dai documenti cartacei, coincide con la sempre più presente necessità di gestire le informazioni in modo automatico. Al momento attuale infatti l'informazione, nel senso più generale del termine, prodotta e conservata sotto forma cartacea è nettamente superiore a quella archiviata in forma digitale.

Il maggior problema da risolvere è consentire agli elaboratori di *leggere* documenti cartacei. Col termine leggere si vuole indicare la capacità degli elaboratori di poter interpretare, estrarre, processare ed archiviare i dati contenuti nei documenti. È molto importante evidenziare che estrarre informazioni non significa acquisire semplicemente il documento in forma digitalizzata, infatti l'informazione deve essere "riconosciuta", in modo da poter essere successivamente elaborata.

Ad esempio, su un database di immagini digitali di documenti, non è possibile effettuare una ricerca per chiavi di informazioni, a meno che non vengano immessi "manualmente" i valori dei campi su cui è indicizzato il database. È possibile inoltre rilevare che un file contenente un'immagine ha una occupazione di memoria superiore, in genere, di quella necessaria all'informazione in essa contenuta come nel caso di una immagine di una pagina di testo, la cui occupazione di memoria risulta notevolmente superiore di quella che si sarebbe ottenuta archiviando il testo in formato ASCII.

Nonostante l'aumento delle capacità delle memorie di massa renda sempre più attraente una memorizzazione dei documenti sotto forma di immagine digitale, le precedenti considerazioni costituiscono alcune delle principali motivazioni che hanno portato allo studio di tecniche di *document processing* per l'estrazione delle informazioni dalle immagini di documenti.

Il presente lavoro, riguarda in particolare uno dei moduli fondamentali di un sistema di document processing: il riconoscimento ottico dei caratteri musicali (OMR, *Optical Music Recognition*). Come sarà esposto in seguito occorrono una serie di elaborazioni preliminari all'applicazione del riconoscitore di caratteri vero e proprio, che riveste comunque nell'ambito dell'analisi dei documenti un posto di particolare importanza.

1.2 Cos'è l'OMR (Optical Music Recognition)

Il principale obiettivo dell'OMR è convertire spartiti musicali acquisiti otticamente in un formato manipolabile attraverso un computer. I benefici apportati da tale sistema sono numerosi: possibilità di eseguire un brano direttamente dal supporto cartaceo, trasporto tonale automatico, generazione dell'accompagnamento orchestrale, trasposizione editoriale, conversione in formato Braille.

Nonostante il problema di un riconoscitore ottico per caratteri musicali sia allo studio da molti anni, resta ancora un campo di ricerca nel quale rimangono aperti ancora molti problemi: a partire dal termine "riconoscimento musicale", ancora non ben definito, o dagli obiettivi del riconoscimento, non chiari in termini generali, fino ad arrivare alla definizione di un linguaggio per rappresentare la notazione musicale o di una terminologia standard nella misura dei risultati.

Allo stato dell'arte i metodi più diffusi per l'inserimento degli spartiti, non sono automatici, ma prevedono spesso l'utilizzo di programmi di digitazione manuale che si basano su linguaggi specifici di descrizione musicale: in questi casi le procedure di inserimento sono lente e suscettibili di errori. Gli editor con interfacce grafiche riducono i tempi di inserimento e la percentuale di errore, specialmente se associati a dispositivi MIDI che consentono l'inserimento automatico di altezza, ritmo e dinamica della notazione musicale. Scarsi risultati hanno avuto tentativi di riconoscimento musicale via audio. I buoni risultati conseguiti con l'OCR nel riconoscimento dei caratteri, spingono verso il riconoscimento ottico della musica che risulta lo strumento più adeguato all'inserimento degli spartiti.

1.3 OMR e OCR

Un sistema OMR riconosce i simboli su uno spartito e produce in output il risultato in un formato binario. Questa operazione è analoga a quanto compiuto dai sistemi OCR, Optical Character Recognition, che si occupano del riconoscimento del testo.

Nonostante questa apparente somiglianza, sono molti gli elementi che non permettono di adottare le tecniche sviluppate per i sistemi OCR nel campo del riconoscimento ottico delle partiture:

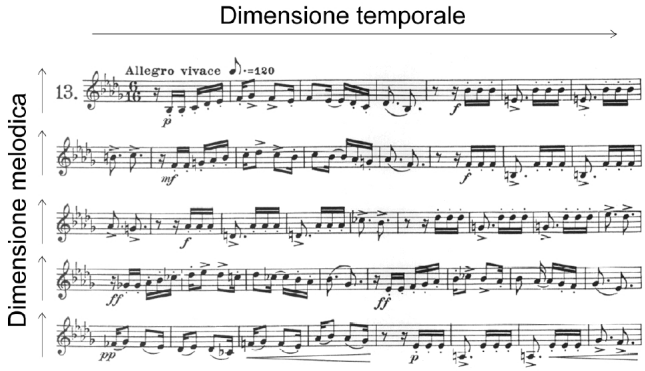


Figura 1.1: Struttura bidimensionale della musica

- Il testo è monodimensionale mentre un brano musicale è bidimensionale: nel primo caso ciò che ha rilevanza è la successione delle lettere in senso orizzontale, mentre sul pentagramma anche la posizione verticale fornisce un'informazione fondamentale, l'altezza della nota (vedi Figura 1.1).

- I caratteri che compongono una parola e quindi, più in generale, una frase o un brano, sono disgiunti, mentre i simboli musicali sono sovrapposti alle linee di pentagramma e, a volte, ad altri simboli (es. nella rappresentazione di accordi).

- I simboli musicali possono essere di dimensioni molto differenti tra loro e alcuni legano il loro significato alla grandezza (es. l'appoggiatura).

- Contrariamente a quanto accade nei testi, i simboli musicali sono formati da componenti che possono essere combinate in modo diverse (es. un gambo (stem) può essere unito a più teste e per mezzo di una trave (beam), ad un'altra gambo).

- Uno stesso simbolo musicale può apparire in forme differenti (es. la lunghezza delle legature, delle travi e dei gambi dipende dal contesto).

- Non c'è in ambito musicale un alfabeto standard: il numero di simboli è molto ampio e in continua evoluzione, infatti i compositori inventano nuove notazioni quando è necessario e per particolari strumenti si utilizzano notazioni specializzate.

- Non esiste un dizionario delle strutture musicali analogo ad un vocabolario per le parole, questo perché nella musica le regole di combinazione tra i simboli non sono riconducibili a quelle usate per il testo scritto.

1.4 Primitive, simboli parametrici e vettoriali

Data la grande complessità e varietà grafica degli spartiti musicali, in letteratura si trovano molte diverse definizioni di *simboli musicali*. Alcuni considerano simboli tutti i *gruppi* che si creano dopo la rimozione delle linee del pentagramma ([19]). Altri invece distinguono tra i *caratteri* di testo, che sono di dimensione fissa, e altri simboli (barre dei gruppi di note, legature) che hanno una forma parametrizzata. Talvolta viene fatta una distinzione tra i simboli musicali che descrivono che cosa bisogna suonare e quelli che indicano come lo si deve fare ([19], [21]); quest'ultima distinzione è interessante per il fatto che il riconoscimento del "cosa" suonare è sufficiente per un buon numero di applicazioni.

L'impostazione più comune, proposta da Mahoney ([2]) e ripresa da Kato, da Inokuchi ([3]) e da altri, è quella che considera la notazione composta da **primitive** la composizione delle quali ricrea tutta la simbologia musicale (si veda Figura 1.2). La scelta delle primitive varia tra i diversi autori, ma di solito vengono preferiti elementi tipo la testa delle note, il gambo, le barre dei gruppi, le alterazioni, ecc. Sono state proposte anche rappresentazioni *vettoriali* delle varie forme grafiche, per far fronte ai problemi di variabilità delle dimensioni della notazione ([4]).

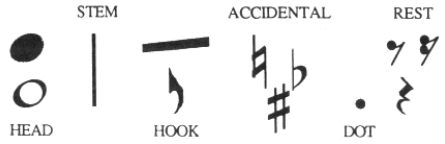


Figura 1.2: Insieme di primitive preso come riferimento da Kato e Inokuchi.

1.5 "Optical Music Recognition" (OMR)

OMR è la dizione comune con la quale vengono indicati i sistemi di lettura e riconoscimento automatico della musica. Genericamente possiamo definire un *sistema OMR* quel software che, attraverso l'elaborazione e l'analisi di uno spartito musicale in forma di immagine digitale, riconosce la notazione musicale e ne costruisce una rappresentazione simbolica.

I sistemi di riconoscimento automatico della musica possono essere classificati e studiati da vari punti di vista, considerato anche il fatto che in letteratura sono stati usati approcci di ricerca anche molto diversi. Quindi, pur non essendo facile definire quali siano le fasi di un generico processo di riconoscimento della musica, di seguito è riportata una lista dei possibili passaggi:

- Digitalizzazione dello spartito musicale.
- Pre-elaborazione grafica.
- Identificazione e/o rimozione delle linee del pentagramma.

- Isolamento (a volte in più fasi) e/o segmentazione degli elementi grafici *primitivi* che compongono la notazione musicale.
- Ricomposizione e classificazione dei simboli musicali.
- Post-elaborazione e classificazione dei simboli musicali.
- Generazione della rappresentazione simbolica nel formato scelto.

La pre-elaborazione e filtraggio grafico dell'immagine è un operazione non sempre ritenuta necessaria, al contrario dell'identificazione e dell'eventuale rimozione del pentagramma che è considerato un passo obbligato per molti autori (si veda il paragrafo 1.5.2). Le fasi di analisi grafica (isolamento primitive, segmentazione, composizione e classificazione dei simboli) sono centrali nei sistemi OMR e risultano le più studiate in letteratura; gli strumenti e le soluzioni proposte sono le più varie e risentono molto anche delle molte tecniche di *pattern recognition* e dei metodi adottati negli OCR. Ultime nella ricerca dal punto di vista storico, ma fondamentali per la soluzione del problema del riconoscimento musicale sono le procedure di applicazione della conoscenza sintattica e semantica del processo, per lo più utilizzate nella fase di post-elaborazione (si veda paragrafo 1.5.3). Infine, essenziale per ogni sistema, è il "salvataggio", cioè la generazione della rappresentazione simbolica del formato scelto; anche in questo caso il problema è aperto non esistendo ancora un linguaggio standard per l'espressione della notazione musicale (si veda paragrafo 1.8).

È interessante, prima di entrare nel dettaglio dei principali lavori esistenti in letteratura (si veda il Capitolo 2), osservare il problema da diversi punti di vista, facendo delle sommarie classificazioni dei sistemi OMR già sviluppati e delle idee proposte dai vari autori.

1.5.1 Sistemi On-line e Off-line

Una prima distinzione grossolana può essere fatta tra sistemi di riconoscimento automatico di tipo *On-line* oppure di tipo *Off-line* ([32]).

In un sistema On-line la macchina analizza lo spartito musicale e genera un risultato quasi istantaneamente. Tale sistema può essere combinato con un robot che, cercando di imitare il comportamento umano, suona uno strumento musicale in tempo reale. In questo caso il computer deve essere in grado di analizzare e produrre le informazioni in brevissimo tempo: questo implica che la lettura dello spartito avvenga in maniera locale e che non possono essere utilizzati metodi di correzione a posteriori. Sistemi di questo tipo sono stati realizzati con risultati interessanti per la rapidità del riconoscimento, ma poco soddisfacenti specialmente dal punto di vista dell'interpretazione e dell'effetto musicale ([37], [38], [39]). Un altro esempio è offerto dai sistemi di immissione dati basate su nuove

interfaccie utente e sulla tecnologia di riconoscimento dei gesti umani. Grazie ad esse, sono stati sviluppati sistemi basati su penne elettroniche ([40],[41]) che aiutano l'utente nello scrivere la musica usando una penna in modo tradizionale. Tali sistemi consistono in un tablet PC, normalmente fornito di uno schermo a cristalli liquidi LCD sensibile al tocco e di una penna elettronica usata per scrivere su di esso. L'obiettivo di tali sistemi è quello di minimizzare il tempo di immissione dati, allo stesso tempo essi devono affrontare tutte le problematiche derivanti dalla difficoltà di riconoscimento della scrittura umana.

In un sistema Off-line, lo spartito viene digitalizzato, di solito attraverso uno scanner, e memorizzato in un immagine. Successivamente un computer analizza l'immagine e la converte in una appropriata rappresentazione simbolica utilizzando la codifica scelta. Poiché in questo caso prima di generare l'uscita può essere analizzato l'intero spartito, l'informazione globale può essere utilizzata per incrementare l'accuratezza del riconoscimento. Sistemi di questo tipo non hanno vincoli temporali stringenti e permettono una sperimentazione molto più vasta e flessibile; inoltre suscitano un maggior interesse visto che le necessità di eventuali prodotti commerciali impongono una buona qualità del riconoscimento associata ad una bassa percentuale di errore, più che una grande velocità di produzione dei risultati.

1.5.2 Sistemi con identificazione del pentagramma

Le linee del pentagramma nella notazione musicale hanno un ruolo fondamentale: esse definiscono le coordinate verticali per l'altezza dei simboli musicali e forniscono la direzione orizzontale per il sistema di coordinate temporali. Inoltre nell'analisi delle immagini musicali sono considerate un riferimento dimensionale, oltre che un indicatore della qualità della digitalizzazione e dell'inclinazione dell'immagine. Molto spesso però il pentagramma viene ritenuto un ostacolo al riconoscimento dei simboli musicali quindi vengono studiate varie metodologie per la sua rimozione o cancellazione.

È dunque opinione comune che, rimosse o meno, le linee del pentagramma devono essere comunque identificate per le importanti informazioni che possono fornire: in molti sistemi infatti, nella fase iniziale, viene fatto un grande sforzo per questa operazione.

D'altra parte in alcune ricerche è saltata la fase iniziale di identificazione ed eventuale cancellazione del pentagramma, per effettuare immediatamente l'analisi della simbologia musicale. In questo caso il pentagramma è considerato un simbolo grafico come gli altri e utilizzato solamente come riferimento sull'altezza delle figure.

1.5.3 La "conoscenza" musicale come strumento di interpretazione

Una prospettiva interessante per comprendere lo sviluppo dei sistemi OMR è l'estensione dell'utilizzo della "conoscenza" musicale al processo di riconoscimento.

La disposizione bidimensionale della notazione musicale fornisce delle informazioni molto importanti; quindi ogni simbolo deve essere analizzato sia singolarmente, che in relazione alla posizione che ha rispetto agli altri simboli. Il significato di queste relazioni può essere espresso attraverso una descrizione sintattica e alcuni autori suggeriscono di utilizzare metodi grammaticali bidimensionali partendo anche da studi sviluppati per linguaggi mono-dimensionali. C'è chi ritiene che la determinazione delle regole sintattiche in forma algoritmica sia la fase più importante nella soluzione del problema del riconoscimento ([19], [21]) oppure chi utilizza la descrizione algoritmica della musica per la generazione della notazione musicale. Altri utilizzano le grammatiche musicali per migliorare la fase finale del processo di riconoscimento come strumento di correzione degli errori di interpretazione ([39]).

Un contributo importante è stato dato da Fujinaga ([30], [31]) il quale sostiene che la notazione musicale può essere formalizzata con una grammatica e che "la grammatica è *context-free* e LL(k); questo è in effetti quello che permette ai musicisti (top-down parser) di leggere la musica in maniera così efficiente". Si insiste inoltre sul fatto che un approccio puramente sintattico, nel quale il contesto non è preso in considerazione, ha molte limitazioni, e suggerisce di introdurre informazioni semantiche all'interno della grammatica. Partendo da queste considerazioni alcuni autori hanno cercato di estendere la "conoscenza" musicale al controllo di tutto il processo di riconoscimento ([16], [17]): la grammatica aggiunge un'altra dimensione, il *contesto*, che permette non solo di utilizzare regole sintattiche per la correzione posteriore, ma anche di aggiungere un *livello grafico* per migliorare le fasi di segmentazione e etichettatura degli oggetti grafici.

Comunque non mancano autori che preferiscono concentrare il processo di riconoscimento tutto sulle tecniche di identificazione e classificazione di tipo grafico, senza utilizzare in alcun modo le informazioni musicali di tipo sintattico o contestuale.

1.6 Tipologie di problemi

I fattori che influenzano la realizzazione di un sistema OMR sono di varia natura e complessità; la loro incidenza è andata mutando con l'evoluzione tecnologica del settore. Ad esempio, rispetto ai tempi in cui la ricerca sul riconoscimento automatico di partiture ha preso il via, fine anni '60 e inizio '70, si possono considerare superate le limitazioni hardware costituite dalla lentezza dei processori, dalla ridotta quantità di RAM disponibile sui sistemi di elaborazione e dalle capacità inadeguate delle memorie di massa; oggi è tutt'altro che raro disporre di un processore ad alta velocità, di 8 o 16MByte di RAM, di qualche GByte di hard disk e di uno scanner capace di digitalizzare a 300 d.p.i.. Quindi, potendo acquisire otticamente immagini ad alta definizione, memorizzare ed elaborare in tempi

ragionevoli grandi quantità di dati, il riconoscimento automatico di spartiti è diventato puramente un problema software ([14]).

Invece, fattori tuttora da considerare sono: la qualità grafica del materiale digitalizzato, complessità del brano musicale, il formato da utilizzare per conversione della partitura e l'efficienza dei programmi applicativi usati per completare la riproduzione musicale ([26], [27]).

1.6.1 Qualità grafica del materiale digitalizzato

I problemi legati alla qualità grafica della partitura digitalizzata sono riconducibili a tre principali aspetti: la superficie visiva, il riconoscimento dell'oggetto e la rappresentazione musicale. I problemi nella superficie visiva sono per lo più dovuti ad imperfezioni di stampa:

- la rotazione dei pentagrammi, per cui le linee non sono perfettamente parallele al bordo della pagina;
- la curvatura dei pentagrammi, per cui le linee non sono diritte (questo problema può essere presente nell'originale, ma può anche essere introdotto dall'uso di scanner manuali per l'acquisizione dell'immagine o da una fase intermedia di fotocopiatura);
- le variazioni di spessore delle linee del pentagramma;
- la posizione scorretta dei simboli (es. una nota che copre sia uno spazio che una riga).

Una stampa imperfetta mette in risalto problemi dovuti alla perdita di informazione: le linee del pentagramma sono discontinue,gli oggetti non sono completamente disegnati o non completamente riempiti (es. note da un quarto con macchie bianche nella testa); oppure generano problemi di informazione superflua come la presenza delle macchie, le quali, in base a dove si verificano e in relazione agli altri oggetti, possono essere interpretate erroneamente come simboli o parte di questi. Per quanto possano apparire questioni trascurabili per l'occhio umano, queste irregolarità possono essere molto destabilizzanti per un software di riconoscimento che accetta piccole variazioni, ma solo all'interno di un insieme ben limitato.

I problemi riscontrati nel riconoscimento dell'oggetto variano in base all'approccio adottato. I principali e comuni a molti approcci, sono:

- Le dimensioni: le variazioni di dimensione tra oggetti uguali (es. la lunghezza dei gambi e la lunghezza delle travi nelle vecchie partiture o la diversità fra un'appoggiatura e una nota comune) generano difficoltà nella fase di riconoscimento se questa viene condotta basandosi esclusivamente sulle dimensioni.

- Le forme e la rappresentazione: un esempio di inconsistenza nella rappresentazione è dato dalla nota non perfettamente centrata sulla linea del pentagramma o in uno spazio tra due linee di pentagramma. L'attraversamento delle linee di pentagramma da parte delle travi (gruppi di note) può far nascere forme irriconoscibili o difficilmente riconducibili a simboli musicali o parte di questi. Si tratta di un fenomeno frequente poiché l'angolo di inclinazione delle travi è variabile e la copertura da esse generata dipende dal piazzamento verticale delle note raggruppate. Costituiscono un altro esempio tipico di inconsistenza le chiavi e le pause da un quarto (pause di seminimima) che hanno ampie variazioni nella loro rappresentazione grafica.

- La contiguità e la sovrapposizione: ad esempio le legature possono creare problemi tollerabili se si limitano ad attraversare una gamba, ma se toccano una nota o attraversano un segno di dinamica creano oggetti apparenti che non vengono ritrovati nell'insieme dei simboli grafici.

Lo studio sulla rappresentazione della musica occidentale in notazione musicale tradizionale non ha portato all'esaurimento o alla descrizione sistematica di tutte le anomalie come, ad esempio, quando più oggetti (note di diversa durata) condividono parti in comune (il gambo) in modo improbabile.

Tali esempi interferiscono con un approccio grammaticale al riconoscimento degli oggetti, poiché, nella sua rappresentazione grafica, la logica sottostante la notazione è circondata da ogni parte da eccezioni di natura impredicibile. In più, i programmi di riconoscimento possono essere confusi da oggetti che non si intende acquisire.

1.6.2 Complessità del brano musicale

Si possono delineare due aspetti della complessità di un brano musicale, uno legato alla suddivisione fra parte strumentale e direttoriale, l'altro legato all'architettura musicale. Per quanto concerne il primo aspetto, la parte strumentale, generalmente, presenta una linea melodica (monovoce) su un solo pentagramma, ad eccezione degli strumenti come il pianoforte, il clavicembalo, la celeste e l'arpa che necessitano di un doppio pentagramma e dell'organo con tre pentagrammi (multivoce). La parte direttoriale, invece, è strutturata in modo da presentare una visione contemporanea delle singole parti strumentali ed è quindi caratterizzata da un gruppo di pentagrammi. Si capisce, quindi, come in un caso si abbia a che fare con un solo pentagramma alla volta (al massimo un numero ristretto), mentre nell'altro con un gruppo; questo andrà ad influenzare il processo di identificazione dei pentagrammi e il trattamento dell'informazione topologica. L'informazione sul numero e la disposizione dei pentagrammi è l'aspetto principale nell'analisi del layout della pagina musicale. Raggruppare i pentagrammi per strumento significa distribuire l'informazione musicale allo specifico strumento e non casualmente.

L'architettura musicale è riconducibile ai concetti di monofonia e polifonia del brano. Un brano monofonico, presenta una sola voce musicale o layer e le note musicali vengono suonate una alla volta in successione. Un brano polifonico, invece, presenta più voci ovvero le note possono essere suonate anche simultaneamente (ad esempio note in accordo) e avere differente durata. Di fatto, la conoscenza della struttura monofonica o polifonica è molto importante dal punto di vista del riconoscimento dell'informazione e della semantica musicale. Se nel primo caso la dimensione del riconoscimento è quella di una voce e quindi tutta l'informazione può essere vista come una sequenza di simboli da riconoscere, nel secondo il numero non è precisato a priori, ma è legato al contesto. Quest'ultimo aspetto genera una problematica nel trattamento dell'informazione dovuta alla contemporaneità di simboli che una volta riconosciuti devono essere distribuiti a voci diverse mantenendo le relazioni musicali.

1.6.3 Manoscritti e musica stampata

Il riconoscimento di manoscritti musicali oltre ai problemi fin qui discussi introduce e deve affrontare tutte le difficoltà legate al riconoscimento della scrittura manuale. Nei manoscritti la forma dei simboli varia da autore ad autore ed è legato al periodo storico; la regolarità che offre la stampa è difficilmente riscontrabile. In questo senso, le linee del pentagramma presentano un'alta variabilità dello spessore e degli andamenti rettilinei divenendo curvilinei ed inclinati. Anche i simboli presentano una notevole variabilità nelle forme, nelle dimensioni e nell'intensità della scrittura; rispetto alla musica stampata si hanno molti più simboli sovrapposti, simboli diversi possono apparire connessi tra loro anche se non esiste una relazione musicale tra di essi. La degradazione della carta e la presenza di inchiostro richiede uno notevole sforzo e la realizzazione di algoritmi per effettuare una ripulitura dell'immagine. Questi sono solo alcuni dei principali problemi e condizionano pesantemente la scelta dei metodi da usare nel riconoscimento. La tecnologia usata per il riconoscimento di spartiti musicali stampati non è completamente riusabile per i manoscritti e tanto meno un sistema in grado di riconoscere gli spartiti stampati può essere esteso per riconoscere i manoscritti. Tali sistemi generalmente sono progettati in modo diverso

1.7 Specifiche standard di un sistema OMR

I progetti e le ricerche sviluppate fino ad oggi si sono disinteressati alla definizione di standard sul problema OMR, introducendo talvolta nuove definizioni di linguaggi di rappresentazione musicale, nuovi formati di salvataggio, riferimenti a notazioni musicali spesso incomplete e differenti, l'utilizzo di una tecnologia diversa per la valutazione dei risultati ottenuti.

Questa incertezza negli strumenti standard da utilizzare per la progettazione di un sistema di riconoscimento automatico non solo causa un rallentamento nel processo di ricerca, ma non consente di confrontare e valutare il lavoro svolto da altri autori in ambito dell'OMR.

1.7.1 Valutazione dei risultati

L'assenza di una terminologia standard e di un database di immagini da usare come riferimento nei tests non consente una facile e corretta valutazione dei risultati. Generalmente viene utilizzata, come nei sistemi OCR, la **percentuale di errore** o **accuratezza** (*error rate*), che indica il rapporto tra simboli riconosciuti e simboli totali. Ma in campo musicale è molto difficile definire quando un simbolo sia stato riconosciuto, quali siano le caratteristiche da individuare, quali relazioni con i simboli vicini siano essenziali e che importanza abbia il contesto. Inoltre non tutti gli autori fanno riferimento a notazioni musicali complete e a volte l'accuratezza viene calcolata rispetto al numero di simboli rappresentabili con il linguaggio scelto.

Una soluzione plausibile, anche se non del tutto soddisfacente, è quella basata sulla classificazione dei sistemi OMR rispetto alla complessità del **livello di riconoscimento** che viene preventivata nel progetto. Molti ricercatori hanno definito quale sia la "profondità" del riconoscimento della notazione musicale in stretta relazione al problema che volevano affrontare. In letteratura si possono distinguere almeno tre *livelli* di specifiche di riconoscimento ([10]):

1. Riconoscimento dei caratteri, dei simboli e delle linee (analogo alla vettorializzazione).

2. Riconoscimento di sequenze parallele di note con altezza e durata.

3. Riconoscimento completo e interpretazione di tutti i simboli.

In questo modo è possibile definire la **percentuale di errore** rispetto ad un obiettivo più preciso e la valutazione dei sistemi OMR diventa più uniforme e comprensibile.

Questa però non può essere considerata la soluzione definitiva poichè non esiste una classificazione standard dei livelli di riconoscimento, e il problema, anche se in forma più comprensibile, si ripropone.

1.7.2 Convenzioni sulle notazioni musicali

Nel campo della composizione e dell'editoria vengono usate notazioni anche molto diverse e, pur facendo riferimento alle stesse regole sintattiche, è molto facile trovare l'eccezione

per ogni regola. Anche nei simboli musicali di base talvolta, a seconda della casa editrice, dell'epoca di composizione o della particolare esecuzione, si riscontrano delle notevoli variazioni.

Ciò non toglie che, per le applicazioni informatiche in campo musicale, è fondamentale definire nel modo più accurato possibile la simbologia, le regole, la sintassi e la semantica musicale ad esse associata. Infatti pensare che lo stesso pezzo musicale possa essere scritto e interpretato in molti modi diversi impedisce una corretta progettazione dei software musicali e complica il panorama degli standard proposti.

Una soluzione potrebbe essere quella di costruire un database di spartiti musicali digitalizzati associati alla corretta interpretazione ([10]); in questo caso sarebbe opportuno concordare anche il formato e il linguaggio di rappresentazione dei risultati dell'iterpretazione. Un database di questo tipo, che non avrebbe la pretesa di coprire l'intera letteratura musicale, potrebbe comunque fornire una base di riferimento di grande utilità anche per il confronto delle prestazioni di sistemi diversi.

1.8 Formato di conversione dell'immagine digitalizzata

Una delle questioni fondamentali per l'utilizzo dell'informatica in campo musicale è la definizione di un formato standard di salvataggio delle informazioni. Il problema coinvolge tutte le applicazioni che hanno a che fare con le notazioni musicali: dagli editor ai programmi orientati all'esecuzione o alla stampa, dai database fino ai sistemi di riconoscimento automatico. Sono stati sviluppati moltissimi formati di salvataggio, di solito legati a progetti specifici. Di seguito sono presentati i principali formati di rappresentazione della notazione musicale.

1.8.1 Il linguaggio "ideale" per l'espressione della conoscenza musicale

Lo scopo del linguaggio scelto dovrebbe essere quello di fornire una struttura per esprimere la *conoscenza astratta* della notazione musicale. Questa è un'impostazione molto diversa da quella di alcuni linguaggi esistenti, i quali spesso hanno lo scopo di descrivere un tipo particolare di spartito o solamente le sue caratteristiche logiche o grafiche.

Se un linguaggio potesse davvero "catturare" l'essenza della conoscenza musicale, allora ogni notazione musicale potrebbe essere descritta. In questo senso la specificazione della conoscenza musicale, al pari della scelta corretta delle *primitive* grafiche diventa una *componente dinamica* del sistema e in un programma OMR che seguisse questa filosofia, prima dell'interpretazione, dovrebbe essere indicata quale tipo di semantica si vuole utilizzare.

Dunque il linguaggio ideale per l'espressione della musica dovrebbe fornire le specifiche riguardo alle configurazioni corrette delle *primitive* grafiche e un metodo per esprimere

la semantica musicale. In questo modo si può riuscire a dare significato agli elementi grafici delle notazioni musicali. Bainbridge ([4]) sostiene che, siccome non esiste un formato standard di interscambio musicale, il formato di rappresentazione deve essere definito attraverso una *struttura interna di dati* interlacciabile ai particolari formati di interesse; inoltre vengono proposte come caratteristiche fondamentali del linguaggio di specifica della semantica musicale la **decomposizione strutturale/procedurale** (stile object oriented) e il **supporto Abstract Data Type**.

Nell'ambito dei sistemi di riconoscimento automatico ci sono state molte discussioni sul legame che ci deve essere tra il linguaggio e le coordinate della pagina; infatti se in alcuni casi la conoscenza della posizione è inutile, in altri campi, come quella dell'analisi tipografica, diventa essenziale. Una proposta interessante è quella che precisa che con il termine *estrazione* si deve intendere l'estrazione delle rappresentazioni musicali legate alle coordinate fisiche della pagina e con il termine *riconoscimento* il riconoscimento delle rappresentazioni musicali indipendenti dalla pagina. Un'alternativa (utilizzata dal formato NIFF) potrebbe essere quella di separare l'informazione musicale nelle sue componenti logiche, grafiche ed esecutive.

1.8.2 Gli standard proposti

In questa sezione è presentata una rassegna dei più importanti e attuali formati di codifica dell'informazione musicale. La rassegna non vuole essere esaustiva ed è focalizzata prevalentemente agli aspetti rilevanti che caratterizzano i vari formati.

MIDI, Music Instrument Digital Interface – Il MIDI è un linguaggio orientato all'esecuzione e pertanto non è in grado di modellare le relazioni fra i simboli musicali e codificare spartiti musicali in modo professionale: nel MIDI gli accenti, i mordenti, i trilli, le legature per esempio sono difficilmente rappresentabili e riconoscibili. Tuttavia il MIDI è il formato di codifica più usato e il più diffuso su Internet. Può essere facilmente generato dalle tastiere elettroniche, è stato adottato dalle industrie musicali per l'esecuzione musicale da parte dei computer grazie al poco spazio di memoria richiesto e può essere eseguito e manipolato da sintetizzatori musicali diversi. Una grande quantità di file MIDI sono stati prodotti e ricoprono svariati generi musicali, tuttavia essi non sono in grado di rappresentare in modo completo e dettagliato lo spartito originale del brano musicale tradotto. Nel formato MIDI è stata prevista anche la possibilità di personalizzare il formato con l'aggiunta di istruzioni che ne consentano un estensione. In tal senso, sono state definite numerose versioni di formati MIDI ([28]), ma nessuna di queste ha avuto larga diffusione ed è stata in grado di sostituire il formato classico. L'uso del linguaggio MIDI è principalmente quello di interscambio tra formati, ma la sua capacità di modellazione della notazione musicale è molto limitata e pertanto può portare a transcodifiche con perdita

di informazione. La maggioranza degli editor musicali è in grado di caricare e salvare la notazione musicale in formato MIDI.

SCORE – È probabilmente l'editor di notazione musicale più usato tra gli editori musicali per l'alta qualità e professionalità della stampa in formato Postscript ([54]). In SCORE ciascun simbolo musicale può essere posizionato con precisione all'interno della pagina musicale secondo le reali esigenze dei copisti musicali. I simboli complessi possono essere prodotti usando elementi grafici e posizionati sul pentagramma in qualunque posizione. Le relazioni tra i simboli non sono definite, così se una nota viene rimossa o spostata essa non influenza i simboli ad essa vicini. SCORE non presenta alcuna distinzione per esempio tra legature di portamento e legature di valore e non prevede alcuna conversione di una sequenza di pause in un'unica generica. SCORE è un editor e un linguaggio orientato alla pagina, nel senso che l'informazione musicale serve solo per la preparazione della pagina: l'editor è in grado di gestire la stampa di una pagina per volta. Poiché lo spartito musicale è realizzato pagina per pagina - una pagina un file - l'estrazione automatica delle parti è un'operazione complessa così come ogni modifica che coinvolge la fine e l'inizio di due pagine successive.

MusiXTEX – È un set di macro definite per il linguaggio LaTex e/o Tex per la realizzazione di spartiti musicali ([55],[56]). Il linguaggio è interessante in quanto la sua struttura è sostanzialmente simbolica mentre i comandi grafici possono essere aggiunti per fornire un preciso posizionamento. Le relazioni tra i simboli dipendono dall'ordine dei simboli che compaiono nella codifica. Il linguaggio è orientato alla stampa e permette il posizionamento di simboli grafici in qualunque punto della pagina e sono disponibili alcune semplici regole per l'inserimento dei simboli (definizione se i gambi delle note devono essere rivolti verso l'alto o il basso). Con MusiXTEX, è possibile definire regole specifiche per l'organizzazione visuale dei simboli musicali all'interno della pagina, aumentando in questo modo la potenza dei linguaggi LaTEX e TEX. In MusiXTEX, il lavoro di scrittura musicale deve essere eseguito manualmente; esso non supporta meccanismi di raggruppamento automatico delle note, di definizione automatica della direzione del gambo delle note e la gestione automatica degli abbellimenti e simboli di accento.

MusicXML – È un formato di interscambio in linguaggio XML ed è stato sviluppato da Recordare ([59]). È basato su due formati testuali per la rappresentazione della notazione musicale: MuseData e Humdrum ([28]). La musica è rappresentata secondo le modalità *time-wise*,dove le parti sono organizzate secondo l'ordine delle battute, o *part-wise*, dove le battute sono organizzate seguendo l'ordine delle parti. Ll linguaggio XSLT (Extensible Stylesheet Language Trasformation), usato per la trasformazione di documenti XML, permette la trasformazione da un formato all'altro. Il formato copre la notazione musicale

occidentale a partire dal XVII secolo in poi ed è principalmente orientato alla descrizione delle strutture logiche della musica anche se possono essere aggiunti alcuni dettagli grafici. Un plug-in sviluppato per l'editor Finale consente di caricare e salvare i files usando questo formato; l'applicazione Sharpeye (un OMR commerciale) si avvale di tale formto come formato di interscambio con FINALE. Al livello di linguaggio XML, MusicXML è fortemente basato sull'uso di Tag piuttosto che di attributi, limitandone pertanto la flessibilità nella definizione di nuovi simboli considerati come valori degli attributi. L'aggiunta di un valore in XML è più semplice che definire una nuova regola per gestire il nuovo Tag.

FINALE – Questo formato è prodotto da FINALE ed è basato sulla codifica Enigma. Tale codifica risulta parzialmente documentata. L'editor e il formato sono principalmente orientati alla preparazione della pagina piuttosto che alla definizione delle relazioni fra i simboli. Testimonianza di ciò è la mancanza di legami fra di essi: infatti possono essere liberi da vincoli ed essere posizionati in qualunque parte della pagina. Il formato è stato recentemente esteso e permette la definizione di alcune relazioni fra i simboli della notazione musicale pur tuttavia non essendo questo la filosofia del formato la scelta della definizione dei vincoli è rimandata all'utente. Il modello in Finale non mostra una chiara traccia per le voci (*layer*) che passano da un pentagramma ad un altro nelle parti multi pentagramma (parti per pianoforte, organo, arpa) ed in molti casi, la sistemazione dei simboli musicali è abbastanza complessa poichè il meccanismo automatico di completamento delle battute è un fattore di disturbo.

GUIDO – È un formato testuale per la descrizione della musica in rappresentazione simbolica. La descrizione è estremamente compatta e sembra essere ottimizzata per l'immissione diretta da parte dell'utente, non è previsto l'uso di un editor specifico per generare la codifica. Un insieme di tools sono previsti per trasformare il linguaggio GUIDO in MIDI, FINALE, PostScript or GIF o per convertire in GUIDO files MIDI e FINALE. Il formato GUIDO è stato progettato su 3 livelli. Il livello *Basic* descrive i simbuli musicali di base della notazione occidentale (note, pause, legature, ecc.) e la loro struttura (pentagrammi, voci, accordi). Il livello *Advanced* estende il precedente introducendo il supporto per la gestione delle informazioni relative alla formattazione dello spartito e aspetti musicali più sofisticati. Infine, il livello *Extended* introduce concetti che vanno oltre la notazione musicale convenzionale. I tools, attualmente, supportano il livello Basic e Advanced, nessuna specifica è disponibile per i livelli Advanced ed Extended. Le regole di formattazione automatica sono codificate nel modulo di rendering, il linguaggio supporta il raggruppamento automatico delle note (*beaming*) lasciando tuttavia la possibilità di effettuare un posizionamento preciso forzando la posizione del gruppo di note. I simboli espressivi supportati

nel livello Basic coprono le più importanti indicazioni espressive (staccato, tenuto, accento, marcato) e abbellimenti (trillo, mordente, gruppetto, tremolo, glissando); non sono presenti simboli per specifici strumenti musicali (violino, piano, arpa, etc.). Il linguaggio GUIDO non consente di introdurre nuovi simboli definiti dall'utente.

NIFF, Notation Interchange File Format – Molti produttori di software musicale commerciale, insieme a programmatori e utenti esperti, hanno intrapreso il progetto di definizione del formato NIFF. Consapevoli della difficoltà di creare un modello informatico perfetto per la notazione musicale, i soci del progetto NIFF si sono prefissi un traguardo più ragionevole: creare in tempi brevi un formato pratico ed usabile. Hanno optato per una soluzione funzionale e solida, rispetto ad un linguaggio che risolvesse ogni possibile problema e descrivesse anche i casi più inusuali.

Quindi NIFF, sviluppato secondo la struttura RIFF (Resource Interchange File Format) elaborata da Microsoft ([49]), è nato con l'intento di consentire lo scambio di notazione musicale tra editor musicali, programmi, esistenti ed in via di sviluppo per la stampa esistenti e per il riconoscimento ottico degli spartiti. Le caratteristiche peculiari che lo rendono uno standard di fatto per l'archiviazione della musica, sono la sua *estendibilità*, *flessibilità* e *compattezza*. Infatti, NIFF consente la rappresentazione delle situazioni più comuni che possono presentarsi nella notazione musicale convenzionale, offrendo al tempo stesso ai programmatori la possibilità di definire estensioni al linguaggio in modo da gestire in proprio i casi più inusuali. Permette inoltre l'inclusione di file e font EPS (Encapsulated PostScript), così come i dati e notazione MIDI.

Un pregio fondamentale è la gestione della informazione grafica. Infatti l'informazione musicale possiede tre distinte componenti:

- *Informazione logica*: la parte strettamente musicale della notazione.
- *Informazione grafica*: la parte grafica della notazione.
- *Informazione esecutiva*: può essere descritta mediante formato MIDI.

I progettisti di NIFF hanno ritenuto opportuno suddividere la componente grafica in due sotto livelli:

1. *page-layout information*: informazione grafica relativa all'impaginazione
2. *non page-layout information*: informazione grafica relativa alla disposizione dei simboli musicali.

La sola informazione che NIFF esige assolutamente è quella logica; anche se è strutturato come un formato *page-ordered*, può essere impiegato con successo anche da programmi

che impiegano solo la componente logica. Quando l'informazione grafica è presente al completo in un file NIFF, il programma che legge tale file può o rispettare l'impaginazione e le altre informazioni relative al posizionamento dei simboli oppure ignorare i dati grafici e disporre i simboli musicali sulla base di impostazioni proprie. Quando nessun dato grafico è specificato nel file, il programma che lo importa deve calcolare la disposizione di tutti gli elementi grafici.

WEDELMUSIC – Il linguaggio WEDELMUSIC ([57]) può essere considerato l'evoluzione del formato MOODS ([58]), al quale sono stati aggiunti caapcità multimediali ed una formalizzazione basata sul linguaggio XML. Gli aspetti multimediali relativi ai contenuti ed i formati supportati riguardono:

- Identificazione (ISMN);
- Classificazione (Z39.50, UNIMARK);
- Protezione (Encryption, watermark, Digital Right Management);
- Stampa (immagini e rappresentazione simbolica degli spartiti);
- Visualizzazione simbolica della notazione musicale;
- Visualizzazione immagini di partiture musicali;
- Audio (MP3, WAV, MIDI);
- Sincronizzazioni audio (audio e spartito musicale, slide show);
- Video (AVI, MPEG);
- Documenti (MS Office, PDF, PostScript, HTML, etc.);
- Visualizzazione della lirica in lingue differenti;
- Immagini (GIF, TIFF, PCX, etc.).

Relativamente alla rappresentazione simbolica della notazione musicale, l'idea principale della modellizzazione della notazione è mantenere separate le parti singole e riprodurre lo spartito direttoriale componendo le descrizioni delle parti. Per questo ragione, la formalizzazione XML è divisa in due gruppi. Il primo gestisce l'informazione relativa allo sparito direttoriale ed include gli aspetti di identificazione e classificazione, la struttura dei sistemi di pentagrammi ed annotazioni testuali. Il secondo gruppo definisce l'insieme dei file, uno per ogni singola parte, le informazioni di identificazione e classificazioni specifiche per la parte e la descrizione della notazione musicale simbolica.

Nel formato WEDELMUSIC, una parte musicale è vista come l'insieme di uno o più pentagrammi ed una sequenza di battute ciascuna con la propria descrizione (tempo, chiave, tonalità). La battuta è composta da uno o più voci (*layer*) distribuite su uno o più pentagrammi e per ogni voce è previsto la possibilità di passare da un pentagramma all'altro. La voce è considerata come l'insieme delle figure musicali (note, pause, cambi chiavi, etc.) e ciascuna figura conosce la propria posizione relativamente al numero del pentagramma. Le figure possono avere associate simboli minori come: le alterazioni, gli accenti, i punti di valore, le espressioni dinamiche. Dopo la descrizione di ogni battuta con le proprie voci e figure, segue la descrizione dei simboli orizzontali. Questi simboli sono principalmente le legature, i crescendo, i decrescendo, i cambi di ottava, le indicazione dei gruppi irregolari. I simboli orizzontali sono associati alle note o alle pause e agli ancoraggi. Quest'ultimi definiscono dei punti nascosti e di riferimento sul pentagramma.

Nel formato, gli elementi della notazione musicale (note, pause, cambi chiave, etc.) sono identificati in modo univoco per mezzo di identificativi numerici assegnati al momento dell'inserimento. Gli identificativi sono riferiti alla parte, alla battuta, al layer, al codice del simbolo e sono definiti *Indirizzi Simbolici*. Altri dettagli numerici presenti nella descrizione simbolica riguardano i parametri relativi alla giustificazione della notazione musicale rispetto al pentagramma. I parametri per la giustificazione sono diversi se si considera la parte direttoriale o una singola parte e se si considera la modalità di visualizzazione (a schermo o in stampa). La giustificazione può essere di tipo lineare o logaritmica.

La formattazione della musica è eseguita in modo automatica sia a livello di parte direttoriale che singola ed è realizzata per mezzo del linguaggio MILLA. Il linguaggio permette la scrittura di regole di formattazione che sono applicate sulla base del contesto quando è stata selezionata la modalità AUTO. Oppure, definendo dei riferimenti (il gambo della nota è rivolto verso l'alto o verso il basso) è possibile vincolare l'applicazione di regole specifiche.

Infine, nella formalizzazione XML la descrizione dei pedali del piano forte, dell'arpa, le fretboards, etc., utilizza le stringhe per codificare la posizione del pedale.

1.9 Da prototipo a sistema completo

Ripercorrendo i progetti dei sistemi OMR sviluppati in letteratura, si riscontra una grande difficoltà a migliorare le prestazioni ottenute nella realizzazione del prototipo. Anche in altre applicazioni l'operazione di *scaling up* (cioè il passagio al sistema completo) è abbastanza complicata, ma in campo musicale presenta degli aspetti che talvolta sembrano irrisolvibili.

Probabilmente molti dei problemi che si presentano nell'estensione dei progetti esistenti dipendono da un'errata impostazione nella fase di analisi e di progetto del sistema.

La complessità della notazione musicale e delle sue relazioni ha indotto molti autori a sviluppare delle applicazioni di carattere limitato con l'obiettivo di riconoscere, almeno all'inizio, spartiti piuttosto semplici. Di conseguenza la scelta dei metodi e degli strumenti di sviluppo spesso fatta con prospettive di ampio respiro, ma con l'attenzione puntata al problema specifico. Per esempio, in letteratura si trovano molti sistemi realizzati specificamente per il riconoscimento della musica monofonica: gli stessi autori però spesso trovano difficoltà ad applicare gli stessi metodi a spartiti di musica polifonica.

Fujinaga sostiene ([30]) che la relazione alla musica monofonica *"non è critica, perché un sistema OMR completo deve contenere un certo numero di sottoprogrammi, ognuno dei quali è progettato in maniera specifica per l'analisi di certe tipologie di spartiti"*. Quest'affermazione può essere anche condivisibile, a patto che l'analisi dei requisiti del sistema e la scelta delle metodologie di soluzione siano fatte sulla base del problema del riconoscimento musicale preso in tutta la sua complessità. Con queste considerazioni non si vuole affermare che la realizzazione di un *prototipo* non è utile nello sviluppo di un sistema OMR, ma che essa diventa infruttuosa se viene fondata (come spesso succede) su basi teoriche non adeguate ed estendibili.

Capitolo 2

Panoramica sui sistemi OMR

2.1 Introduzione

In questa sezione, sono discusse in maniera sintetica le ricerche più significative nel campo del riconoscimento automatico della musica. Anche se in ogni paragrafo si cerca di dare un'idea complessiva del lavoro di ogni autore, viene dato particolare risalto alle caratteristiche peculiari e innovative che vengono proposte: sono spesso tralasciati particolari legati all'implementazione per concentrarsi sulle fasi salienti del sistema proposto. Nella maggior parte dei casi è difficile stabilire un criterio evolutivo o di confronto, quindi si è preferito scegliere un ordine di presentazione pressochè cronologico: in questa disposizione è comunque interessante notare il riutilizzo e l'evoluzione di alcune idee e proposte risultate vincenti.

2.2 Prerau (1970)

Per la prima volta Prerau ([19]) introduce il concetto della *segmentazione* dell'immagine musicale con lo scopo di individuare gli elementi "primitivi" della simbologia. Egli utilizza dei metodi di "frammentazione e assemblaggio" per identificare le linee di pentagramma, isolare frammenti di notazione e ricomporre successivamente i simboli musicali. Il processo di riconoscimento proposto può essere schematizzato come segue:

- Scansione dei pentagrammi alla ricerca delle parti dei simboli musicali interne, superiori e inferiori alle linee. Attraverso l'estrazione delle parti individuate (frammentazione dei simboli) si ottiene in pratica la rimozione dei pentagrammi.
- Ricombinazione dei frammenti per riformare i simboli completi: le regole di assemblaggio risultano talvolta troppo semplici perché si basano solo sulla sovrapposizione orizzontale delle parti estratte (non hanno sempre successo).

- Misurazione delle dimensioni orizzontali e verticali di ogni simbolo (*bounding box dimensions*): Prerau sostiene che l'altezza e la larghezza sono caratteristiche più che sufficienti per l'identificazione dei simboli.

- Classificazione dei simboli attraverso il confronto delle dimensioni individuate con quelle di una tabella di riferimento costruita con la misurazione del maggior numero possibile di tipologie di simboli musicali. Di solito vengono trovate dalle tre alle cinque corrispondenze per ogni simbolo e solo attraverso test euristici si riesce a fare una corretta identificazione: questi test si basano sulla conoscenza della posizione e di altre proprietà caratteristiche di ogni simbolo, ma anche alcune informazioni sintattiche.

Questo sistema di riconoscimento (considerate anche le successive evoluzioni - [20], [21]) identifica un numero consistente di simboli musicali. Purtroppo, le prove sono state fatte solamente su piccoli esempi di spartiti musicali; il riconoscimento ha successo anche su simboli più complessi come chiavi, alterazioni, semiminime, crome (composte anche in gruppi con barre multiple), ma non su semicrome e accordi.

2.3 Fujimoto (1980)

All'inizio degli anni '80 fu sviluppato in Giappone un incredibile robot (Wabot-2) che leggeva la musica e suonava una tastiera. Per la prima volta si cercava di affrontare il problema OMR con un sistema On-line che in tempo reale leggeva, interpretava e produceva un output.

In questo sistema, per rispettare i vincoli *real time*, viene utilizzato un riconoscimento di tipo *locale* e diretto (viene saltata qualsiasi operazione di pre-elaborazione, come la rimozione del pentagramma). L'implementazione risulta molto interessante perché scinde il riconoscimento in due fasi, corrispondenti a due livelli di gerarchia diversi:

- Riconoscimento attraverso *template matching* dei simboli di *alto livello* (linee di pentagramma, teste di note, barre di battuta) che ricorrono molto spesso nello spartito musicale. Per garantire velocità di esecuzione e qualità dei risultati, l'implementazione del template matching viene fatta in **hardware**.

- Riconoscimento attraverso *template matching* dei simboli di *basso livello* (pause, aste, gambi, punti, alterazioni, ecc.), la posizione dei quali è legata ai risultati ottenuti in precedenza. L'implementazione viene fatta via **software** con algoritmi di ricerca di tipo locale.

In quest'ultima fase viene intrapresa una leggera correzione sintattica, rivolta in particolare alla distinzione tra note piene e vuote. Questa fase non può essere sviluppata come sarebbe necessario, perché i vincoli temporali non permettono un'analisi sintattica e semantica approfondita e a largo raggio.

Il risultato più importante del sistema è la velocità con cui viene esaminata una pagina di musica (dai 10 ai 15 secondi) con un riconoscimento eccellente (solo su particolari spartiti per organo con 3 pentagrammi e notazioni piuttosto semplici). Altri autori hanno continuato lo sviluppo del sistema, anche passando ad un approccio Off-line, con risultati buoni su spartiti di diversa natura.

È importante sottolineare lo sforzo fatto, in particolare da Itagaki, Isogai, Hashimoto e Ohteru ([33]), per la definizione di un linguaggio comune di rappresentazione dei vari tipi di informazione musicale: quella legata alla riproduzione, alla stampa o ad esigenze particolari come braille, danza e teatro.

2.4 Aoyama e Tojo (1982)

Il sistema proposto e sviluppato da Aoyama e Tojo ([22]) è strutturato in tre stadi:

- Input
- Segmentazione
- Riconoscimento e controllo sintattico

Nella fase di input viene digitalizzata l'immagine, si ottiene l'altezza degli spazi e delle linee del pentagramma, con l'individuazione di quest'ultime. Nella fase di segmentazione si procede con la rimozione del rigo e la divisione dei simboli utilizzando l'*analisi delle componenti connesse*. Da ultimo, i simboli divisi vengono classificati e verificati.

In merito allo spartito si osserva che:

1. è bidimensionale
2. l'informazione spaziale è importante
3. linee, immagini e caratteri sono mischiati, e la loro posizione non è specificata
4. a causa della sottigliezza delle linee è necessaria un'alta risoluzione nella scansione
5. simboli con lo stesso significato possono avere rappresentazione differente
6. i simboli sono posizionati in accordo con regole sintattiche di tipo spaziale

Dal punto di vista del riconoscimento, gli spartiti contengono simboli che sono:

- adatti per il confronto con modelli;
- adatti per un metodo di analisi strutturale.

Input

Lo spartito in ingresso è assunto essere una stampa e privo di simboli incompleti, ma può essere di qualsiasi dimensione (entro certi limiti) e i pentagrammi possono essere inclinati o leggermente discontinui. Il sistema usa un drum scanner con scala di grigi a 8-bit con risoluzione di 254 d.p.i..

L'immagine viene esaminata due volte. Nella prima scansione, si ottengono gruppi di linee di scansione verticale. Le linee del pentagramma sono individuate nel modo seguente:

1. Utilizzando un istogramma si ottiene la conversione in binario delle linee di scansione.
2. Si considera la proiezione sull'asse verticale di ogni gruppo; se ciascun gruppo contiene n linee, le proiezioni con n o n-1 pixel vengono considerate candidate ad essere linee del pentagramma.
3. Utilizzando il risultato del passo 2 e creando un istogramma di linee nere e bianche da quelle candidate, si ottengono l'altezza degli spazi e delle linee del pentagramma.
4. I candidati ad essere linee sono scelti utilizzando le informazioni ottenute al passo 3.

Nella seconda scansione, a causa della grande quantità di informazione coinvolta, ogni pentagramma viene considerato separatamente. Nella finestra di ogni pentagramma l'immagine è codificata in base alla lunghezza dei tratti verticali (questa è la direzione in cui la pagina è fisicamente scansionata nel loro drum scanner).

Segmentazione

Il sistema rimuove la maggior parte del rigo, ma per evitare una successiva segmentazione dei simboli, come le note da 2/4 e le code, le regioni del rigo a sinistra e a destra del tratto adiacente al simbolo vengono segnate come da non cancellare. Alla fine, i tratti che attraversano la posizione del rigo e che hanno la larghezza del rigo vengono rimossi.

Le teste di nota nere vengono cercate sulle linee, o tra di esse, con un modello e, se trovate, sono rimosse, ma solo momentaneamente, poiché il principale obiettivo di questa sezione è di trovare i vuoti (negli uncini, note da 2/4 e interi). Una volta trovati, questi simboli possono essere marcati, cosicché, quando viene rimosso il resto del rigo, i simboli non siano frammentati. I vuoti vengono riconosciuti con un sistema che cerca piccoli tratti orizzontali bianchi tra le linee del pentagramma. Una volta che questi sono segnati, le teste sono rimesse e il rigo finalmente rimosso.

L'immagine risultante è segmentata attraverso l'analisi delle componenti connesse. Si utilizzano l'altezza e la larghezza della bounding box di ogni segmento per dividere approssimativamente le componenti connesse in dieci gruppi. L'altezza e la larghezza vengono normalizzate utilizzando l'altezza del pentagramma.

Riconoscimento e controllo sintattico

Nel gruppo che comprende le note con uncino e quelle con travi, quest'ultimi vengono separati dalle teste con la rimozione delle regioni sottili (le gambe). L'analisi della configurazione delle note è realizzata considerando elementi come la larghezza, l'altezza, il baricentro, il rapporto area/aera della bounding box, il conteggio delle teste, il conteggio degli uncini e H-type (una qualsiasi tra 11 configurazioni testa-gamba).

In un altro gruppo, quello che comprende alterazioni e pause, viene utilizzato un albero classificatore che, per separare i membri di questa classe, si basa sulla lunghezza dei tratti orizzontali e verticali. Per il riconoscimento di simboli composti (es. punto a corona, chiave di basso, ecc.) viene impiegata una tavola contenente informazioni riguardanti la posizione relativa delle componenti.

Infine attraverso l'uso di regole sintattiche, inerenti alla posizione dei simboli e il numero costante di quarti in una misura, si esegue un secondo controllo sul risultato del riconoscimento.

Le regole di tipo spaziale sono:

1. le alterazioni in chiave appaiono dopo il simbolo di chiave
2. se vi è una chiave di violino e le alterazioni in chiave iniziano con un diesis, il diesis deve essere sulla linea più in alto
3. le alterazioni appaiono alla sinistra della testa della nota

Benché non sia stata implementata, viene suggerita la possibilità di riconoscere i segni di espressione (*pp*, *andante*, *a tempo*, *ecc.*) tramite il conteggio dei caratteri.

2.5 Mahoney (1982)

Con l'obiettivo di progettare un sistema realizzabile in pratica, Mahoney ([2]) concentra tutta la sua attenzione alla fase iniziale di ricerca grafica dei simboli musicali. Egli sostiene con forza che la realizzabilità è legata alla semplicità che caratterizza ogni task in cui viene diviso il processo: la distinzione tra i simboli e le primitive che li compongono *suddivide* e *semplifica* il riconoscimento. Inoltre introduce una nuova tecnica di ricerca delle primitive fondata su questa convinzione: "Non c'è bisogno del contesto per il riconoscimento delle primitive e tutte le considerazioni sintattiche sono lasciate alla routine di analisi, l'unico

scopo della quale è quello di trovare i rapporti tra i simboli già classificati e gli oggetti ricavati dalla loro combinazione".

Partendo da questo concetto egli utilizza ripetutamente la seguente strategia per l'isolamento delle primitive: prima, attraverso l'analisi dell'immagine, raccoglie una serie di candidati per una o più tipologie di simboli, poi utilizza i descrittori (o modelli) di ogni tipo di simbolo per individuare i candidati corrispondenti. Nell'analisi dello spartito spesso vengono ripetute le ricerche basate sullo stesso modello (es. il cerchio) in presenza e in assenza delle linee che "toccano" i simboli (linee di pentagramma, gambi delle note). In questo senso viene fatta una distinzione tra la rimozione delle *linee reali* (parti delle linee che non toccano i simboli) e la rimozione delle *linee ideali* (il pentagramma e le altre linee).

I passi del processo di riconoscimento possono schematizzarsi così:

- Costruzione dei *candidati* (o modelli) per la ricerca delle primitive e analisi "tematica" dell'immagine nel seguente ordine:

  1. Le linee di pentagramma, modellate con descrittori le cui specifiche principali sono lo spessore, la lunghezza, la distanza tra le righe, ecc.
  2. Le altre linee orizzontali.
  3. Le linee verticali.
  4. Le teste delle note, descritte con linee chiuse a forma di cerchio: in questo caso si preferisce ripetere la ricerca con o senza linee di pentagramma.
  5. Analoga ricerca di molti altri simboli, fatta di solito dopo la rimozione del pentagramma: gambi, barre dei gruppi note, alterazioni, ecc.

- Ricostruzione e riconoscimento dei simboli musicali attraverso una procedura di analisi che utilizza informazioni sintattiche.

La realizzazione di questo progetto è solo parziale ed è particolarmente concentrata sul riconoscimento delle primitive: viene sviluppato anche una modalità interattiva per la costruzione e la correzione dei modelli della simbologia in modo da affinare l'identificazione.

Sui pochi esempi riportati i risultati sono buoni, anche se questo metodo sottolinea la necessità dell'intervento umano nella pre-calibrazione e nella costruzione dei descrittori.

2.6 Clarke (1988)

Clarke ha concentrato tutto il suo lavoro nello sviluppo di un sistema che potesse funzionare su un personal computer IBM compatibile, equipaggiato con il sistema operativo DOS ([43]). La strategia scelta è quella di operare un riconoscimento di carattere locale (un

pentagramma alla volta) e di utilizzare, per la classificazione dei simboli, metodi non molto onerosi dal punto di vista computazionale.

Il procedimento proposto può essere schematizzato nelle seguenti fasi:

- Identificazione e rimozione completa del pentagramma con l'utilizzo di un algoritmo veloce, ma sensibile a disturbi. Alla fine dell'operazione vengono esaminate le vicinanze delle linee per verificare se ci sono simboli attraversati dal pentagramma.
- La classificazione iniziale dei simboli viene fatta in base all'altezza e alla larghezza. Successivamente, invece di utilizzare un *template matching* completo (computazionalmente troppo pesante), vengono confrontate alcune righe e alcune colonne particolari del simbolo in questione con un modello predefinito.

Questo sistema, ancora incompleto, è interessante principalmente per lo sforzo fatto nella risoluzione del problema OMR in sistemi con risorse limitate. I risultati riportati si riferiscono a spartiti molto semplici e non hanno particolare rilevanza.

2.7 Roach e Tatem (1988)

Le novità proposte da questi due autori sono principalmente due: un nuovo metodo molto funzionale per l'identificazione delle parti delle linee del pentagramma non a contatto con i simboli e un approccio che prevede l'utilizzo della conoscenza musicale nel processo di segmentazione e riconoscimento ([42]).

La più importante è certamente la seconda che si basa sulla convinzione che le proprietà, le caratteristiche e le regole della simbologia siano utili non solo per il riconoscimento dei simboli musicali, ma anche per l'identificazione delle primitive che li compongono. Nel metodo sviluppato si possono distinguere le seguenti fasi:

- Identificazione e rimozione delle parti "scoperte" delle linee del pentagramma (non viene rimosso tutto il pentagramma per non distorcere o cancellare parte dei simboli musicali). Il metodo consiste nel far scorrere sull'immagine una finestra di dimensioni adeguate alla ricerca di cammini rettilinei con una determinata *angolazione*. Una volta individuati i cammini, la scelta viene fatta verificando lo spessore delle linee.
- Identificazione delle primitive in un procedimento organizzato in due momenti:
    1. Ricerca procedurale delle primitive fondamentali (teste delle note, gambi, barre dei gruppi).
    2. Analisi finale attraverso un codice realizzato in Prolog, il quale permette l'utilizzo della conoscenza musicale per l'isolamento delle primitive e la ricomposizione dei simboli.

- Classificazione dei simboli musicali.

L'ingresso utilizzato dagli autori è uno spartito di musica manoscritta e, a complicare ulteriormente le cose, la risoluzione di digitalizzazione scelta (100 dpi) risulta davvero insufficiente per una buona rappresentazione dei simboli musicali. Dunque, è molto difficile valutare i risultati ottenuti nei pochi test fatti dagli autori, i quali sostengono che, principalmente per l'utilizzo della conoscenza a partire dalla fase grafica, i risultati sono soddisfacenti e molto migliorabili con la musica stampata e a risoluzioni maggiori.

2.8 Carter (1988)

Il contributo che, dal 1988 in poi, Carter ha dato allo studio del problema OMR è molto importante ([7], [8], [9], [10], [11], [12], [13], [14]). Oltre a concentrarsi su questioni specifiche (in particolare sull'identificazione del pentagramma), si è impegnato nel coordinamento delle ricerche esistenti, con l'obiettivo di definire alcuni standard di riferimento sull'OMR.

Senza dubbio, il suo contributo più importante riguarda il processo di segmentazione delle immagini, basato su un metodo che utilizza il *Line Adjacency Graph (LAG)* trasformato: l'immagine musicale viene prima analizzata verticalmente alla ricerca di singoli percorsi verticali di pixel neri chiamati (segmenti) e poi orizzontalmente operando la costruzione del LAG trasformato. I nodi del grafo corrispondono a sezioni (unioni di segmenti adiacenti che si sovrappongono verticalmente), gli archi corrispondono a giunzioni (sovrapposizioni di diversi segmenti in una colonna adiacente). Il grafo ottenuto da queste operazioni viene analizzato e elaborato alla ricerca delle linee di pentagramma e dei simboli ad esse collegati.

Carter ha utilizzato questa tecnica ideata da Pavlidis adattandola alle immagini musicali e ha ottenuto risultati molto importanti:

1. Identificazione delle parti "scoperte" delle linee di pentagramma, evitando la distorsione dei simboli che da esse vengono attraversati o toccati (tangenti). La marcatura di tali parti di pentagramma mette in evidenza i *paragrafi* che contengono i simboli musicali o un loro raggruppamento.

2. Identificazione del pentagramma anche se l'immagine musicale è sottoposta a una rotazione maggiore di 10^o.

3. Identificazione del pentagramma anche se le linee stampate presentano lievi curvature o imperfezioni.

4. Identificazione del pentagramma anche se le linee sono affette da variazioni locali dello spessore.

5. Rimozione di piccoli elementi grafici di rumore su tutta l'immagine.

Nelle condizioni di difficoltà appena descritte gli altri metodi di identificazione e rimozione delle linee di pentagramma non riescono a ottenere gli stessi successi della tecnica proposta da Carter e spesso, provocando la cancellazione parziale o totale di alcuni simboli, determinano una notevole perdita di informazione.

Le fasi principali che caratterizzano i sistemi OMR progettati da Carter sono:

- Applicazione del metodo basato sul Line Adjacency Graph trasformato. I risultati che si ottengono sono:
  1. Identificazione, anche in condizioni molto difficili, delle parti scoperte delle linee di pentagramma.
  2. Isolamento dei *paragrafi* che contengono singoli simboli musicali oppure frazioni o composizioni di simboli che si toccano o si sovrappongono.
- Classificazione degli oggetti ottenuti dalla segmentazione in base alle loro dimensioni, al numero e all'organizzazione dei paragrafi costituenti. Per il riconoscimento di simboli tangenti o sovrapposti vengono proposti algoritmi specifici; un maggiore sforzo viene fatto per gli oggetti che non sono fisicamente legati al pentagramma.
- Nel sistema proposto nel '90 ([9]) viene generato un output in un formato ASCII compatibile con SCORE.

La procedura di segmentazione messa a punto dall'autore e sperimentata in vari progetti può costituire una soluzione pratica e funzionale alla prima fase del processo OMR: i risultati sono molto buoni anche in condizioni difficili (separazione pressoché perfetta tra pentagramma e simboli musicali). Invece, non sono sviluppate allo stesso livello le parti del sistema che devono operare il riconoscimento degli oggetti ottenuti dalla segmentazione; i risultati non sono di grande rilievo. Deve essere sottolineato lo sforzo che in alcuni progetti viene fatto per la creazione di un output di salvataggio nei formati più diffusi o compatibili.

2.9 Fujinaga, Alphonce e Pennycook (1988)

La novità più importante introdotta da Fujinaga ([30]), e poi ripresa anche da Alphonce e Pennycook ([31]), è il forte utilizzo delle proiezioni in situazioni diverse e a vari livelli di dettaglio grafico. Egli propone un approccio secondo il quale non è necessario rimuovere il pentagramma, ma è sufficiente identificare la sua posizione (in questo la proiezione Y può essere molto efficiente); in secondo luogo, le informazioni fornite da una serie di proiezioni, prima grossolane e poi dettagliate, sono sufficienti per individuare i simboli

e ricavarne le caratteristiche peculiari; infine, l'utilizzo di conoscenza sintattica sia nella fase di individuazione che di classificazione dei simboli è complementare all'analisi per proiezioni e innalza il livello di astrazione del riconoscimento.

Fujinaga ha anche dato un contributo teorico importante, punto di partenza per molti altri autori, affermando che la notazione musicale può essere formalizzata con una grammatica "... *context-free* e LL(k); questo è in effetti quello che permette ai musicisti (top-down parser) di leggere la musica in maniera così efficiente." Egli insiste sul fatto che, un approccio puramente sintattico, nel quale il contesto non è preso in considerazione, ha molte limitazioni e suggerisce di introdurre informazioni semantiche all'interno della grammatica.

Nelle implementazioni realizzate prima da Fujinaga e poi in collaborazione con Alphonce e Pennycook, si distinguono le seguenti fasi:

- Identificazione del pentagramma attraverso l'analisi della proiezione Y dello spartito musicale: i gruppi (di cinque picchi se l'immagine è diritta) che danno graficamente un contributo più consistente, ripetitivo sull'asse verticale, indicano il pentagramma. A questo punto viene diviso lo spartito in "fette" orizzontali contenenti un singolo pentagramma (solo per musica monofonica).

- Identificazione dei simboli sul pentagramma attraverso una proiezione X: i valori che superano il rumore di background (linee di pentagramma) indicano la presenza di simboli musicali. A supporto della individuazione della presenza di simboli vengono utilizzate alcune regole sintattiche legate principalmente alla posizione della notazione.

- Classificazione dei simboli: una volta individuato il simbolo vengono calcolate le proiezioni X e Y in modo più dettagliato e da esse vengono ricavate le caratteristiche di classificazione (larghezza, altezza, area, numero di picchi della proiezione X). Un miglioramento proposto è quello di introdurre anche la derivata prima e seconda dei profili delle proiezioni.

I risultati ottenuti sono piuttosto buoni relativamente agli esempi proposti (spartiti non complessi e di musica monofonica). Con notazioni più complesse la classificazione a base di proiezioni non è sufficiente, ma essa si rivela uno strumento utile nella identificazione grossolana dei simboli musicali (proiezione X sul pentagramma).

2.10 Modayur (1990)

Modayur ([44]) nei sui lavori propone una tecnica completamente nuova per l'identificazione del pentagramma e per il riconoscimento dei simboli: la **morfologia matematica**.

Essa fornisce la teoria e gli strumenti per l'analisi delle forme; infatti, la notazione musicale è ricca di combinazioni complesse, ma di simboli con forme ben definite. Le operazioni morfologiche utilizzate (*dilation*, *erosion*, *opening*, *closing* e *hit/miss transformation*) insieme ad alcuni simboli strutturati standard (*line*, *disk* e *box*) hanno lo scopo di estrarre ogni singolo simbolo musicale nella sua rappresentazione completa.

Viene comunque sottolineato che una componente essenziale del processo è l'utilizzo della conoscenza musicale per verificare i risultati dell'analisi grafica.

Il sistema sviluppato è formato da un modulo a basso livello di analisi visiva (pentagramma e simboli) e un modulo ad alto livello di analisi conoscitiva (correzione e interpretazione); si possono distinguere le seguenti fasi:

- Identificazione delle linee del pentagramma. Quest'operazione viene fatta utilizzando sia tecniche morfologiche, che metodi classici di elaborazione delle immagini.

- Riconoscimento dei simboli attraverso una sequenza di operazioni morfologiche. In un primo momento sono estratte le caratteristiche primitive di ogni simbolo; successivamente, per l'identificazione, vengono utilizzate le relazioni topologiche conosciute tra le caratteristiche dei simboli.

- Elaborazione di alto livello sulla base di regole sintattiche (verifica dei risultati del riconoscitore morfologico).

- Produzione di un output in formato ASCII con una rappresentazione proprietaria.

Visto che le operazioni morfologiche riescono ad identificare correttamente molti simboli e che esse possono essere implementate anche in hardware, il sistema ha buone prestazioni nel riconoscimento e vicine ai vincoli real-time in velocità.

Nell'ultima realizzazione il sistema è stato migliorato introducendo una *strategia di attenzione selettiva*, per guidare il riconoscitore simbolico in regioni specifiche dell'immagine alla ricerca di determinati simboli ([47]). Questo ha provocato un significativo incremento della velocità del processo e una riduzione degli errori di riconoscimento.

2.11 Couasnon e Camillerap (1990)

Come altri autori Couasnon e Camillerap ([16], [17]) partono da un principio basilare: nel processo di riconoscimento l'utilizzo della *conoscenza musicale* è molto importante. In più essi sostengono che, di solito, queste informazioni sono sfruttate in modo molto riduttivo e che non è sufficiente, come succede nella maggior parte dei casi, utilizzarle solo ad alto livello (correzione degli errori a posteriori). Infatti, la conoscenza musicale non deve essere solo uno strumento di verifica, ma deve controllare l'intero processo.

La formalizzazione della conoscenza musicale viene basata sulla definizione di una *grammatica* che descrive le regole sintattiche e rappresenta il contesto, introducendo anche un livello grafico (l'unico precedente in letteratura è Andronico e Ciampa - [1]). Un vantaggio dell'utilizzo della grammatica è la separazione tra la parte operativa del sistema e la definizione delle regole musicali, in modo da rendere facile l'adattamento ad un altro tipo di documento strutturato (semplice ridefinizione della grammatica con lo stesso *parser*).

Una conseguenza importante di questa impostazione è che anche le fasi di segmentazione e etichettatura delle primitive sono controllate dalla grammatica scelta: quindi, le primitive non vengono etichettate nella fase di estrazione, ma si subordina l'assegnazione delle etichette alla verifica della consistenza delle regole sintattiche.

Infine gli autori affrontano due problemi aperti dell'OMR: gli oggetti che si toccano (o sovrappongono) e gli oggetti graficamente rovinati (rotti). Le soluzioni proposte si basano su un metodo di adattamento automatico della descrizione degli oggetti, in modo tale da poter utilizzare gli stessi strumenti conoscitivi descritti sopra.

Le fasi salienti del processo di riconoscimento proposto sono:

- Segmentazione e etichettatura delle primitive estratte. Già in queste operazioni entra in gioco la conoscenza musicale per il controllo della corrispondenza degli elementi grafici trovati con le regole sintattiche.

- Ricostruzione e classificazione dei simboli musicali. Questa fase diventa immediata dato che è già stata verificata la consistenza della posizione e della sequenza delle primitive di base.

- Verifica e correzione dei risultati con controllo del valore delle note e allineamento delle figure.

Un elemento di grande importanza è il fatto che, l'estensione della conoscenza musicale al controllo di tutto il processo, permette una semplice evoluzione del sistema verso il riconoscimento di spartiti più complicati (*scaling up*), in quanto, è sempre possibile definire nuove grammatiche per formalizzare notazioni di qualsiasi complessità.

I risultati ottenuti su spartiti completi con polifonia (2 voci) sono abbastanza buoni e, vista la struttura del sistema, facilmente estendibili a problemi ancora più difficili.

2.12 Bainbridge (1990)

Bainbridge dal '91 ha iniziato ad analizzare il problema della "comprensione" automatica della musica con un attenzione particolare al processo conoscitivo umano. Nel lavoro che qui viene brevemente riportato ([4]), viene progettato un ipotetico riconoscitore musicale con prestazioni ottimali.

La ricchezza delle notazioni musicali, la loro evoluzione, i "dialetti" e le personalizzazioni degli autori e delle case editrici evidenziano la *natura dinamica* del problema OMR. Nel processo naturale di comprensione della musica si possono individuare due fasi: il riconoscimento delle forme grafiche e l'applicazione della conoscenza musicale per ricavarne il significato. Dunque, un sistema OMR ottimale dovrebbe essere composto da un **Drawing Package**, nel quale vengano descritte le forme grafiche *primitive* alla base della simbologia musicale, e da uno **Specially Designed Music Language**, che fornisca una struttura per esprimere la "conoscenza astratta" della notazione musicale. Entrambi i moduli devono avere una forma flessibile e dinamica per essere facilmente adattati a nuove notazioni musicali oppure a nuove interpretazioni della simbologia.

Inoltre Bainbridge ritiene che il linguaggio deve prevedere due meccanismi di funzionamento: un metodo per specificare le configurazioni legali delle primitive alla base della simbologia musicale; un metodo per esprimere la semantica musicale. Per il primo problema egli propone di assumere nel linguaggio elementi come le relazioni spaziali tra primitive, le restrizioni spaziali e le regole grafiche, la combinazioni di espressioni booleane e così via, seguendo un modello gerarchico (object oriented). Per l'espressione della semantica, siccome non esiste un formato musicale standard di interscambio, propone di implementare una rappresentazione interna strutturata (Object Oriented e Abstract Data Type) interfacciabile ai vari formati esistenti.

Nel sistema sviluppato in parte troviamo realizzati gli aspetti appena descritti:

- Nel modulo grafico (*drawing package*) vengono pienamente supportate le primitive dinamiche. Le forme geometriche specificate nel pacchetto possono essere direttamente importate nel programma di identificazione e la tolleranza di riconoscimento può essere specificata in modo dinamico.

- Il linguaggio definito è invece più semplice delle specifiche richieste in precedenza, anche se le configurazioni legali delle primitive sono già definite in modo dinamico. L'algoritmo di riconoscimento, invece, (per la semplicità del linguaggio) è funzionante, ma modesto.

- Viene implementata una struttura dati interna per rappresentare il significato musicale delle forme grafiche identificate. Ad esso sono associati alcuni moduli di traduzione verso i formati più comuni.

2.13 Kato e Inokuchi (1990)

Nel complesso sistema realizzato da Kato e Inokuchi ([3]) vengono introdotte delle novità molto interessanti. Vista la complessità di alcune situazioni (connessioni e sovrapposizioni tra simboli) essi sostengono che la conoscenza musicale è indispensabile per ottenere un

buon livello di riconoscimento. Inoltre, dato che ogni simbolo musicale differisce dagli altri in dimensione, posizione, frequenza di apparizione e importanza, è difficile trovare un singolo metodo per riconoscere tutti i simboli: siccome ci sono vari metodi di *pattern recognition* è opportuno scegliere, in ogni situazione diversa, quello più appropriato.

L'idea principale, utilizzata nel riconoscimento, si basa su un processo che integra il pattern recognition con l'analisi semantica: siccome nella fase di analisi grafica ci si deve confrontare con varie situazioni inaspettate (sovrapposizioni, contatti, linee deteriorate, rumori), gli autori propongono di affiancare al pattern recognition un approccio *top-down* che sfrutta la conoscenza e le regole della notazione musicale. Infatti, sebbene negli spartiti si possano trovare soluzioni notazionali molto varie, esse si basano su un numero ben definito di *primitive*, la cui combinazione può ricreare gran parte della simbologia musicale.

Comunque, la novità più interessante è la LAYERED WORKING MEMORY: Kato e Inokuchi usano un insieme di moduli specializzati che comunicano attraverso una comune memoria di lavoro. In questa memoria l'informazione musicale viene rappresentata a cinque livelli di astrazione:

1. Livello dell'immagine digitale (pixel).
2. Livello delle *primitive*: teste di note, gambi, alterazioni, pause, ecc. Il modulo operativo corrispondente opera un **estrazione di primitive**.
3. Livello dei *simboli musicali*: note e pause che sono la sintesi della combinazione delle primitive del secondo livello. Il modulo operativo corrispondente opera una **sintesi dei simboli**.
4. Livello del *significato* di ogni simbolo: altezza, durata delle note, ecc. Il modulo operativo corrispondente opera un **riconoscimento dei simboli**.
5. Livello di *interpretazione* dell'intera battuta. Il modulo operativo corrispondente opera un'**analisi semantica**.

Il lavoro di ogni modulo è regolato da una *soglia* di controllo della corrispondenza del "matching". Le parti dell'immagine di alta qualità vengono riconosciute con una soglia molto restrittiva. Successivamente, con precisione minore, viene tentato il riconoscimento degli altri simboli. È interessante notare che è previsto un processo di **retroazione**, secondo il quale, se a livelli più alti sono formulate ipotesi errate, ai livelli inferiori viene richiesta la ripetizione delle operazioni con parametri diversi.

Infine, viene introdotto un nuovo metodo di identificazione e rimozione dei pentagrammi, che si basa sull'analisi *run-length* di alcune colonne dell'immagine. L'istogramma 0-pixel individua lo spazio tra i pentagrammi, l'istogramma 1-pixel lo spazio tra le righe del pentagramma. La rimozione viene fatta in modo completo su tutte le linee del pentagramma.

Nel sistema realizzato per il riconoscimento della musica per pianoforte ([3]) possiamo identificare le seguenti fasi:

- Identificazione delle linee di pentagramma.
- Identificazione delle barre delle battute. Questa fase permette successivamente di impostare il lavoro in maniera *locale*, una battuta per volta.
- Eliminazione delle linee del pentagramma.
- Identificazione dei simboli attributivi (chiave, tonalità, tempo).
- Riconoscimento dei simboli delle note (testa, gambi, ecc.).
- Identificazione dei simboli globali, i quali oltrepassano i limiti delle battute (legature, dinamiche).
- Unificazione dei risultati e salvataggio in un formato proprietario.

La struttura di questo sistema permette un riconoscimento di spartiti anche molto complessi con risultati davvero notevoli: con "Per Elisa" di Beethoven si ha una percentuale di riconoscimento del 95,6%. Questo deriva, principalmente, dalla costruzione stratificata dei livelli di astrazione dell'informazione musicale e dal meccanismo che permette operazioni di correzione e di rianalisi successive dello spartito.

2.14 Martin e Bellisant (1991)

Nel sistema proposto ([23]) viene utilizzata una rete neurale sia per la rimozione del rigo che per la classificazione degli oggetti con componenti connesse.

Il problema dell'inclinazione del rigo viene risolto introducendo il concetto di *corda*. Una corda di orientazione θ in P è il segmento di inclinazione θ inscritto in un componente connesso C contenente P (vedi fig.2.1)

La lunghezza della corda L(P,θ) è definita come la distanza tra i due punti del confine di C, dati dall'intersezione con la corda. Nel continuo ci sarebbe un numero infinito di corde di θ per P, ma il numero è finito nel discreto e se si limita θ a piccole variazioni (pochi gradi) il numero viene notevolmente ridotto.

Nell'ipotesi che l'intera pagina abbia un'inclinazione di un certo numero di gradi, tutti i punti nella colonna centrale dell'immagine sono considerati P e vengono esaminati pochi valori di θ per trovare P_0 e θ_0 tali che L(P_0,θ_0) sia massima. La rotazione di -θ_0 centrata in P_0 è applicata all'intera immagine per eliminare l'inclinazione. La lunghezza della corda è calcolata utilizzando un efficiente algoritmo di line-tracing.

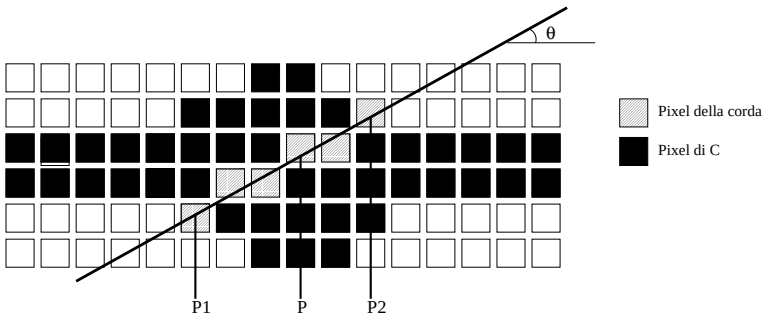


Figura 2.1: Corda di orientazione θ in P (Martin e Bellisant 1991)

Attraverso una proiezione sull'asse y di tutta l'immagine, corretta dall'inclinazione, si determina approssimativamente la posizione del rigo. Con questa informazione ha inizio il processo di cancellazione delle linee del pentagramma che non sono coperte da simboli musicali. Vengono calcolati il limite superiore ed inferiore di ogni linea, permettendo così una maggiore accuratezza nella valutazione della posizione delle teste delle note.

Per la cancellazione del rigo, si esegue una scansione per colonna; se si trova un tratto nero, vicino alla posizione dell'istogramma della proiezione sull'asse y, con larghezza simile e non appartenente ad un simbolo, la si cancella. Per riconoscere se un punto appartiene o meno ad un simbolo, viene utilizzata una rete neurale multi-livello con 228 input e retropropagazione del gradiente.

La classificazione delle note avviene attraverso un sistema basato su regole ad hoc che utilizza confronti con modelli di forma ellittica. Per contare il numero di uncini e travi attaccati alle gambe viene utilizzato il metodo Sonde verticale e orizzontale. I rimanenti simboli sono classificati assottigliandoli e processandoli con un'altra rete neurale. Dopo aver effettuato una classica operazione di assottigliamento, alcuni punti sono segnati come estremi, giunzioni o "bending". Il rettangolo circoscritto, con le dimensioni normalizzate, viene suddiviso arbitrariamente in finestre. Si fa uso di un set di variabili binarie come input per la rete. Si definiscono due classi di variabili:

1. *(t,w)* con t che può essere un estremo, una giunzione o un punto di bending e w una finestra

2. *(w_i,w_j)* con *$i \neq j$*, per ogni i e j, in cui:

   - *(w_i,w_j)=1* se almeno un segmento dello scheletro ha uno dei suoi estremi in w_i e l'altro in w_j
   - *(w_i,w_j)=0* altrimenti

La rete neurale utilizzata sembra includere un algoritmo di costruzione di un decision-tree per includere celle nascoste specializzate, connesse solo ad alcune celle di input, come anche celle nascoste totalmente connesse, quelle connesse a tutte le celle di input.

Le conlusioni dell'autore affermano che nonostante la percentuale di riconoscimento della rete sia del 96.5%, "la prestazione della classificazione è meno impressionante quando viene confrontata con metodi statistici; abbiamo notato, come altri prima, che un classificatore nearest-neighbour è di solito sufficiente per raggiungere la stessa percentuale di riconoscimento. Ma si deve sottolineare che il nearest-neighbour può anche essere realizzato con reti multi-livello".

2.15 McGee e Merkley (1991)

Questo sistema ([34]) è rivolto alla notazione del canto gregoriano con neumi quadrati. Il pentagramma di quattro linee viene eliminato ricercando linee orizzontali sottili "sufficientemente lunghe". Allo stesso tempo si esegue il raddrizzamento. La classificazione avviene utilizzando un insieme di rettangoli che racchiudono ciascun neuma. Gli autori hanno anche sperimentato un "thin-line coding", originariamente sviluppato per l'identificazione delle impronte digitali, per la classificazione dei neumi. L'input in ingresso deve avere una risoluzione di 300 d.p.i..

2.16 Miyao (1992)

I due elementi interessanti di questo sistema ([45]) sono una grammatica per la notazione musicale, utilizzata nella fase di riconoscimento, e la rimozione del rigo che avviene, diversamente da molti sistemi, dopo che le note sono state estratte (incluse le teste, le gambe, gli uncini e le travi).

Vengono fatte tre osservazioni riguardanti le carattestiche della notazione musicale:

1. Le posizioni della chiave, delle alterazioni in chiave e del tempo possono essere dedotte dalla posizione del pentagramma e delle stanghette.

2. Gli altri simboli, inclusi i punti, le legature, gli accenti, gli staccati, le corone hanno una posizione relativa rispetto alle gambe, alle stanghette e alle note.

3. Le dimensioni dei simboli sono relative all'altezza dello spazio del pentagramma.

Il sistema determina la posizione dei pentagrammi, quindi procede con la ricerca e la rimozione delle note. Dopo l'eliminazione del rigo, i simboli rimanenti sono raggruppati in modo approssimativo in base alla loro dimensione e posizione, e la classificazione avviene utilizzando le proprietaà strutturali o confrontandoli con modelli.

Viene utilizzata la trasformata lineare di Hough per trovare il rigo, sulla base delle altezze della linea e dello spazio calcolate dalla lunghezza dei tratti verticali neri e bianchi. Le stanghette che attraversano due pentagrammi sono individuate utilizzando una proiezione lungo l'asse x. Le teste delle note piene sono estratte mediante una maschera rettangolare (altezza dello spazio del pentagramma per la larghezza della testa della nota, che è 2 volte l'altezza dello spazio; la posizione è rilevata con la stessa maschera. Le teste di nota bianche vengono distinte dalle teste di note piene dal numero di pixel bianchi nell'area della maschera, le note da 2/4 o gli interi, invece, attraverso un confronto con modelli.

Le note candidate, trovate al di fuori del pentagramma, sono verificate cercando i tagli, se non si trovano la candidatura viene revocata. Data una testa di nota, la ricerca delle gambe avviene esplorando i bordi sinistro e destro, se non viene identificata l'esistenza di alcuna gamba, viene eliminata la candidatura della nota. Le regola della notazione, come ad esempio "non tre gambe per una testa di nota", sono applicate per assicurarsi che i simboli riconosciuti siano grammaticalmente corretti. Il numero di uncini e travi viene determinato contando il numero di tratti neri vicini alle gambe.

Dopo la rimozione del rigo, le componenti connesse vengono raggruppate per altezza, larghezza e posizione relativa dalla linea di mezzo del pentagramma. Tutte le misurazioni sono normalizzate con l'altezza dello spazio del pentagramma.

I simboli di dimensione fissa raggruppati in modo approssimativo vengono successivamente classificati utilizzando modelli 6x6. Il simbolo viene diviso in una maglia 6x6 e ciascuna di queste viene rappresentata dal rapporto del numero dei pixel neri e bianchi, i trentasei numeri sono quindi rappresentati come un vettore e comparati con i vettori prototipi utilizzando la misura della distanza euclidea. I simboli sconosciuti vengono classificati come il più vicino prototipo sopra una certa soglia. Quelli non classificati vengono riconnessi inserendo le linee dello spartito che erano state rimosse, e quindi si ripete il calcolo della distanza. Per quanto riguarda i simboli di dimensioni variabili, come le legature e le "forcelle" della dinamica, vengono utilizzate le lunghezze dei tratti verticali e orizzontali. Da ultimo si fa uso di regoli spaziali per portare a termine le decisioni della classificazione.

L'accuratezza riportata da questo sistema, con in ingresso una scansione con risoluzione di 240 d.p.i., è compresa tra il 93% e il 98%, con un tempo di esecuzione variabile da i 3 ai 20 minuti per pagina utilizzando una workstation Sony (NWS-821).

2.17 Kobayakawa (1993)

Per determinare il rigo ([24]), vengono scansionate 32 linee verticali lungo la pagina seguendo i tratti neri; qualsiasi tratto la cui lunghezza sia inferiore della lunghezza media

dei tratti neri viene considerato come candidato per le linee del pentagramma. Per ognuno di questi candidati, l'immagine viene scansionata orizzontalmente e se viene trovata una linea orizzontale che copre il 70% della larghezza della partitura. Queste linee vengono rimosse se c'è un pixel bianco ad una certa distanza, sia sopra che sotto il centro della linea.

Per trovare le teste di nota piena, l'immagine viene scansionata orizzontalmente seguendo i tratti neri lungo le posizioni del rigo e del punto centrale tra le linee. Vengono trovati due massimi nell'istogramma delle lunghezze di queste strisce: il massimo con meno pixel ("circa 2 pixel") viene considerato derivare dai segmenti di linee verticali (gambe e stanghette) mentre il secondo picco ("circa 15-18 punti") viene considerato formato da teste di nota nere. Vengono contati il numero di pixel dell'area a forma di rombo (o di diamante) intorno al centro dei tratti più lunghi, se il conteggio è maggiore del 95% della regione allora è stata trovata una testa di nota piena.

I diesis e i bequadri sono distinti dalle teste di nota, stabilendo se vi è poca distanza tra due segmenti verticali. Le stanghette vengono separate dalle altre linee verticali per la loro altezza poiché è la stessa di quella del pentagramma o più lunga, se sono vicine vengono considerate fine battuta doppie e, in questo caso, vengono cercati due piccoli punti indicanti il segno di ripetizione. Le linee verticali restanti vengono considerate gambe se sono vicine a teste di note o se ci sono teste di note tra i due estremi della linea.

Dopo la rimozione delle gambe, si esegue la scansione del lato opposto alla testa della nota in direzione verticale per ricercare uncini e travi, se viene trovato un qualsiasi pixel nero, viene assemblata una componente connesssa. Se la larghezza della componente è minore del doppio della larghezza della testa di nota e l'inclinazione (presumibilmente l'angolo della linea che connette i punti medi dei bordi sinistro e destro del componente) è eccessiva, allora è considerata un uncino.

La fase del riconoscimento avviene utilizzando un template-matching, con modelli acquisiti da varie partiture di esempio, editate con un editor bitmap, quindi codificate in base alla lunghezza dei tratti.

Il sistema è stato implementato su Sun Sparc 2 e workstation Omron Luna, connessi ad uno scanner da 200 d.p.i. e un sintetizzatore Yamaha DX7.

2.18 Roth (1994)

Il sistema proposto da Roth ([18]) è completo e funzionante, anche se non con buone prestazioni. Il suo lavoro risulta interessante per l'applicazione di varie tecniche di rimozione delle linee sia in direzione orizzontale, che verticale. Con un approccio tutto grafico si cerca di risolvere il problema dei simboli spezzati dalla rimozione delle linee e degli oggetti che si toccano.

Il sistema OMR si basa sui seguenti sette passi:

1. Rotazione: L'immagine bitmap può essere ruotata per ottenere una corretta angolazione: questo processo non è automatico e una corretta disposizione si ottiene in modo empirico manualmente.

2. Analisi statistica su percorsi verticali: La lunghezza media dei percorsi verticali di pixel neri e bianchi è usata per trovare lo spessore delle linee del pentagramma e la distanza tra due linee.

3. Localizzazione e rimozione pentagramma: Il pentagramma viene identificato cercando gruppi di 5 picchi nella proiezione Y. Successivamente viene rimosso con l'accortezza di eliminare le linee di spessore non superiore a quello calcolato al passo precedente.

4. Localizzazione e rimozione linee verticali: L'identificazione di tutte le linee verticali (barre di battute, gambi di note, le parti verticali delle alterazioni, ecc.) è stata realizzata prima con le proiezioni X e poi (con risultati migliori) con l'utilizzo di operazioni morfologiche.

5. Etichettatura oggetti: Gli oggetti rimasti vengono identificati basandosi su caratteristiche grafiche (dimensioni, numero di pixel, baricentro).

6. Riconoscimento simboli: I simboli vengono riconosciuti valutando le caratteristiche osservate attraverso una serie di algoritmi che tengono conto di un insieme di regole di carattere grafico.

7. Salvataggio in formato proprietario: La notazione riconosciuta viene salvata in un file in un formato che fa riferimento all'editor musicale Lipsia (sviluppato in ambito universitario).

I risultati ottenuti sono modesti, anche perché attualmente è previsto il riconoscimento di un insieme limitato di simboli. Tra i progetti di miglioramento del sistema è di notevole interesse l'idea di introdurre una *retroazione* controllata dall'interpretazione semantica. Infatti, i sistemi che utilizzano la semantica, di solito lo fanno solo per una verifica e una correzione dei risultati degli stadi precedenti; appare invece interessante, una volta individuate le zone in cui non è stato possibile fare un corretto riconoscimento, ripetere l'analisi grafica con metodi più precisi o variando i livelli di tolleranza che controllano il processo.

2.19 McGee e Merkley: MusicReader (1994)

MusicReader è un sistema di riconoscimento interattivo realizzato da William F. McGee e Paul Merkley ([35]) presso l'Università di Ottawa nell'Ontario, Canada, su una macchina con processore 386 a 16MHz e sistema operativo MS-DOS.

Il sistema può tradurre l'informazione nel formato Darms, per la stampa, o MIDI, per la riproduzione e ha un approccio di tipo classico: rimuove il rigo, identifica le componenti, classifica le entità musicali e produce codice Darms in output. Inoltre, viene fatto uso di routine interattive nel processo di scansione, nella classificazione e nell'editing del file Darms.

Per ridurre le dimensioni dei file derivanti dalla scansione si utilizza una codifica che tiene traccia solo dei tratti di pixel neri. Per poter avviare l'identificazione del rigo, l'immagine deve essere ruotata e ciò è realizzato o girando la partitura o via software.

Inizialmente si esegue una scansione a 50 d.p.i. per ottenere un'immagine di bassa qualità su cui l'operatore può selezionare, con l'uso del mouse, il vertice superiore sinistro e quello inferiore destro di un rettangolo che racchiuda i pentagrammi che si desidera analizzare subito. Quando tutti i rettangoli sono stati indentificati viene fatta una scansione con risoluzione finale di 300 d.p.i. e il risultato codificato, in modo da considerare solo i tratti neri, e ruotato.

Il classificatore produce un file di testo con i seguenti elementi:

- numero del pentagramma
- lato sinistro della componente (in pixel)
- lato destro della componente (in pixel)
- posizione verticale della componente (in rappresentazione Darms)
- identificatore della componente (il più vicino possibile al codice Darms)

Il classificatore, parte principale del sistema, implementa molte funzioni, ma le principali sono:

1. Indentificazione e rimozione del rigo
2. Identificazione e classificazione delle componenti
3. Classificazione delle travi e degli accordi

Nella prima fase, il classificatore cancella lungo il pentagramma, rimuovendo il rigo; quando incontra e identifica delle componenti connesse, esegue la classificazione, scrive sul file di output e le cancella dalla memoria. L'algoritmo di rimozione correla la linea corrente

con la successiva, nella scansione da sinistra verso destra, e se vicino al rigo corrente c'è un elemento lungo e sottile, esso viene associato alla linea e viene sistemata la posizione del rigo. Come conseguenza della scelta di utilizzare un classificatore ad un passo, dovuta alle scarse risorse di memoria, si ha la perdita di parti comuni al rigo e ad altri oggetti. Le uniche assunzioni fatte sono che il rigo è formato da cinque linee parallele. Quando gli elementi del rigo terminano (di solito perché sono state incontrate le stanghette) vengono rimossi dalla memoria.

Nonostante l'algoritmo di rimozione del rigo non sia raffinato, è importante il punto di partenza. Inizialmente, il programma non conosce né il numero né la posizione del pentagramma, tuttavia, una volta stabilito che il numero di piccoli tratti è multiplo di cinque, il programma si arresta, emettendo un beep, e visualizza quello che ritiene essere un rigo. L'operatore può confermare e salvare l'informazione o rifiutare e premere un tasto per inserire il numero di pentagrammi (un intero tra 1 e 9). In questo secondo caso, al programma vengono fornite la posizione della prima e dell'ultima delle cinque linee di ciascun rigo. Occasionalmente le chiavi iniziali potrebbero essere omesse. L'effetto è in parte dovuto alla difficoltà incontrata nell'identificazione accurata delle chiavi di Do, che hanno un gran numero di differenti rappresentazioni.

L'analisi delle componenti connesse viene effettuata contemporaneamente alla rimozione del rigo. Se una componente connessa supera una certa soglia sulla dimensione e posizione, essa viene rilevata e successivamente se ne tenta una classificazione Darms chiedendo all'operatore se salvare, cambiare, analizzare (per le travi e/o accordi), o cancellare la componente; nel caso si decida di cambiare l'operatore potrà digitare la rappresentazione Darms. Nell'analisi dei gruppi e degli accordi, la componente viene ruotata in memoria, e le gambe vengono identificate come linee di pentagramma; ad ogni gamba si associa un codice Darms in base ad un'analisi delle componenti connesse eseguita dopo la rimozione delle gambe e si chiede all'operatore di salvarlo, cambiarlo o scartarlo. Le restanti componenti connesse sono classificate come travi o note e quindi viene determinata la rappresentazione Darms.

L'operazione di classificazione delle componenti utilizza le seguenti proprietà: le dimensioni del rettangolo che contiene la componente, la posizione relativa del pixel alto e basso (utilizzati per distinguere diesis, bemolli e bequadri), la distanza dal punto medio verticale e in orizzontale, la densità della componente confrontata con quella del rettangolo che la contiene, il numero di vuoti. Le proprietà sono normalizzate allo spazio del pentagramma.

L'uscita del classificatore non è in ordine, pertanto è necessario un processo di ordinamento per pentagramma e posizione nello stesso. Le osservazioni sul numero di accordi e gruppi vengono tolte in questa fase e si avanzano alcune semplificazioni sintattiche (es. le note che appaiono allo stesso istante vengono - opzionalmente - fuse a formare un accordo).

Le prestazioni in MusicReader sono molto buone con sorgenti monofoniche in notazione musicale standard, discrete con semplici partiture per pianoforte e ragionevoli per musica più complessa.

2.20 NoteScan (1994)

È un sistema OMR ([36]) realizzato dalla Grande Software Inc., di Seattle (WA) per Macintosh IIci a 25MHz (convertito per MS-DOS) che converte l'informazione musicale riconosciuta in un formato di file intermedio, NoteScan NTIF, che può essere a sua volta trasformato in una grande quantità di formati proprietari utilizzati dai programmi di notazione commerciali. Come dispositivo di scansione della partitura può essere utilizzato un qualsiasi scanner commerciale che produca file TIFF.

La versione per Macintosh produce delle immagini digitali che vengono visualizzate sullo schermo e possono essere modificate con programmi come *Adobe Photoshop*. La partitura ricostruita può essere visualizzata, ampliata, ridotta e modificata con programmi per applicazioni musicali come *Music Printer Plus* e *Nightngale*.

NoteScan rileva informazioni per il layout della partitura, la posizione e le dimensioni del pentagramma, le dimensioni della pagina, la posizione delle note, il loro raggruppamento, le alterazioni, i cambiamenti di chiave e di tonalità; ignora in modo automatico i titoli, le lettere di ripetizione, segni di articolazione, il testo, le indicazioni dell'esecuzione.

Il software è progettato per la stampa, la riproduzione sonora e l'analisi musicale; nelle prove effettuate dagli autori ha riportato un'accuratezza del 90% circa nel riconoscimento.

2.21 K.C. Ng e Boyle (1994)

Il sistema è stato sviluppato da Kia C. Ng e Roger D. Boyle ([25]) presso l'Università di Leeds, Gran Bretagna, per piattaforma Unix e converte l'informazione musicale riconosciuta in Standard Midi File.

Esso ha in ingresso un'immagine di tipo bitmap e la visualizza sullo schermo, in uscita però non fornisce una rappresentazione dello spartito ricostruito perché non rientra tra gli obiettivi il supporto della stampa musicale, di conseguenza non c'è modo di editare l'informazione. La filosofia del sistema considera l'esatto contrario del processo di scrittura della musica: mentre un compositore normalmente scriverebbe prima la testa della nota e poi la gamba o la trave e per ultimi altri segni come le legature di valore o legature di espressione, questo sistema seleziona prima gli elementi lunghi e fini come le legature, seguite dalle travi e quindi dalle gambe. In questo modo gli elementi compositi complicati vengono scomposti in primitive di un livello grafico inferiore prima del riconoscimento.

Il sistema è strutturato in diverse fasi nelle quali, prima, si ricercano gli elementi essenziali per l'interpretazione dello spartito, poi si verifica la loro mutua coerenza e quindi si permette una ricerca intelligente per il riconoscimento di elementi più ambigui.

Pre-processing

In questa fase, eliminata l'eventuale inclinazione della partitura con una rotazione dell'immagine, vengono ricercate le linee del pentagramma e successivamente calcolata un costante, data dalla somma dell'altezza media di una linea e della distanza media tra due linee, che sarà il valore base per molti processi successivi. Si opera una segmentazione iniziale utilizzando l'etichettamento dell'immagine e un classificatore riconosce i simboli musicali primitivi non connessi e i simboli che solitamente occupano posizioni particolari rispetto al pentagramma (es. l'armatura di chiave è sempre all'inizio).

Sub-segmentazione e riconoscimento

Nel processo di sub-segmentazione i simboli composti sono divisi in gruppi di primitive musicali di livello più basso come teste di nota, linee verticali, linee orizzontali (e oblique) e curve. Il sistema può rilevare curve (legature di valore e di espressione) che coprono o sono interconnesse con altri simboli. Un classificatore nearest-neighbour riconosce le possibili primitive, alcune con molta sicurezza, mentre altre con delle ambiguità. In questa fase si utilizza la costante, calcolata come detto in precedenza, per paragonare le possibili primitive.

I due processi vengono iterati fino al riconoscimento degli elementi o al soddisfacimento dei criteri di terminazione che dipendono dalle dimensioni e dalla densità dei simboli. Si riassemblano, poi, le primitive nello spartito originale utilizzando la sintassi base della notazione musicale (la posizione relativa degli oggetti ne rivela l'identità).

Miglioramento e conoscenza di alto livello

Le primitive corrette sono state ricostruite come simboli musicali e quindi possono essere analizzate utilizzando come base la conoscenza di alto livello per determinare informazioni più globali come i raggruppamenti, il tempo, la tonalità o per aiutare il riconoscimento di elementi come i raggruppamenti irregolari, i simboli incompleti, le alterazioni che sono difficili da rilevare.

Per determinare la tonalità si utilizza la distribuzione delle note con un sistema di regole che dipendono, ad esempio, dalle alterazioni isolate che sono state trovate, dall'ordine di quelle che costituiscono l'armatura di chiave, oppure che seguono considerazioni come: "le note più suonate in genere sono la tonica e la dominante".

Per riconoscere il tempo viene contato il numero dei sedicesimi presenti nelle battute che sono state completamente riconosciute e se ne deduce il significato in base ad una tabella che fa corrispondere al numero di semicrome il tempo. Nel caso non sia possibile determinare univocamente la divisione si utilizzano altri criteri come, ad esempio, il numero di minime (che fa preferire 2/2 a 4/4), il modo di raggruppare le note, l'incidenza di particolari modelli ritmici e regole euristiche.

Dalle informazioni fornite dalle alterazioni in chiave e dal tempo si ricavano utili condizioni che aiutano a determinare specifici elementi musicali. Quindi si possono ripetere i processi di livello inferiore per correggere ciò che era stato riconosciuto in modo ambiguo.

Questo miglioramento progressivo potrebbe essere utile per le partiture scritte a mano, grazie all'eliminazione dell'ambiguità.

2.22 Lee Sau Dan e Choi (1996)

Sviluppato presso la Hong Kong University nel 1996 ([32]), accetta in ingresso un'immagine in formato TIFF e restituisce un file ASCII che codifica l'informazione riconosciuta in un modo ideato dallo svilupattore del sistema. Sono di facile stesura programmi che convertono il file nei formati più noti ed utilizzati.

Opera su partiture per pianoforte relativamente semplici, su pentagrammi multipli e simultanei se connessi dalle stanghette. La musica polifonica potrebbe non essere gestita in modo appropriato. Vengono ignorate le pause, le alterazioni, le legature, la tonalità e il tempo. Non vengono gestite le immagini inclinate più di 10 gradi.

Le prestazioni riportate segnalano un tempo di riconoscimento per un foglio in formato A4 su una macchina Sun Sparc con SunOs o i486 con Linux di circa 3 minuti.

Il sistema è suddiviso in 3 fasi: pre-processing, bar-unit processing e post processing.

Pre-processing

Nella fase di pre-processing si ha il riconoscimento e la rimozione del pentagramma e delle stanghette.

Riconoscimento e rimozione del rigo – Queste operazioni vengono effettuate utilizzando il LAG, *Linear Adjacency Graph*, una struttura dati che resta invariata rispetto a piccole rotazioni dell'immagine, realizzabile in modo efficiente e che permette ai processi seguenti di operare direttamente su di esso, riducendo la quantità di memoria necessaria. Prima di optare per il LAG, era stata vagliata l'ipotesi di usare la proiezione lungo l'asse verticale che richiedeva però assenza di inclinazione per l'immagine. Essendo questa condizione impossibile da realizzare senza qualche intervento di correzione è stata provata la

trasformata di Hough, che però si è rivelata inadatta in quanto riportava molte false linee del pentagramma.

Dal pentagramma si traggono informazioni come lo spessore delle linee e le dimensioni dello spazio fra due linee, importanti per conoscere la qualità della stampa della partitura originale, la risoluzione del processo di scansione e per misurare le dimensioni e le distanze dei simboli in unità normalizzate a questi due valori, in modo da evitare l'inflessibilità di misure assolute e valori soglia statici. Si ottiene anche l'inclinazione del pentagramma che permette al sistema di migliorare l'accuratezza del riconoscimento, nel caso di un'inclinazione troppo elevata.

Riconoscimento e rimozione delle stanghette – Il riconoscimento e la rimozione delle stanghette permettono di dividere lo spartito in unità più piccole (bar-unit) cosicché il processo possa operare su unità più piccole e quindi in modo più veloce e con un bisogno inferiore di memoria. Il riconoscimento avviene proiettando lo spartito, corretto dall'inclinazione, sull'asse orizzontale.

Bar-unit processing – In questa fase l'immagine viene processata bar-unit per bar-unit.

Riconoscimento dei simboli di una nota – Il riconoscimento dei simboli di una nota viene realizzato grazie al LAG, in cui i simboli sono divisi in componenti primitivi. Utilizzando la proiezione sull'asse orizzontale si cerca di individuare le componenti primitive, ovvero la gamba, la testa, gli uncini o le travi. I punti e le alterazioni sono riconosciuti utilizzando delle *bounding box*.

Riconoscimento degli attributi – Nel riconoscimento degli attributi dovrebbero essere riconosciute le chiavi, le alterazioni e il tempo, cosa che il sistema non fa.

Post-processing

Nella fase di post-processing si hanno il riconoscimento dei simboli globali, come i segni di dinamica, del pedale e le legature (non realizzato) e l'unificazione di tutti i risultati in cui si ricostruisce la partitura. In questa fase viene suggerita l'aggiunta di un controllo semantico, che però non è stato realizzato in questo sistema.

2.23 Vladimir T. Bushel (1996)

Il sistema di riconoscimento ottico realizzato da Bushel ([15]) su PC IBM con Windows è orientato alla riproduzione sonora della partitura e produce in output un file Midi. Il sistema è suddiviso in cinque fasi:

- la "rectification" dell'immagine, ovvero l'eliminazione dell'inclinazione per mezzo della seguente trasformazione:

$$\begin{cases} X = x\cos\theta - y\sin\theta \\ Y = x\sin\theta + y\cos\theta \end{cases}$$

  Con (X, Y) = coordinate originali e (x, y) =coordinate dopo la correzione

- il riconoscimento e la rimozione del rigo, prestando particolare attenzione a non rimuovere parti comuni con altri oggetti

- etichettamento delle componenti e costruzione del file in cui ogni oggetto viene rappresentato tramite le sue caratteristiche, come le dimensioni del rettangolo circoscritto, la posizione del centro di massa, l'area, il rapporto altezza/larghezza, ecc.

- classificazione, che utilizza una grammatica musicale per il riconoscimento delle componenti che procede da sinistra verso destra e dall'alto verso il basso. In questo approccio viene definito un alfabeto con i simboli come le teste di nota, le gambe, le travi, ecc., le frasi come collezione di simboli e un linguaggio come infinito numero di frasi.

- riproduzione del file Midi.

2.24 Adaptative Optical Music Recognition (1996)

Il sistema realizzato da Ichiro Fujinaga ([29]) presso la McGill University di Montreal, Canada, per Sun SPARC2, è di tipo adattativo in modo da permettere l'apprendimento di nuovi simboli musicali e notazioni scritte a mano e il miglioramento continuo dell'accuratezza con cui questi oggetti vengono riconosciuti grazie al perfezionamento dei parametri interni. Un altro aspetto rilevante è costituito dal fatto che differenti copie del sistema possono evolvere lungo linee differenti in quanto ogni sistema sviluppa le proprie esperienze in accordo con i bisogni degli utenti.

La realizzazione di questo sistema adattativo si basa su un apprendimento incrementale basato su esempi, che identifica gli oggetti sconosciuti tramite la loro somiglianza ad uno o più esempi conosciuti e memorizzati.

Il sistema è composto da un database di simboli e tre processi interdipendenti: un riconoscitore che individua, separa e classifica i simboli musicali in categorie musicalmente significative utilizzando un algoritmo *k-nearest neighbour* e il database; un editor di notazione musicale (il *Nutation* di Glen Diener, ma può essere utilizzato un qualsiasi editor musicale disponibile in commercio) per permettere la correzione ad un operatore umano; un learner, ovvero un processo di apprendimento che migliora la velocità e l'accuratezza delle sessioni di riconoscimento successive risistemando continuamente il database e ottimizzando le strategie di classificazione.

Il programma è diviso in 7 sezioni: rimozione del pentagramma, rimozione del testo, segmentazione, estrazione delle proprietà, classificazione, ricostruzione dello spartito e apprendimento.

Rimozione del pentagramma

La rimozione del pentagramma è effettuata in modo che non vengano rimosse grosse quantità di simboli musicali. Si utilizza la rappresentazione *vertical run-lengths* dell'immagine per determinare l'altezza delle linee e degli spazi del pentagramma. Vengono eliminate tutte le linee verticali nere con spessore maggiore del doppio della linea del pentagramma e minore dell'altezza dello spazio. Si cerca di eliminare l'inclinazione dello spartito spostando verso l'alto o verso il basso di una certa quantità una parte dell'immagine. La rimozione di alcuni oggetti, come le legature e dinamiche, la cui altezza è simile a quella della linea di pentagramma, avviene confrontando le bounding box minimali che li contengono con quella contenente la linea. Con la proiezione sull'asse verticale si selezionano le candidate a formare il pentagramma, mentre con quella sull'asse orizzontale si stabilisce se c'è più di un pentagramma e i margini destro e sinistro della pubblicazione.

Rimozione del testo

L'intenzione è utilizzare separatamente un programma OCR. Con delle efficaci euristiche si procede alla individuazione dei testi, che possono apparire quasi ovunque nella pagina. Non vengono rimosse lettere che non sono all'interno di parole, come ad esempio i simboli di dinamica. Fallisce in due circostanze: se le lettere sono connesse ad altre o se toccano le linee del pentagramma perché potrebbero far interpretare una nota che tocca lo spartito come una lettera.

Segmentazione e estrazione delle proprietà

Durante la fase di segmentazione vengono ricercati ed isolati gli oggetti da classificare partizionando l'immagine digitale in regioni disgiunte. Dato un simbolo se ne estrae l'insieme

delle proprietà misurabili, come la larghezza e l'altezza, che forniscono una descrizione numerica della forma. Per mezzo di algoritmi genetici si ricerca l'insieme di pesi da utilizzare per scegliere le proprietà. Con queste misurazioni viene prodotto il *feature vector*, ovvero l'insieme di misurazioni per ogni singola componente connessa.

Classificazione

La classificazione utilizza la tecnica k-NN per decidere, in base al suo feature vector, a quale classe appartenga un oggetto sconosciuto. Viene utilizzato k-NN per la semplicità e perché non richiede una conoscenza a priori della distribuzione sottostante dei simboli nello spazio delle proprietà, permettendo al sistema di apprendere nuove classi di simboli. Inoltre una classe di simboli può occupare due o più regioni disgiunte come, ad esempio, le travi e le legature che variano molto in forma e dimensione, o come le pause da un quarto e le chiavi di violino che hanno forme completamente diverse a seconda dell'editore musicale.

Ricostruzione dello spartito e apprendimento

Durante la riscostruzione dello spartito si tenta una realizzazione elementare della partitura per verificare l'accuratezza del classificatore. Durante la fase di apprendimento, si cerca di migliorare l'accuratezza e l'efficienza del riconoscimento perché:

- dopo una fase di training, il riconoscimento può essere effettuato senza l'intervento umano, attraverso un'esecuzione background e, se necessario, su più computer;
- la velocità di esecuzione è direttamente legata al numero di proprietà usate e al numero di simboli memorizzati nel database;
- i sistemi OMR, incluso AOMR, non ottengono un'accuratezza del 100% e il risultato deve essere controllato da operatori umani, con un processo di correzione che, in relazione alla complessità musicale, può durare pochi minuti come un'ora. Quindi il tempo di esecuzione di un sistema OMR deve essere comparato a quello dell'operatore umano.

2.25 Cantor (1996)

Questo sistema, sviluppato da David Bainbridge e Timm Bell ([5], [6]) presso l'Università di Waikato e Canterbury in Nuova Zelanda, è stato progettato per essere estensibile, senza la necessità di cambiare il codice sorgente. L'estensibilità è ottenuta fornendo strumenti per il riconoscimento musicale che vengono usati per confezionare il sistema adatto al tipo di musica che si deve riconoscere. Gli strumenti includono una selezione di metodi per

identificare i pentagrammi e isolare gli oggetti, metodi per descrivere e identificare forme musicali primitive e una grammatica per specificare le relazioni tra le forme che sono state riconosciute. Il sistema è abbastanza flessibile da lavorare con insiemi di simboli di differenti editori e anche con differenti tipi di notazione musicale.

Il sistema è diviso in quattro fasi: identificazione del pentagramma, individuazione degli oggetti musicali, identificazione dei simboli e determinazione della semantica della notazione musicale.

L'identificazione del pentagramma viene realizzata con una proiezione orizzontale preceduta da un rilevamento dell'eventuale angolo di rotazione del rigo: si effettua una proiezione orizzontale per piccole sezioni sul lato sinistro e sul lato destro della pagina; si uniscono le corrispondenti linee individuate e si determina l'angolo di rotazione; si corregge l'immagine e si procede con la proiezione orizzontale. Viene, quindi, applicato un metodo OCR per escludere il testo presente. La rimozione del pentagramma avviene eliminando i pixel se sopra o sotto tale linea, nelle vicinanze del pixel considerato, non ce sono di neri. Si conserva l'immagine originale per permettere un controllo successivo nel caso di un'errata cancellazione di un pixel.

In questa fase si ricercano i pixel neri e quelli a loro connessi utilizzando un algoritmo flood-fill. Un metodo per affrontare gli oggetti frammentari può essere l'unione, dopo un confronto, di bounding box adiacenti che sono in prossimità del rigo. Un altro approccio potrebbe mirare alla ricostruzione dei simboli in una fase successiva definendo un simbolo come costituito da parti. Oppure si può basare il riconoscimento di un oggetto sulla posizione in cui questo deve essere, cercando nell'immagine originale se tale oggetto è propenso alla frammentazione

Il riconoscimento degli oggetti primitivi (teste, gambe, uncini, ecc.) viene realizzato rendendo disponibili molteplici approcci attraverso un linguaggio chiamato **Primela**, progettato per questo scopo, con il quale una qualsiasi combinazione di queste tecniche può essere utilizzata per identificare un oggetto primitivo. Vi è anche un editor grafico per disegnare forme particolari che possono essere usati come modelli, come profili per le proiezioni o curve per la trasformata di Hough. Gli oggetti riconosciuti con elevata sicurezza vengono rimossi dall'immagine per semplificare le ricerche successive, quelli incerti sono classificati con un peso, compreso tra 0 e 1, e lasciati nell'immagine.

Gli oggetti primitivi vengono assemblati in elementi musicali utilizzando un approccio di tipo grammaticale, *Definite Clause Grammars* (DCG). Le condizioni usate nel DCG, che misura la "vicinanza" degli oggetti, sono normalizzate rispetto alle dimensioni del pentagramma considerato, in modo da poter trattare le partiture che hanno pentagrammi di dimensioni differenti per parti differenti. Poiché il parser si basa su back-tracking, la quantità di tempo richiesta può aumentare notevolmente; è, quindi, necessario un uso prudente dell'operatore di taglio per ottenere complessità temporali trattabili.

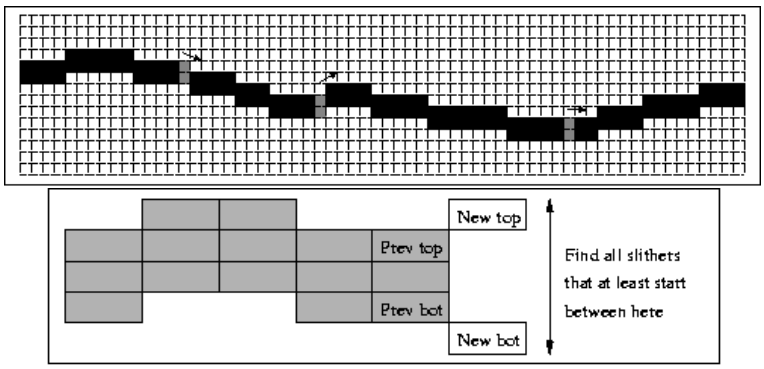


Figura 2.2: Il processo *wobble*

Durante la fase di parsing viene costruito un albero degli elementi musicali assemblati cui può essere applicato un controllo semantico.

Il sistema Cantor è stato disegnato per essere adattabile a forme alternative della notazione musicale, non solo la notazione musicale tradizionale.

La fase di rimozione del pentagramma rileva automaticamente il numero di linee in input cercando gruppi di linee con spaziatura regolare; inoltre non assume che la spaziatura o il numero di linee sia il medesimo per ogni gruppo. Il linguaggio Primela permette all'autente di definire nuove forme per gli oggetti, mentre il DCG può essere modificato per tenere in conto costrutti diversi.

Processando una partitura che utilizzava una notazione con teste quadre, Cantor ha identificato correttamente 5 chiavi e tutte le 121 note nell'opera. Primela ha mancato il riconoscimento di un punto e ha classificato una macchia come un punto; di conseguenza la durata per queste note era scorretta. Tutte le altezze erano corrette. La qualità della stampa era inferiore a quella dei campioni in notazione musicale tradizionale utilizzati per i test, così la tolleranza nella descrizione di Primela è stata più elastica. L'esame di dati incerti nel sistema ha eliminato ogni ambiguità in tutti i casi in cui più di una descrizione in Primela corrispondeva ad una particolare parte musicale. In (rif. bibliografico) l'autore propone una fase di post-processing per l'identificazione del pentagramma che ne migliorerebbe notevolmente l'accuratezza. Il passo, chiamato *wobble*, lavora da sinistra verso destra tentando di seguire il rigo considerando tratti larghi un pixel. Si cerca il tratto successivo del rigo individuato in base a quello precedente, infatti esso deve essere in una posizione verticale vicina. Il tratto trovato può essere poco più alto o poco più basso del precedente. In questo modo si possono seguire anche righi ondulati.

2.26 Newell e Homeda: MidiScan (1998)

MidiScan ([48]) è stato il primo software per il riconoscimento ottico di partiture di tipo commerciale. È stato sviluppato da Christopher Newell (ZH Computer, Minneapolis) e da Wladyslav Homeda (CPZH, Varsavia, Polonia) e commercializzato dalla Musitek (Ojai, California).

Il programma accetta in ingresso un'immagine in formato TIFF. Nella prima fase del riconoscimento, avviene la ricerca dei pentagrammi e viene messo in evidenza l'inizio e la fine di ognuno per mezzo di alcuni delimitatori grafici. Se il riconoscimento automatico non è corretto, puè essere effettuato manualmente col mouse.

Avvenuta la localizzazione dei pentagrammi, il riconoscimento è completamente *bath processing* e non permette alcuna interazione. Al termine, viene ceata una descrizione simbolica (MNOD, Music Notation Object Description), dell'immagine riconosciuta, che successivamente viene utilizzata per la ricostruzione.

Eventuali errori di riconoscimento possono essere corretti mediante un editor MNOD, che consente di aggiungere, cancellare o cambiare i simboli. L'editor divide lo schermo orizzontalmente in due finestre e consente la visualizzazione dell'immagine bitmap di partenza e quella MNOD ricostruita. In questo modo è possibile comparare le due versioni e quindi identificare gli errori.

Come uscita, il programma genera un file Standard Midi direttamente dal documento MNOD, che puo' essere archiviato. Poiché lo scopo del programma è quello di generare un file Midi, esso ignora completamente gli accenti, le legature, il testo, il tempo, la diteggiatura, la dinamica, ecc.

Il riconoscimento

La fase di riconoscimento sembra eccellente se la scansione dello spartito è di alta qualità, in caso contrario, il fattore di riconoscimento diminuisce rapidamente, è sufficiente una risoluzione di 200 d.p.i. perché il programma perda di efficienza.

Sono state fatte alcune assunzioni:

- I pentagrammi non devono essere inclinati. Sembra che MidiScan abbia delle routine per correggere l'inclinazione, ma molto spesso assume che le linee siano diritte. Se ciò non accadesse, le note riconosciute mostrerebbero un errore sistematico dove la linea piega più della metà della distanza tra i pentagrammi.

- Ogni pentagramma deve iniziare con una chiave. MidiScan spesso trova uno dei simboli di chiave, ma non controlla se questo è davvero o no una chiave.

Ovviamente MidiScan ricerca le teste delle note e le gambe, e successivamente cerca

Figura 2.3: Esempi di primitive grafiche

di contare il numero di travi o di uncini. Qualche volta può ottenere un risultato sbagliato (uncini al posto di travi), ma dal punto di vista dell'esecuzione MIDI, ciò non ha peso.

Il programma ha seri problemi nel riconoscimento del tempo della battuta (4/4, 3/4) e gli identificatori di terzine, quartine, ecc., il più delle volte, devono essere inserite manualmente.

2.27 Fahmy e Blostein (1998)

Nell'analisi delle immagini, il riconoscimento delle primitive ricopre un ruolo molto importante per la comprensione di diagrammi (come quelli circuitali, partiture, ...), presentati come immagini; con il termine primitive si indicano gli elementi grafici elementari, la Figura 2.3 ne è un esempio. Famhy e Blostein [46] ipotizzano che il processo di riconoscimento delle primitive produca un insieme di possibili interpretazioni per ciascuna di esse. Per ridurre le ambiguità, prodotte dal sistema di riconoscimento delle primitive, vengono usate delle informazioni contestuali provenienti dall'immagine e, applicati dei vincoli ricavati dal dominio derivante dall'immagine stessa. Questo processo viene chiamato *soddisfazione dei vincoli*, *etichettatura* o *rilassamento discreto*. Con il termine "rilassamento" si indica quel processo attraverso il quale gli oggetti, nel caso di applicazioni sull'analisi di immagini, interagiscono fra loro per ridurre l'incertezza. In particolare con il termine rilassamento discreto si indica che ogni oggetto ha associato un proprio insieme di etichette (label set) equiprobabili. Tale processo procede eliminando dal label set gli elementi che non soddisfano i vincoli. Esiste anche un rilassamento stocastico nel quale ad ogni oggetto è associato un vettore il quale indica, per ogni etichetta, la verosimiglianza che questa ha con l'oggetto stesso. Il processo procede modificando il valore della verosimiglianza in accordo con il contesto locale in cui si trova la primitiva. Sia il processo di rilassamento discreto che quello stocastico definiscono dei modelli che specificano quali elementi cooperano, e come questi interagiscono, per aggiornare il label set. I metodi esistenti per il rilassamento discreto sono limitati dal fatto che assumono una "conoscenza" a **priori** del modello di prossimità: prima dell'inizio della fase di rilassamento, il sistema può determinare come l'insieme delle primitive sono relazionate dai vincoli. Questo fa sì che tali metodi non possano essere applicati a quei domini nei quali è necessario un'analisi complessa per determinare come le primitive sono relazionate dai vincoli. Per esempio,

nella notazione musicale, si deve riconoscere quali note appartengono ad una battuta, prima che sia possibile applicare il vincolo che la durata della battuta sia consistente con il tempo in chiave. Il modello di prossimità specifica quali oggetti comunicano direttamente tra loro. Il modo con cui tali oggetti comunicano (o si vincolano a vicenda) è dato dal modello di interazione. Questo modello è composto da due parti: una per rappresentare la conoscenza delle relazione fra le etichette, ed una per indicare i meccanismi con i quali applicare tale conoscenza. Quindi possiamo concludere che il rilassamento discreto è un metodo appropiato per ridurre l'ambiguità nelle immagini in cui il grado di località dei vincoli è alto.

Il lavoro svolto da Famhy e Blostein, e applicato a documenti musicali, utilizza il metodo del *graph-rewriting* per la risoluzione del problema del rilassamento discreto e per la creazione del modello di prossimità quando c'è incertezza associata con l'identità delle primitive, che sono ad esempio: teste di note, gambi, beam, etc. (si veda Figura 2.4). Il

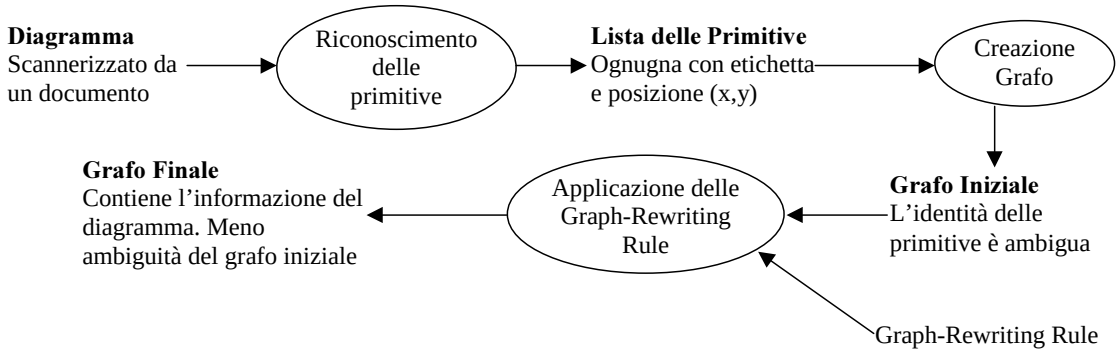


Figura 2.4: Diagramma per l'acquisizione dell'informazione contenuta in un documento cartaceo, in presenza di incertezza, usando il metodo del graph-rewriting

graph-rewriting permette di esaminare e stabilire le associazioni fra le primitive, applicare vincoli, e integrare questa riduzione di incertezza con la costruzione di una interpretazione ad alto livello. Si nota inoltre l'utilizzo del modello gerarchico per il rilassamento discreto. Questo è dovuto al fatto che si analizzano documenti, come gli spartiti musicali, in cui un solo livello di astrazione non è sufficiente per ridurre l'incertezza, ma ne occorrono svariati. Per progettare un sistema di rilassamento gerarchico avente K livelli, bisogna definire un *modello di costruzione*, oltre al modello di prossimità e al modello di interazione, ad ogni livello. Il *modello di costruzione* costruisce gli oggetti e le etichette al livello $m+1$ basandosi sugli oggetti e le etichette a livello m(si veda Figura 2.5).

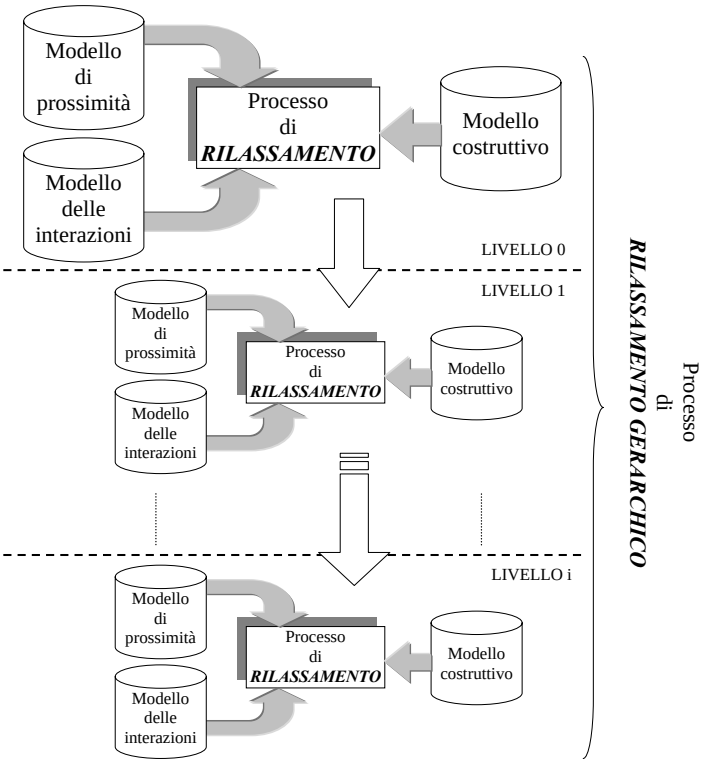


Figura 2.5: Schema relativo alla costituzione del modello gerarchico del rilassamento. Il modello costruttivo di ogni livello viene creato solo se i livelli sono maggiori di uno.

Graph Rewriting

Il *Graph-rewriting* permette di specificare delle trasformazione da effettuarsi su dei grafi. Tali trasformazioni sono rappresentate anch'esse sotto forma di grafo in cui i **nodi** rappresentano le **primitive** e gli **archi** rappresentano le **relazioni** che intercorrono fra le primitive. Vediamo adesso come vengono definite e applicate le regole di riscrittura di grafi (graph-rewriting rule), il cui scopo principale è quello di sostituire un sottografo con un'altro.

Con riferimento alle figure 2.7 e 2.6, una regola di riscrittura di grafi consiste delle seguenti parti:

- $g_l \rightarrow g_r$

  g_l e g_r sono due grafi. Per applicare una regola ad un grafo ospite (host) g si deve determinare il sottografo g_l^{host} di g che è isomorfo a g_l e sostituirlo con un nuovo sottografo g_r^{host} isomorfo a g_r. In altri termini, si richiede che g_l^{host} sia un sottografo indotto del grafo ospite g cioè che se un arco di g termina in un nodo del sottografo g_l^{host} allora questo appartiene a g_l^{host}.

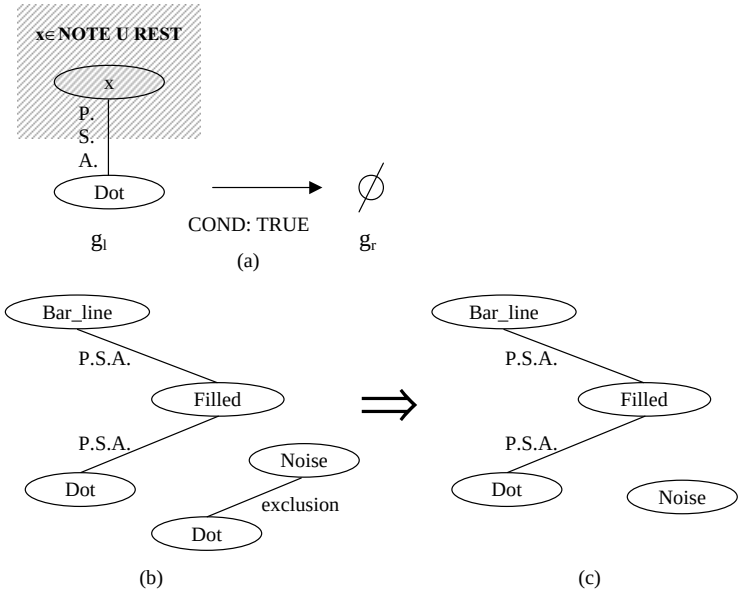


Figura 2.6: Esempio di regola di riscrittura di grafi. (a) regola (b) grafo iniziale (c) grafo risultante. Questa regola permette di eliminare i nodi etichettati con "Dot" a meno che questi non siano collegati, tramite un arco etichettato con P.S.A., ad un nodo etichettato con un elemento appartenente alla classi NOTE o REST. L'etichetta P.S.A. sta ad indicare che esiste un'*associazioni potenzialmente significative* tra i nodi

- i nodi e gli archi di g_l e g_r possono essere etichettati. In questo caso le etichette di g_l devono corrispondere a quelle di g_l^{host} per considerare g_l isomorfo a g_l^{host}.

- EMB: regola di incastro.

  La regola di incastro specifica come il sottografo g_r^{host} viene attaccato al grafo ospite g dopo che è stato rimosso il sottografo g_l^{host}. Per esempio se il sottografo g_l contiene un nodo denotato con k e g_r contiene un nodo denotato con k', allora la coppia (k,k') è usata nell'incastro.

- ATTR: regola di calcolo degli attributi.

  Queste regole specificano come calcolare il valore degli attributi del sottografo g_r^{host} a partire dal valore degli attributi del sottografo g_l^{host}.

- COND: condizione sull'applicazione.

  Le condizioni sull'applicazione specificano le restrizioni che g_l^{host} deve avere affinché sia possibile applicare la regola.

- CPC: Condizione Proibitive Contestuali

  Queste condizioni, denotate dal rettangolo ombreggiato di grigio nei grafi delle figure 2.6 e 2.7, sono specificate da un sottografo PC (Proibitive Contestuali) che è

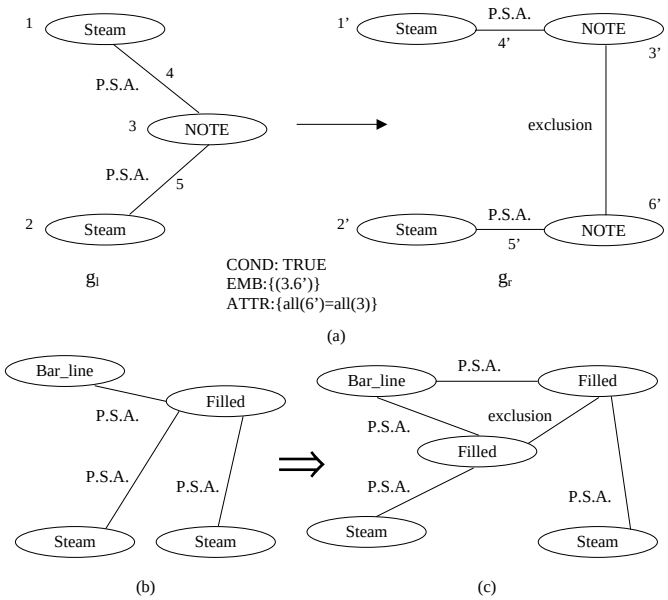


Figura 2.7: Esempio di regola di riscrittura di grafi. (a) regola (b) grafo iniziale (c) grafo risultante. I numeri da 1 a 5 e da 1' a 6' indicano i nodi e gli archi e sono utilizzati da EMB, COND e ATTR per riferirsi a questi. Si può notare che la regola EMB indica che gli archi sostituiscono la partenza dal nodo 1 a 1', da 2 a 2' e da 3 sia a 3' che a 6'; questo comporta la duplicazione del nodo etichettato con Filled. ATTR indica che gli attributi di 6' sono uguali a quelli di 3, oltre ad avere le uguaglianze tra 1' e 1, 2' e 2, 3' e 3, 4' e 4, 5' e 5. Infine, COND non impone nessun vincolo aggiuntivo.

connesso con g_l. Il sottografo PC rappresenta il contesto che inibisce l'applicazione della regola. In questi casi una regola di riscrittura di grafi può essere applicata solo nel caso in cui il sottografo PC non possa essere determinato nel grafo ospite g, nel punto dove questo è connesso con g_l^{host}.

- Etichette variabili per nodi e archi.

  Ai nodi e agli archi di g_l, g_r e PC possono essere associate delle etichette variabili. Il nodo etichettato con NOTE della Figura 2.7 ne è un esempio; a questo possono, infatti, corrispondere tutte le etichette appartenenti all'insieme NOTE (nota vuota, piena, in beam, etc...).

Per quanto riguarda la procedura di applicazione di una regola di riscrittura di grafi questa può essere divisa in sette passi:

1. Localizzare il sottografo g_l^{host}
2. Per ogni CPC della regola di riscrittura di grafi, si deve testare se g_l^{host} può essere esteso in modo da individuare il grafo PC e se la condizione è soddisfatta. In questo caso non è possibile applicare la regola.

3. Calcolare tutte le COND. Se sono false allora non è possibile applicare la regola.
4. Rimuovere il sottografo g_l^{host} così come ogni arco nel grafo ospite g che termina in g_l^{host}. Il grafico di g rimanente viene indicato con *RestGraph*.
5. Creare il sottografo g_r^{host} isomorfo a g_r.
6. Connettere g_r^{host} al *RestGraph* usando le EMB.
7. Attribuire il valore agli attributi dei nodi e degli archi di g_r^{host} usando le ATTR.

Una volta definita la procedura di applicazione di una regola di riscrittura di grafi bisogna specificare in cosa consiste un sistema basato sul graph-rewriting. Questo consiste in un insieme di regola di riscrittura di grafi le quali sono applicate ad un grafo ospite g in maniera non deterministica.

MUBEENA

Nel progetto MUBEENA, sviluppato da Famhy e Blostein, viene analizzato il problema del rilassamento discreto utilizzando il metodo del graph-rewriting. Le trasformazioni grafiche sono usate per esaminare gli oggetti, ai diversi livelli di astrazione, che in questo caso sono in numero pari a 3. Per ogni livello, escluso l'ultimo, sono stati definiti 4 diversi insiemi di regole grafiche che sono: BUILD, CONSTRAINT, SPLIT e INCORPORATE e che vengono applicate nella sequenza rappresentata nella Figura 2.8. Il fatto che l'ultimo livello di astrazione sia composto solo dalle regole di tipo BUILD, SPLIT e CONSTRAINT è dovuto dal fatto che i vincoli che sono applicati nei primi due livelli sono di tipo binario, relativamente al numero di primitive che coinvolgono, mentre il terzo livello rappresenta un vincolo ad alto livello in cui i vincoli sono di tipo n-ario. In questo progetto sono state definite: per quanto riguarda il primo livello, 7 regole di tipo BUILD, 41 di tipo CONSTRAIN, 15 di tipo SPLIT e 10 di tipo INCORPORATE; nel secondo 5 di tipo BUILD, 51 di tipo CONSTRAIN (oltre a quelle del primo livello che vengono riutilizzate), 11 di tipo SPLIT e 11 di tipo INCORPORATE; nel terzo 3 di tipo BUILD, 2 di tipo SPLIT e 23 di tipo CONSTRAIN (oltre a quelle del primo e del secondo livello nuovamente riutilizzate).

Il primo livello di astrazione viene ottenuto applicando i primi quattro insiemi di regole al grafo iniziale. Tale grafico è ottenuto associando un sottografo per ogni primitiva riconosciuta. I sottografi sono creati facendo corrispondere un nodo ad ogni interpretazione appartenente al label set della primitiva e collegandoli con degli archi di tipo "exclusion". Gli archi così etichettati indicano che le interpretazioni sono mutuamente esclusive. È importante osservare che vengono memorizzate, negli attributi dei nodi, le informazioni derivanti dal riconoscitore di primitive quali, ad esempio, la posizione dell'oggetto relativamente al pentagramma. Un caso particolare è dato dalla situazione in cui l'interpretazione

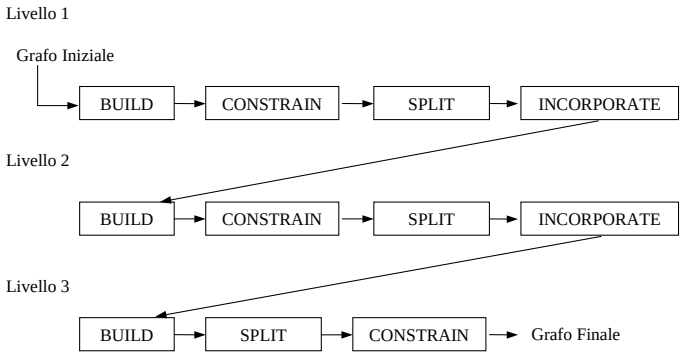


Figura 2.8: Modello gerarchico del progetto MUBEENA.

di una primitiva è certa, in questo caso tale primitiva viene chiamata *anchor* e il nodo associato ha la particolarità di non aver nessun arco incidente etichettato come *exclusion*.

L'obiettivo del processo è : massimizzare il numero di nodi anchor tramite il recupero dell'informazione contenuta nella notazione musicale e, determinare quali interpretazioni sono globalmente consistenti. Il risultato dell'applicazione dei vincoli porta ad un grafo finale i cui nodi rappresentano un'interpretazione soddisfacente i vincoli del dominio, siano questi binari o n-ari.

Analizziamo adesso in dettaglio i 4 gruppi di regole grafiche precedentemente introdotti.

BUILD Questo tipo di regole costruiscono gli archi fra le primitive per rappresentare le *associazioni potenzialmente significative* tra nodi. Questi nuovi archi vengono etichettati con P.S.A. Un esempio di questo tipo di regola è rappresentata nella Figura 2.9. Gli archi etichettati con P.S.A. hanno un attributo di *affinità* che è

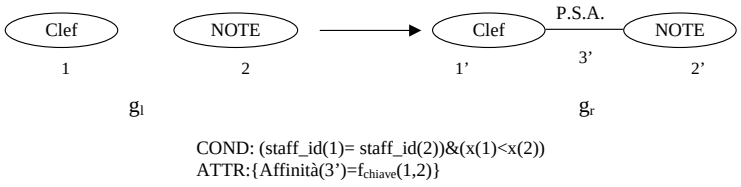


Figura 2.9: Esempio di regola BUILD che forma un arco tra una nota ed una chiave se queste stanno nello stesso pentagramma e se l'ordinata della chiave è minore di quella della nota. Il valore dell'affinità del nuovo arco è calcolato tramite la funzione f_{clef}.

calcolato sulla base di appropiate *funzioni di affinità*. Per calcolare tali funzioni si considera la distanza tra due primitive. L'inverso della distanza orizzontale è usato

per calcolare l'affinità tra le note e la barra di fine battuta, e tra le note e le chiavi. L'inverso della distanza Euclidea è invece utilizzato per calcolare l'affinità fra le note e i gambi.

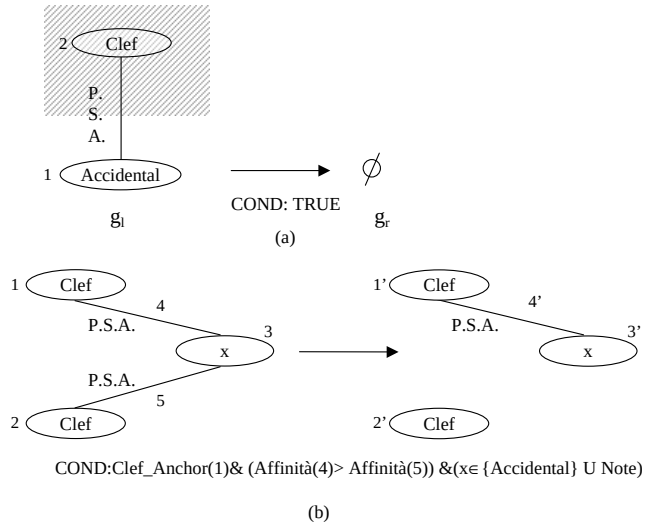


Figura 2.10: Esempio di regole CONSTRAIN. Il significato di ambedue le regole è che non possono esserci note o alterazioni senza una chiave ad esse collegata. In particolare la regola (b) tiene conto dell'arco con affinità minore, che viene eliminato.

CONSTRAIN Questo tipo di regole riducono l'ambiguità cancellando quelle interpretazioni e associazioni che non soddisfano i vincoli. Queste regole possono cancellare nodi, modificare gli attributi dei nodi e cancellare archi, come si può notare dalla Figura 2.10. Alcune delle associazioni non interessanti e conflittuali, create dalle regole di tipo BUILD, sono cancellate da questi tipi di regole. È importante osservare che le regole appartenenti a questa categoria eliminano delle interpretazioni e delle associazioni solo quando si è certi che queste sono false.

Un particolare tipo di queste regole sono le *weed-out* che sono usate per rimuovere le associazioni contraddittorie (vedi Figura 2.10, esempio (b)). Le regole weed-out vengono applicate solo nel caso in cui il nodo affine è un anchor; questo serve ad evitare la cancellazioni di interpretazioni che possono risultare corrette. Questa incertezza non è trattata dalle regole di tipo CONSTRAIN ma è considerata nelle regole di tipo SPLIT.

SPLIT Questo tipo di regole separano i nodi in modo da esplicitare tutte le possibili interpretazioni. Quando un nodo x conta k associazioni contraddittorie, e non c'è informazione sufficiente per giudicare quale sia quella valida, allora si divide il nodo

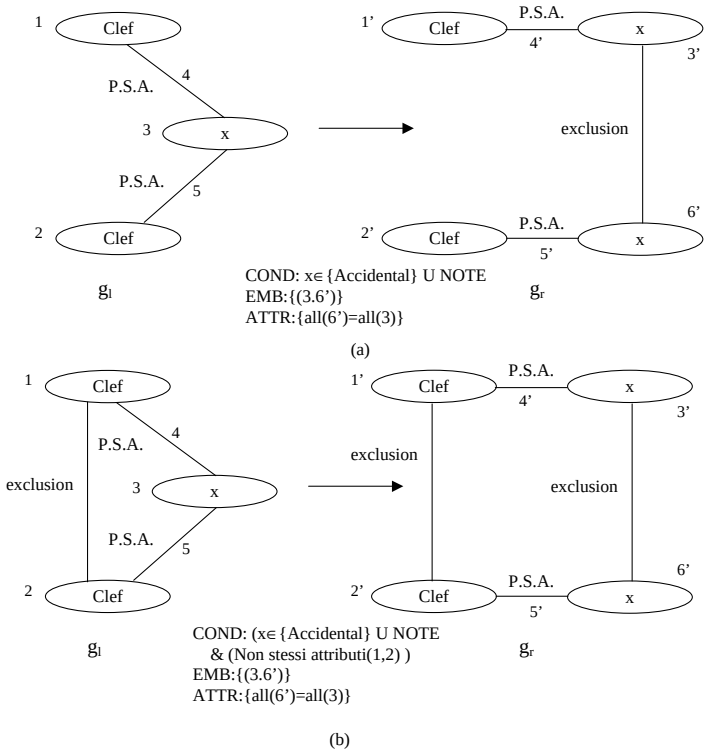


Figura 2.11: Esempio di regola SPLIT. Questi esempi trattano il caso in cui un simbolo ha due chiavi collegate alla sua sinistra e la più vicina non è un anchor. Le regole (a) e (b) differiscono per la presenza, nella seconda, dell'arco exclusion; ciò si ripercuote sulla condizione COND.

x in k nodi così da considerare tutte le possibili associazioni. Vengono riportati degli esempi nella Figura 2.11.

L'applicazione di questo tipo di regole fa si che l'incertezza aumenti. Questo è dovuto al fatto che aumentando il numero di nodi e di archi aumenta anche il numero di interpretazioni. Dall'altra parte l'applicazione delle regole SPLIT è necessaria per poter applicare le regole di tipo INCORPORATE, che incorporano le associazioni semantiche negli attributi e nelle etichette dei nodi.

INCORPORATE Questo tipo di regole costruiscono un'interpretazione ad alto livello dell'immagine e sono necessarie nell'analisi di quei documenti in cui esiste un'interpretazione dell'immagine ad alto livello. Per esempio nel caso della notazione musicale l'associazione di una nota con un punto di valore provoca la variazione della durata della nota stessa. Una volta che questa associazione è stata analizzata, la durata delle note può essere utilizzata per ridurre l'incertezza tramite la verifica che la durata della battuta sia consistente con il tempo in chiave.

Le regole INCORPORATE identificano la semantica dell'associazione tra i nodi e

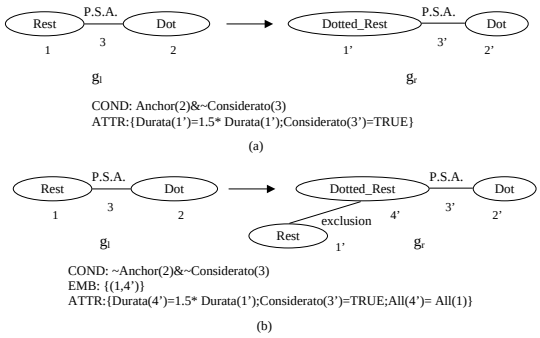


Figura 2.12: Esempi di regola INCORPORATE. Le regole (a) e (b) differiscono per il fatto che il nodo 2 sia o meno un anchor: nel caso (a), è un anchor, la regola modifica solo la durata della pausa; nel caso (b) la regola fa uno split per creare la pausa con e senza il punto.

la memorizzano negli attributi e/o nelle etichette dei nodi stessi. L'applicazione di queste regole permette di ridurre il numero dei nodi nel grafo in quanto il significato di una certa configurazione di un sottografo può essere memorizzata negli attributi di un nodo.

Bisogna fare una distinzione tra i nodi che compongono le regole INCORPORATE che possono essere: *acceptor node*, cioè quelli che assorbono l'informazioni o *donor node*, cioè quelli che forniscono le informazioni. Il *donor node* può essere presente o meno in una regola mentre l'*acceptor node* deve essere sempre presente. Un esempio di tale tipo di regole è dato dalla Figura 2.12.

Il vincolo di ordine alto dell'ultimo livello di astrazione serve per controllare quale tra le configurazioni possibili delle primitive è consistente con il tempo in chiave. Bisogna osservare che i vincoli di tipo n-ario, che compongono quest'ultimo livello, interessano sempre un numero di primitive maggiore di due.

Vediamo adesso in dettaglio come viene creato quest'ultimo vincolo. Per prima cosa si crea un nodo che viene etichettato con B.P.M. (Beats Per Measure) e a cui viene collegato il nodo che rappresenta la barra di fine battuta e tutti quei nodi etichettati come note o pause, tramite degli archi di tipo P.S.A. A questo punto vengono applicate delle regole di tipo SPLIT che permettono di generare tutte le possibili configurazioni del grafo. Infatti alla fine dell'applicazione delle regole di tipo SPLIT si avrà che il nodo B.P.M viene convertito in un insieme n-ario di nodi, etichettati sempre come B.P.M., i quali definiscono le n possibili configurazioni temporali della battuta. Vengono quindi applicate delle regole di tipo CONSTRAINT in modo da eliminare tutte quelle configurazioni che non rispettano il tempo in chiave e tutti quei nodi etichettati come note, pause o barre che non sono connessi con un nodo B.P.M. Il risultato finale di tale processo è un grafo

consistente con il tempo in chiave che rappresenta la configurazione di ogni battuta.

Fahmy e Blostein hanno dimostrato che tale metodo è **convergente** e, converge ad un **unico** grafo finale che non è però ottimale.

2.28 Luth (2002)

Il lavoro condotto da Luth ([61]) ha come obiettivo il riconoscimento dei manoscritti musicali. Esso si basa sull'uso di algoritmi di elaborazione dell'imagine come l'*edge detection*, la scheletonizzazione e il *run-length code*. L'identificazione automatica inizia con la digitalizzazione del manoscritto e procede con le seguenti fasi:

Elaborazione dell'immagine – Lo scopo è l'estrazione di un'immagine binaria minimizzando i disturbi e massimizzando le strutture importanti. L'approccio proposto è quello di attuare un meccanismo a soglia adattativa da applicare in piccole regioni dell'immagine insieme alla stima della soglia ottima per ciascuna regione in questo modo viene raggiunta una buona qualità nella conformità tra i simboli grafici originali e quelli risultanti nell'immagine binaria.

Analisi dell'immagine – L'immagine è decomposta in 5 livelli *layers*, ciascuno dei quali corrisponde a particolari strutture: (i) linee orizzontali (pentagrammi), (ii) linee verticali (barre e gambi delle note), (iii) piccole strutture circolari (teste delle note), (iv) strutture contenenti linee (chiavi, uncini o alterazioni), (v) altre strutture (informazioni testuali e stringhe di caratteri).

Riconoscimento oggetti – Il riconoscimento viene condotto basandosi sulla conoscenza a-priori e su regole di struttura di modelli astratti (linee del pentagramma, gambi delle note, teste delle note). Il significato contestuale viene assegnato alle strutture estratte. Questa operazione genera, inoltre, uno speciale Grafo della Notazione. Tale grafo si basa su un livello sintattico della scrittura manuale per descrivere le relazioni tra le strutture in termini di geometria, topologia e caratteristiche.

Identificazione – Un albero delle caratteristiche colleziona due diverse forme di descrizione strutturali: la struttura relativa alle proprietà dell'oggetto astratto e delle primitive all'interno dell'immagine. Una volta fornite le proprietà dell'oggetto come descrizione strutturale, un algoritmo di *graph matching* stabilisce: quali immagini primitive appartengono allo stesso oggetto, l'identità della strutture e le proprietà reali dell'oggetto.

2.29 Software presente sul mercato (2003)

A completamento della rassegna di sistemi di riconoscimento ottico degli spartiti musicali si riportano i programmi presenti sul mercato. Per ciascuno è riportato il nome e il distributore o l'autore.

- Sharpeye2 - Visiv (http://www.visiv.co.uk)
- Smartscore - Musitek (http://www.musitek.com/smartscre.html)
- Capella Scan - Capella Software (http://www.whc.de)
- Photoscore - Neuratron (http://www.neuratron.com)
- Vivaldi Scan - GoVivaldi (http://www.vivaldistudio.com)

Capitolo 3

La Notazione musicale

In questo capitolo viene riportata una rapida panoramica sulla notazione musicale. Questa panoramica è principalmente rivolta ad evidenziare la struttura della musica dal punto di vista sia del significato che assume per l'esecutore che della rappresentazione grafica e della sintassi della notazione. In questo contesto, solo la musica tradizionale verrà presa in considerazione (p.e. Rossini, Bellini, Donizetti, Verdi, Vivaldi, Scarlatti, Puccini, Bach, Beethoven, ecc.), la musica antica, quella moderna e quella contemporanea saranno considerate a parte. Nella musica moderna e contemporanea (p.e. Berio, Corghi, Donatoni, Grisey, Guarnieri, Huber, Maderna, Manzoni, Nono, Nunes, Pennisi, Sciarrino, Vacchi, ecc.), numerosi altri simboli sono stati aggiunti a quelli propri della musica tradizionale. D'altra parte, la proliferazione dei simboli è principalmente dovuto alla fantasia dei compositori più che ad una reale necessità; infatti, in molti casi, i nuovi simboli sono proposti per esprimere effetti che possono essere descritti anche con la notazione tradizionale. Gli editori tendono a limitare l'adozione di nuovi simboli adottati da almeno un gruppo di compositori, dal momento che il comportamento corrente è quello di creare numerosi nuovi simboli personali per ogni nuova composizione. Per queste ragioni considerare solo i simboli tradizionali non è una limitazione. D'altra parte, dal punto di vista degli editori, delle scuole di musica e dei teatri la notazione musicale è quella che rispettivamente viene pubblicata, insegnata ed eseguita di fronte a grandi platee.

3.1 Le figure e loro attributi

L'elemento base della notazione musicale è la figura. La figura può essere una nota o una pausa. Il suono è associato alle note, mentre il silenzio è associato alle pause. Le figure vengono lette da sinistra a destra. Prima di affrontare l'organizzazione delle figure nello spartito, è necessaria una più dettagliata descrizione degli elementi base correlati alle

figure. Il concetto di nota nella notazione musicale non è direttamente connesso a un determinato suono, poiché il suono di una nota può essere dedotto dalla sua rappresentazione sulla base della:

- chiave musicale;
- posizione della testa della nota sul rigo musicale (per alcuni strumenti il rigo musicale può essere costituito da una a sei linee, nel caso di cinque linee il rigo musicale è anche detto pentagramma);
- presenza di simboli di alterazione (detti anche inflessioni cromatiche) propri della singola nota o appartenenti all'armatura di chiave.

L'altezza di una nota nel rigo musicale rappresenta il suo suono in un certo contesto. L'esempio seguente, Figura 3.1, è fortemente esasperato e così la posizione dei simboli sopra e sotto la nota potrebbe essere assegnata seguendo altre regole. Le regole usate in casi critici si basano sull'esperienza e non sono riportate in nessun libro, neppure in quelli menzionati come i manuali per la stampa della musica (specialmente per la diteggiatura e i simboli strumentali che saranno discussi in seguito). Queste conoscenze vengono tramandate oralmente e rigorosamente mantenute dagli editori.

La durata di una **nota** è convenzionalmente rappresentata da un valore relativo alla durata della pulsazione, ad esempio, una nota da un ottavo, una nota da un quarto, ecc. La durata della pulsazione è stabilita sulla base delle indicazioni di tempo. Il valore effettivo della figura dipende da:

- Il tipo della testa della nota (vuota o piena);
- La presenza o meno del gambo;
- La presenza o meno delle code (nei gruppi di note, queste vengono convertite in barre);
- La presenza di simboli per il cambio di durata come il punto di valore, la corona.

Inoltre, la vera durata di una nota deve essere valutata considerando questi aspetti e il contesto nel quale la nota è inserita poiché il **tempo**, stabilendo il riferimento assoluto del ritmo dell'esecuzione, può essere diverso.

Visivamente la forma e la posizione della testa della nota, nonché la direzione del suo gambo (se è presente), seguono regole ben definite. La posizione degli altri simboli intorno alla nota dipende molto spesso dall'orientamento del gambo. Per la durata delle **pause** vengono applicate regole simili, ma più semplici.

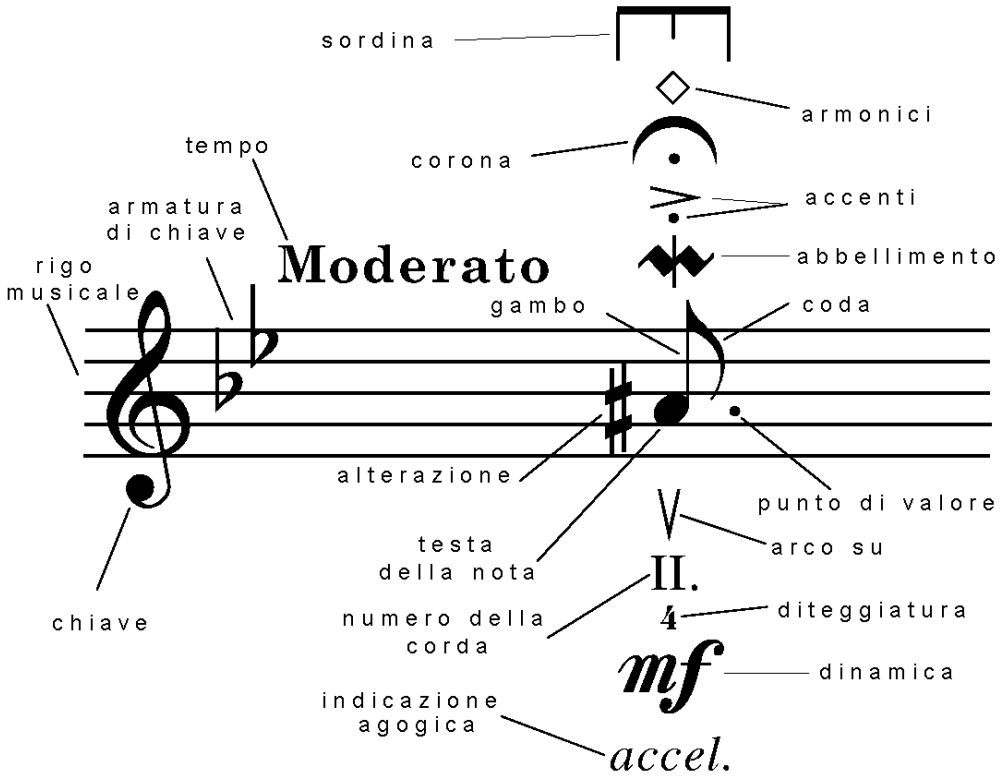


Figura 3.1: La nota, le sue componenti principali e alcuni simboli ad essa correlati.

Il **tempo** di un'esecuzione musicale viene descritto attraverso indicazioni testuali (*grave, largo, larghetto, lento, adagio, andante, andantino, moderato, allegretto, allegro, allegro assai, vivace, presto, prestissimo ecc.*) spesso dette anche indicazioni agogiche. Questi simboli cambiano il contesto di esecuzione della musica poiché, una volta stabiliti, rimangono validi finché non viene fornita una diversa indicazione del **tempo**. Queste indicazioni sono spesso accompagnate da aggettivi o attributi che meglio descrivono l'andamento generale (*solenne, sostenuto, maestoso, marziale, grazioso, scherzando, giocoso, con fuoco, dolce, tranquillo, impetuoso, agitato, appassionato, deciso, con moto*), che a volte vengono tradotti in altre lingue. In generale, queste specificazioni sono soggette all'interpretazione, mentre, quando è richiesto un valore di scansione temporale predefinito, può essere usata un'indicazione metronomica, fissando in tal modo il numero di pulsazioni al minuto.

Altre indicazioni nello spartito, ad esempio *rallentando* (*rall.*), *allargando* (*allarg.*), sono cambiamenti relativi completamente soggettivi nella loro realizzazione. Queste indicazioni sono frequentemente fornite attraverso speciali simboli grafici quali una **freccia** per *allarg.* o un'**onda** per *rall.*

Una nota può avere degli abbellimenti come: *appoggiatura, notine, mordente, trillo, glissando, gruppetto, tremolo.* Questi simboli sono associati alla nota al fine di abbellire il suo suono. Ad esempio, il trillo consiste nell'alternare il suono di una nota con un suono immediatamente superiore. Di solito, questi simboli producono un effetto solo sulla nota alla quale sono correlati. La rappresentazione grafica di alcuni di questi simboli può essere utile ad esprimere la durata dell'effetto come nel caso del trillo. Esso infatti ne definisce, con la sua lunghezza, la durata ed individua i punti del rigo musicale nei quali inizia e termina.

Una nota può avere simboli di espressione (detti anche articolazioni): come *martellato, staccato, accento, tenuto, sforzato.* La sequenza e la relativa posizione dei simboli di espressione seguono un insieme di regole grafiche ben definite. In generale, i simboli di espressione interessano la durata e, in alcuni casi, anche la dinamica. Ad esempio, il *martellato* è rappresentato usando un piccolo cuneo pieno sopra o sotto la testa della nota per indicare che il suono deve essere come quello di un martello; lo *staccato* è rappresentato con un punto sopra o sotto la testa della nota e riduce la durata della nota di circa la metà.

Tutti i simboli di espressione possono essere messi sopra o sotto la testa della nota. La loro collocazione dipende dalla posizione della testa della nota sul rigo musicale; di solito vengono posti dalla parte opposta del gambo, se questo è presente. Ogni nota (nella musica tradizionale) può presentare più simboli di espressione contemporaneamente ad esempio un'indicazione di *staccato sforzato.* Anche in questo caso, la rappresentazione visuale e la posizione del simbolo danno un'idea immediata dell'effetto desiderato. Inoltre, molti altri simboli di espressione sono stati recentemente introdotti dalla musica moderna.

La **dinamica** è un concetto soggettivo che specifica l'intensità del suono. Ad esempio differenti gradazioni di piano: (pianississsississimo) *pppppp, ppppp, pppp, ppp, pp*; forte: *f, ff, fff, ffff, fffff, ffffff* e valori medi: mezzo piano *mp* , mezzo forte *mf* e molti altri: *sf, fp, fz, ffz.* I simboli di dinamica sono tradizionalmente posti al di sotto del rigo musicale.

I simboli di dinamica sono indicazioni non assolute in quanto dipendono dal contesto. La stessa dinamica ha un diverso significato se è applicata ad un solista o ad un'orchestra, dal momento che in un'orchestra l'intensità del suono è il risultato della combinazione dei suoni prodotti dai singoli strumenti. Tali simboli cambiano il contesto dell'interpretazione e restano validi fino a che non vengono introdotte differenti indicazioni. Inoltre la dinamica generale può essere cambiata sulla base di indicazioni specifiche e contingenti tali da accrescerla (*crescendo*) o diminuirla (*diminuendo*). Questo genere di indicazioni può essere formalmente introdotto sia in maniera testuale (*dimin.*) che attraverso l'uso di simboli grafici. I simboli grafici sono molto più efficaci in quanto sono più facilmente interpretabili. La loro direzione indica se l'intensità deve essere aumentata o diminuita, mentre i punti di applicazione (nota di inizio e di fine) indicano esattamente l'intervallo di validità. L'uso

dei simboli grafici in luogo della versione testuale permette la rappresentazione di effetti più sofisticati come un *crescendo* strettamente connesso con un *diminuendo*: <>.

3.2 Aggregazione di simboli

Nella notazione musicale esistono cinque principali tipi di meccanismo sulla base dei quali vengono raggruppate le figure o vengono rappresentati gruppi di figure. Il loro significato è molto diverso:

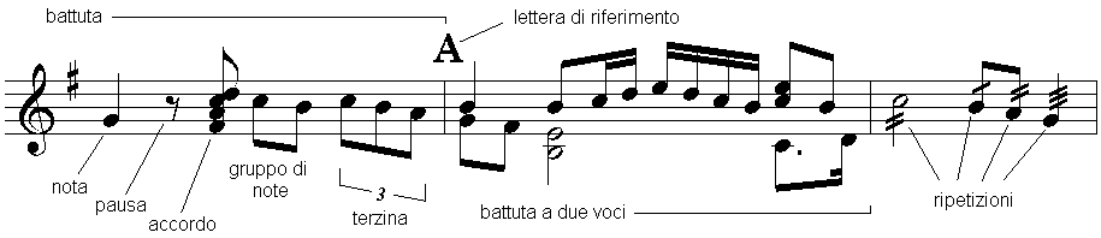


Figura 3.2: Gruppi di figure: accordi, tuple, voci, gruppi di note e ripetizioni.

- Gli accordi sono gruppi di note, sovrapposte verticalmente, che devono essere eseguite simultaneamente. Strumenti come il pianoforte possono eseguire accordi. La posizione della testa delle note dipende da specifiche regole. L'accordo può presentare abbellimenti come l'*arpeggiato*. In alcuni casi viene usato un differente modo di rappresentare gli accordi: si basa sull'indicazione della nota principale sul rigo musicale, mentre le altre note dell'accordo sono lasciate alla decisione del musicista, altrimenti vengono specificate usando i loro relativi numeri: 3, 5 per la terza e la quinta nota sopra la nota data.

- I gruppi di note sono composti da note unite con una larga linea trasversale (singola o multipla in relazione al numero delle code presenti). È un diverso modo di visualizzare un insieme di note per rendere più facile la loro lettura. L'orientamento della linea tra le note viene definito sulla base di regole ben conosciute.

- Le tuple sono gruppi irregolari di note. Una tupla ha una durata nella battuta che risulta inferiore alla somma della durata delle singole note che la compongono. Ad esempio una terzina, se è composta da tre note da un quarto ciascuna, dovrà essere eseguita in due quarti. Questi gruppi irregolari possono contenere delle pause e annidare a loro volta altre tuple.

- La presenza di più di una voce sul rigo musicale viene rappresentata attraverso una sequenza di note per ciascuna voce. Tale tecnica di rappresentazione è usata per

strumenti a più voci o in partiture nelle quali si vuole rappresentare due o più parti sullo stesso rigo musicale (polifonia).

- Le ripetizioni sono spesso usate per rappresentare in maniera concisa una parte del brano musicale. In particolare la presenza di una, due, tre e talvolta quattro piccole linee sul gambo di una nota sta a significare che la nota deve essere interpretata come una sequenza di note distinte del valore rispettivamente di un ottavo, un sedicesimo, un trentaduesimo e un sessantaquattresimo. Tale rappresentazione grafica permette la scrittura di brani complessi in un piccolo spazio. Per questo motivo, viene usata nella stampa delle partiture di musica tradizionale.

3.3 La struttura delle partiture

Nella notazione musicale, le figure sono organizzate in spazi limitati da barre, detti battute (si veda la partitura riportata in Figura 3.3: essa presenta cinque battute). Ciascuna battuta è composta da un numero definito di unità di tempo (pulsazioni) che dipendono dall'espressione del tempo (4/4, 2/4, 6/8). Ad esempio 3/4 significa che sono necessarie tre note della durata di 1/4 per completare la battuta. Pertanto nella battuta deve esserci consistenza tra l'informazione contenuta nell'espressione del tempo e la durata delle figure in essa contenute.

Una sequenza di battute rappresenta un brano. Questo può contenere indicazioni di passaggio da un punto ad un altro, da una battuta ad un'altra per ripetere sequenze musicali (ritornelli). Alcuni tipi speciali di barre servono a segnalare l'inizio e la fine dei ritornelli. In alcuni casi, vengono usate delle lettere per identificare i punti dai quali l'orchestra deve ripartire durante le prove, ad esempio, per studiare i pezzi più difficili del brano da eseguire.

Di solito, se il brano musicale è per un gruppo di strumenti (partitura) vi sarà una parte per ciascun strumento e tali parti saranno allineate verticalmente l'una con l'altra rispetto alle singole battute. Quando uno strumento non deve suonare, nella sua parte sarà inserita una pausa così come accade nella terza battuta del violino nella Figura 3.3.

Quindi, nelle orchestre, il direttore legge la partitura osservando le battute di tutti gli strumenti sulla stessa colonna; i musicisti invece vedono solo la propria parte. Nelle partiture, i righi musicali di strumenti simili sono collegati utilizzando parentesi quadre o graffe. La posizione delle figure nelle battute allineate dipende dalla lunghezza della battuta e dal valore della figura stessa.

All'inizio di un rigo musicale, e quando è necessario, deve essere dichiarata una chiave, ad esempio:

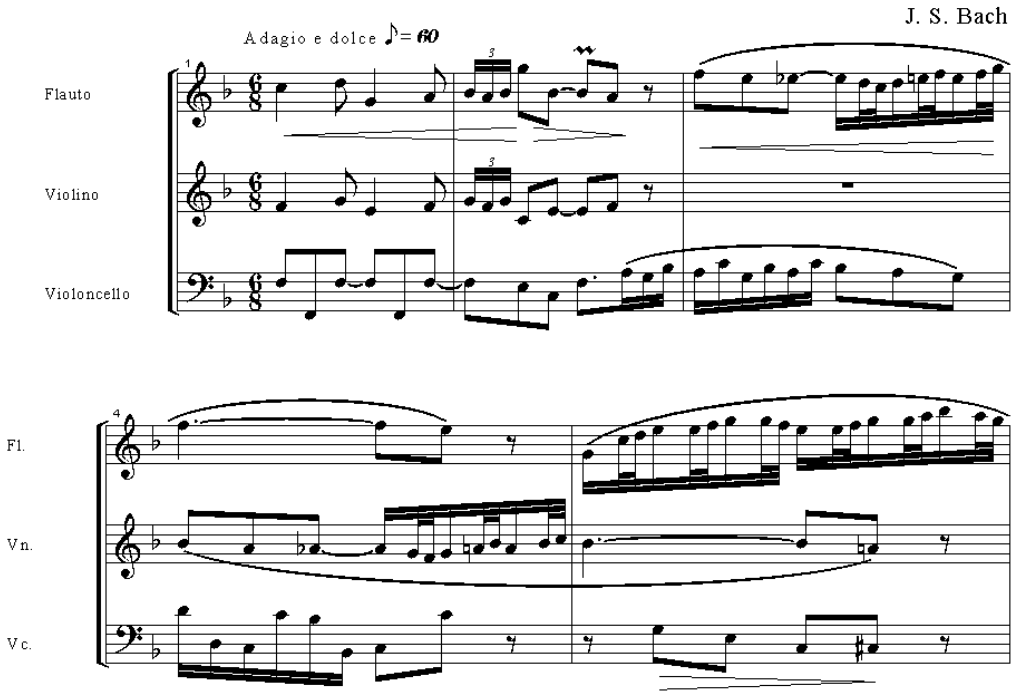
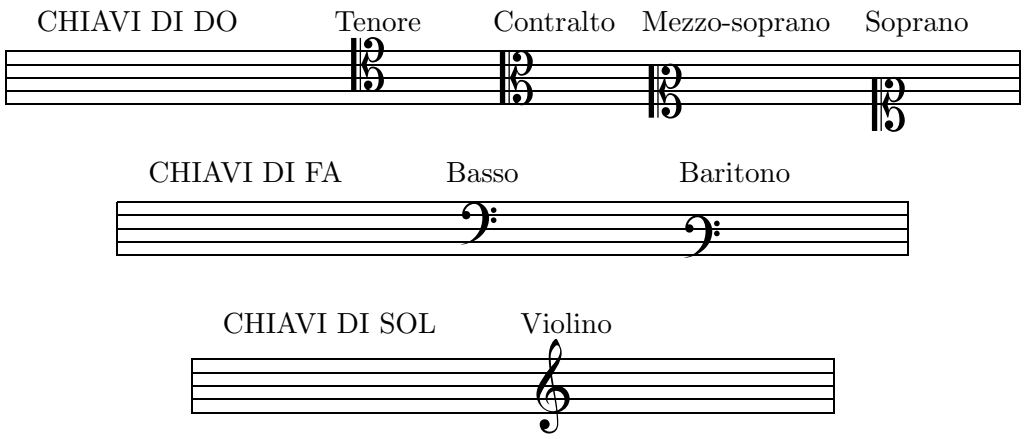


Figura 3.3: Una partitura e le sue parti.



Ciascuna chiave deve essere collocata in una ben precisa posizione nel rigo musicale. La chiave specifica il modo in cui si devono interpretare le linee e gli spazi nel rigo musicale; pertanto l'interpretazione resta la stessa fino a quando non viene posizionata una nuova chiave. La chiave può avere un'armatura di chiave (p.e., un gruppo di diesis o bemolle) per modificare il tono delle note nel rigo musicale. Una nuova chiave può essere inserita direttamente all'interno della battuta, anche nel mezzo della stessa. Inoltre, la partitura del direttore è normalmente formattata in modo diverso rispetto a quella dei musicisti. Essa risulta essere differente poiché:

- Le parti sono stampate utilizzando versioni compresse di simboli di notazione musicale, pause di durata pari a più battute sono rappresentate con un unico simbolo sulla parte mentre nella partitura sono inserite pause per ogni singola battuta.
- Le parti normalmente presentano i simboli generali sopra menzionati e numerosi simboli strumentali che vengono esclusi dalla partitura. Inoltre altri simboli di interpretazione o personali possono essere presenti nelle parti.
- La partitura può presentare specifiche note testuali o simboli di aiuto per il direttore.
- La partitura può presentare un unico rigo musicale per più di una parte, in questo caso le parti vengono trattate come voci distinte sullo stesso rigo musicale. Ad esempio il primo ed il secondo violino.
- Nelle parti, vengono aggiunte altre indicazioni per meglio coordinare i musicisti, ad esempio, incisi di un altro strumento, in formato ridotto (dette guide), i quali servono per aiutare il musicista ad attaccare nell'esecuzione al momento opportuno attraverso l'ascolto dell'esecuzione di specifiche note di altri musicisti.

3.4 Simboli orizzontali di notazione musicale

Fino ad ora, sono state descritte solo semplici relazioni tra i simboli. Nella notazione musicale esistono relazioni tra simboli più complesse nel momento in cui si analizzano simboli che si riferiscono a due figure lungo il rigo musicale. Tali figure possono appartenere a differenti battute e, in alcuni casi, a differenti parti, per strumenti che leggono la musica su più di un rigo musicale. I principali simboli orizzontali, detti in questa sede *simboli di intervallo*, sono:

- I simboli già discussi con la dinamica come il *diminuendo*.
- Le già discusse indicazioni temporali contingenti come: *rallentando, accelerando* con le loro rispettive rappresentazioni grafiche di onda e di freccia.
- Il segno di ottava che viene posizionato sopra o sotto il rigo musicale attraverso una parentesi quadra tratteggiata indicante che le note ivi contenute devono essere eseguite un'ottava sopra o sotto a quella corrente. L'adozione di questo simbolo viene fatta per migliorare la leggibilità della musica in quanto riduce la necessità di usare linee addizionali sopra o sotto il rigo musicale.
- La legatura di valore è un arco tra due note consecutive della stessa tonalità. Essa indica che le due note devono essere considerate come un'unica nota di durata pari alla somma della durata di ciascuna di loro. Una legatura di valore può essere

interrotta alla fine della pagina, in questo caso, nella pagina successiva dovrà ripartire dalla metà dell'arco.

- La legatura di espressione è un arco tra due note che possono non essere consecutive e possono non avere la stessa tonalità. Pertanto tale legatura significa che le note comprese al suo interno devono essere eseguite con continuità (legato).

Questo genere di simboli è totalmente asincrono rispetto alla sequenza delle battute, ciò può essere notato osservando le Figure 3.3,3.4.

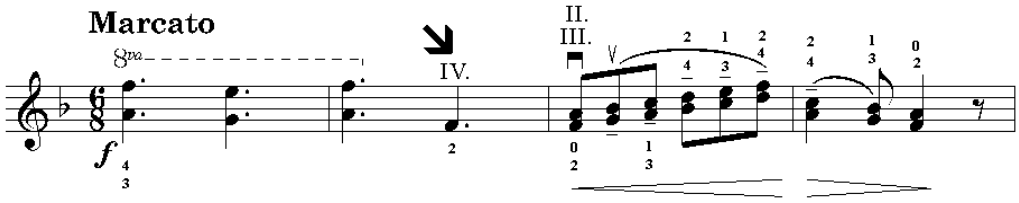


Figura 3.4: Simboli orizzontali ed altri simboli addizionali.

3.5 Simboli generali e strumentali

Nelle partiture musicali possono essere aggiunti molti altri simboli che vengono classificati in due principali categorie:

- I **simboli generali**. Ad esempio, un simbolo per indicare che:
  - Una battuta da 2/4 sarà eseguita dal direttore in 4/4 o viceversa (quattro linee distribuite lungo la battuta);
  - Il musicista deve porre particolare attenzione ad uno specifico punto (una coppia di occhiali o una grossa freccia trasversale);
  - Il musicista deve prendere fiato (una virgola, una V o una |).

- I **simboli strumentali**. Per gli archi: *ponticello, numero della corda (in numeri romani), diteggiatura, punta, tallone, arco su, arco giù, pizzicato.* Per i timpani: la regolazione del tono successivo. Per l'arpa ed il pianoforte l'indicazione dei pedali che devono essere premuti. Per gli ottoni l'adozione di specifici registri per produrre particolari note. Per i tamburi: l'adozione di vari tipi di bacchette: metalliche, a spazzola, ecc.. Per molti strumenti la presenza o meno della sordina o la produzione di suoni armonici. Molti altri simboli possono essere aggiunti per soddisfare le necessità di tutti gli strumenti di una grande orchestra.

Molti dei simboli appena descritti assumono rappresentazioni grafiche o testuali differenti in relazione allo strumento per cui vengono usati. Ad esempio l'inserimento o la rimozione della sordina può essere graficamente rappresentato con l'uso di una "E" orizzontale come nella Figura 3.1 per gli archi, o con un "+" (inserire la sordina) o un "-" (togliere la sordina) per gli ottoni, o con un'indicazione testuale: *con sordina, via sordina*.

Alcuni di questi simboli cambiano il contesto dell'interpretazione e dell'esecuzione. Ad esempio, *con sordina* significa che la sordina deve essere inserita e presuppone che da qualche parte sarà presente l'indicazione di *via sordina*. D'altra parte, molti simboli di interpretazione, che sono di solito trascurati nell'editoria elettronica musicale, sono invece molto rilevanti per le esecuzioni di musica nei teatri e per le lezioni nelle scuole di musica.

Capitolo 4

Architettura generale del O^{3}MR

L'idea dello sviluppo di un sistema OMR completo è nata dall'esigenza di un metodo di inserimento automatico della musica da integrare nell'ambiente WEDELMUSIC (Web Delivery of Music): sistema di rappresentazione grafica di simboli musicali, alla base del quale è presente una formalizzazione strutturata (Object Oriented) della sintassi e della semantica della notazione musicale, e un formato per il salvataggio dell'informazione musicale basato sul linguaggio XML ([57]). L'insieme degli spartiti musicali presi in considerazione nello sviluppo del sistema di riconoscimento è quello relativo al repertorio di musica classica dal XVII Secolo ad oggi e scritta con notazione occidentale. In particolare l'attenzione è stata rivolta a brani monofonici e polifonici a singola voce (*single layer*).

L'approccio utilizzato nello studio e nella progettazione è stato concreto e pragmatico: si è cercato di affrontare fin dall'inizio il problema nella sua complessità e di elaborare soluzioni che non fossero parziali o mirate ad una particolare applicazione. Quindi anche nella scelta delle tecniche e delle metodologie da utilizzare si è data una particolare importanza alle possibilità di evoluzione e di sviluppo del sistema verso il riconoscimento di spartiti musicali sempre più complessi e costituiti da più voci (estensione *multi layer*).

4.1 Struttura del sistema

In questo paragrafo viene presentata un'analisi di carattere generale dei moduli operativi che compongono il sistema (si veda Figura 4.1) di riconoscimento automatico degli spartiti musicali.

- Modulo di Segmentazione: in ingresso al modulo viene fornita un'immagine digitalizzata - a 8 bit (256 tonalità di grigio) e risoluzione minima di 300 d.p.i. - dello spartito musicale. L'immagine, preventivamente ripulita del testo, viene analizzata e decomposta attraverso il calcolo e l'elaborazione condotta su tre livelli: la pagina intera, il pentagramma, il paragrafo elementare di pentagramma. Questa procedura

di estrazione progressiva produce una scomposizione dello spartito in elementi grafici primitivi ad ognuno dei quali è associato, oltre ad un file grafico che contiene la relativa immagine, un insieme di parametri (caratteristiche topologiche, strutturali e di contesto) necessarie ai passi successivi.

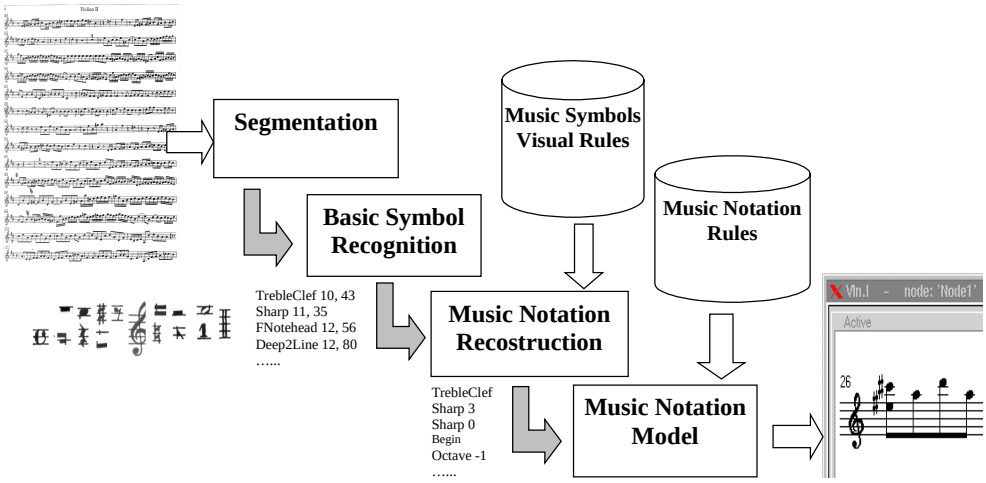


Figura 4.1: Moduli operativi del sistema di riconoscimento automatico degli spartiti musicali.

- Modulo di Riconoscimento Primitive: utilizzando un insieme precostituito di *primitive grafiche*, attraverso delle operazioni di *pattern recognition* sviluppate con l'utilizzo di un modello a Reti Neurali, sono identificati gli elementi grafici ricavati dalla segmentazione e a ciascuno di essi è associata una percentuale di confidenza sul riconoscimento. Successivamente, un'analisi basata sull'informazione di contesto derivanti dal processo di segmentazione consente di effettuare un recupero dell'errore di riconoscimento legata alla confusione cui può incorrere la rete neurale. L'uscita di questo modulo è principalmente simbolica. Per ciascun simbolo di base riconosciuto, sono fornite le coordinate grafiche, il valore di confidenza e le dimensioni del "bounding" box; parallelamente, vengono aggiunte informazioni di contesto (appartenenza a gruppi di note, durata, riferimenti sull'andamento delle linee del pentagramma e di fine pentagramma).

- Modulo di Ricostruzione Simboli: utilizzando le informazioni associate ad ogni elemento grafico e le corrispondenze determinate al passo precedente, vengono ricostruiti e riconosciuti i simboli musicali, tramite l'utilizzo di regole grammaticali e posizionali. Queste regole permettono di ricostruire, a livello locale, dei simboli sintatticamente corretti basandosi sul contesto: la posizione del simbolo di base rispetto alla posizione delle linee del pentagramma, il livello di confidenza e le relazioni sia orizzontali che verticali con gli altri simboli. È importante osservare che le informazioni derivanti dalla fase di riconoscimento possono essere parziali e non

corrette. In uscita vengono prodotti tutti i simboli musicali, completi di proprietà e relazioni con il contesto.

- Modulo di Rappresentazione dello Spartito: questo modulo si occupa di ricomporre lo spartito nella sua forma completa e di effettuare una serie di controlli sintattici e semantici, a livello globale. Il formato di rappresentazione della notazione musicale usato è stato sviluppato, con tecniche object oriented, ed è relativo al formato WEDELMUSIC. La formalizzazione della sintassi musicale presente in questo modulo è utilizzata nelle fasi di costruzione della simbologia e di composizione dello spartito.

Il sistema può essere anche esaminato da una prospettiva, non di carattere procedurale ma conoscitiva, basata sulla distinzione tra livelli di conoscenza acquisita in ogni fase. Durante il processo, i quattro moduli operativi determinano il passaggio tra i seguenti livelli (indicati schematicamente): immagine in pixel, elementi grafici di base, primitive etichettate, simboli musicali, spartito. Se consideriamo l'immagine digitale come lo strato più basso di conoscenza e lo spartito quello più alto, è facile individuare (anche graficamente) che la procedura utilizzata è di tipo *bottom up*. Ma negli ultimi moduli operano delle procedure di valutazione e correzione dei risultati che indicano che ai livelli più alti si può o meno accettare la conoscenza acquisita negli strati inferiori: in questo senso l'utilizzo delle retroazione può essere visto come un approccio *top down*.

Gli studi e le valutazioni che hanno ispirato il progetto del sistema sono discussi nei prossimi paragrafi, suddivisi per fasi operative.

4.2 Segmentazione significa semplificazione

L'informazione musicale viene espressa attraverso notazioni molto complesse, in modalità e stili talvolta imprevedibili. I simboli musicali alla base delle notazioni sono molto numerosi e possono essere variamente combinati tra loro, spesso seguendo criteri grafici non ben definiti.

Un metodo per ridurre la complessità del problema è quello di considerare la notazione come composta da elementi grafici di base (**primitive**), la cui combinazione permette di ricreare tutti i simboli musicali. In letteratura questa impostazione è stata utilizzata da diversi autori: si trovano, però, molte differenze sulla scelta delle forme grafiche primitive, anche se di solito si preferiscono elementi come la testa "piena" e "vuota" delle note, il loro gambo, le barre dei gruppi, le alterazioni, i punti, le pause, ecc.

Il processo di scomposizione dell'immagine musicale in elementi di base generalmente è chiamato *segmentazione*. A seconda del tipo di analisi e di elaborazione grafica che si

intende fare, questo procedimento può diventare anche piuttosto complesso e richiedere molte risorse grafiche e computazionali.

È importante sottolineare come la scelta del metodo di segmentazione è strettamente legata alla tecnica di pre-elaborazione grafica dell'immagine (es. identificazione/rimozione del pentagramma) e al sistema di classificazione previsto per le primitive. Infatti, nella stessa definizione delle primitive, è importante capire se sono presenti o meno porzioni delle linee del pentagramma e non tutte le tecniche di estrazione delle primitive presenti in letteratura sono compatibili e interscambiabili con i relativi metodi di identificazione.

4.2.1 Disposizione dei simboli sullo spartito

In questo progetto, durante la fase di analisi, è stata data una particolare importanza alla composizione grafica della pagina di uno spartito musicale. Indipendentemente dal fatto che la partitura preveda uno o più pentagrammi, l'"aggregazione" di informazioni, che si ha intorno ad essi, suggerisce di ridurre l'elaborazione di una pagina di musica ad una serie di analisi di carattere più locale, un pentagramma per volta.

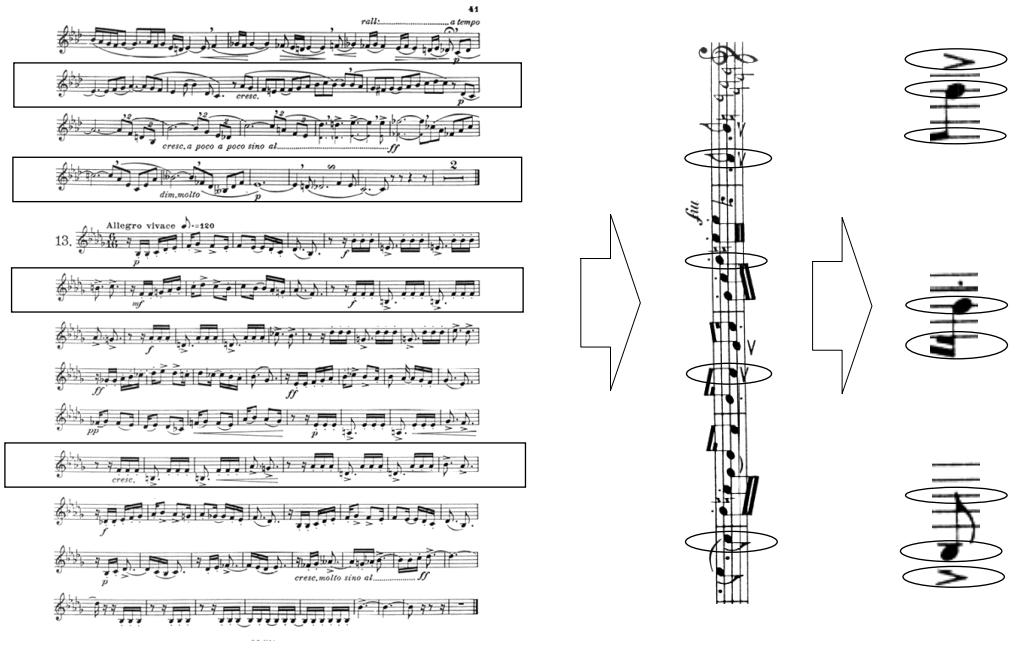


Figura 4.2: Nella figura viene messa in evidenza l'"aggregazione" grafica delle informazioni riscontrabile ad ogni livello.

Se poi, con uno scorrimento da sinistra a destra su ogni singolo pentagramma, si analizza la disposizione della notazione musicale, ci si accorge che generalmente i simboli possono essere "raccolti" in strisce verticali, larghe più o meno quanto la testa di una nota

(si veda Figura 4.2); questo significa che dal punto di vista grafico è possibile ridurre lo studio di uno spartito a quello di elementi ancora più piccoli e più semplici.

L'analisi che è stata fatta finora non tiene conto del significato dei simboli musicali, ma evidenzia solo quali sono le zone più ricche dal punto di vista grafico, cioè dove è più consistente la presenza di pixel neri[1]. Se con gli stessi criteri si osservano, dall'alto in basso, le piccole strisce individuate al passo precedente, anche in questo caso è possibile isolare in modo molto chiaro un certo numero di sotto aree più ricche di elementi grafici.

Queste osservazioni, fatte in maniera immediata e superficiale semplicemente osservando una pagina di uno spartito musicale, trovano un riscontro più preciso se si utilizzano semplici tecniche di analisi grafica: in questo senso la più indicata è sicuramente la **proiezione**. Essa viene detta *proiezione Y* se è fatta in senso orizzontale e consiste nella somma su tutte le righe dei valori dei pixel. La *proiezione X*, fatta in modo analogo nel senso verticale, è la somma su tutte le colonne dei valori dei pixel. Sia che l'immagine sia in bianco e nero che in tonalità di grigio è importante, nel calcolo della proiezione, assegnare al colore nero il valore più alto[2], in modo che le aree dell'immagine con maggiore densità di simboli siano associate a proiezioni con valori più alti.

4.2.2 Le proiezioni e le nuove primitive

La proiezione Y calcolata su tutto lo spartito musicale presenta dei picchi molto evidenti in corrispondenza dei pentagrammi. Un risultato analogo, ma con risvolti più delicati, si ha effettuando la proiezione X su un singolo pentagramma: in questo caso tutte le linee orizzontali determinano uno scalino costante sul profilo, ma questo non impedisce di osservare delle piccole "gobbe" che corrispondono ai raggruppamenti su linee verticali della simbologia. Infine, applicando una nuova proiezione Y ad ogni piccola striscia ricavata dal pentagramma, vengono evidenziate con molta precisione le parti più consistenti dei simboli musicali (teste delle note, gambi, frazioni di barre di gruppi, ecc.).

Il procedimento descritto mette in evidenza che, la complessità della notazione musicale può essere ridotta notevolmente se si individuano gli elementi grafici primitivi che ne stanno alla base. Non è detto che essi debbano corrispondere a parti ben definite di singoli simboli musicali (questa è un'idea che si rifà, probabilmente, al significato che diamo ad ogni segno della simbologia la testa vuota o piena, la forma del gambo, ecc.), ma possono essere semplicemente frutto di una serie di operazioni grafiche di taglio ed estrazione (si veda Figura 4.3).

[1] In questo caso, per semplicità, si considera l'immagine musicale digitalizzata ad un bit (un colore) in modo tale che lo sfondo della pagina è bianco e qualsiasi elemento grafico su di essa è nero.

[2] Per ragioni che verranno evidenziate in seguito, si preferisce lavorare con spartiti musicali digitalizzati a 256 tonalità di grigio (8 bit); siccome nei formati di salvataggio delle immagini spesso viene associato al bianco il valore 255 e al nero il valore 0, nel calcolo della proiezione si deve operare una trasformazione.

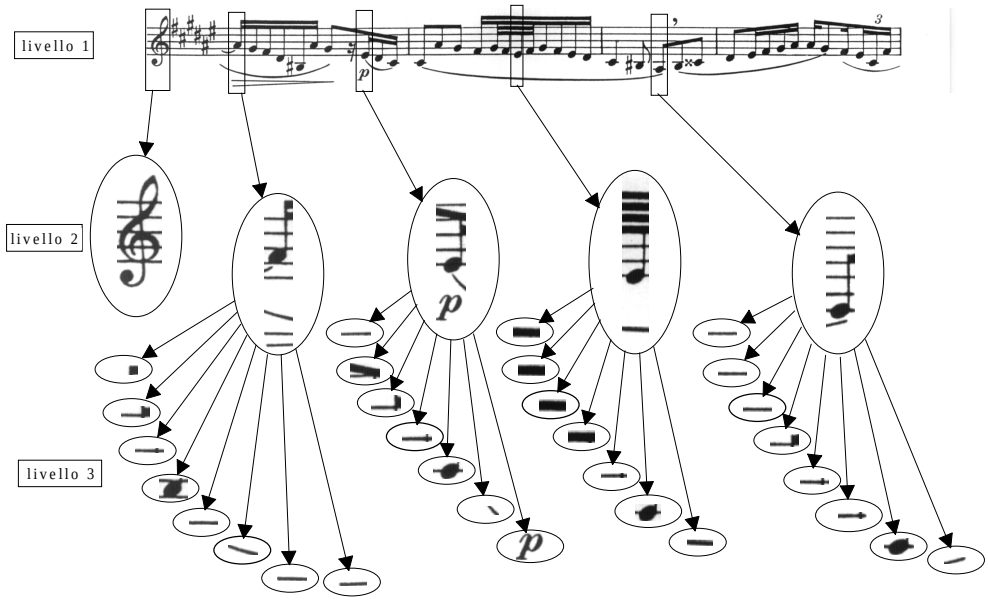


Figura 4.3: Nella figura viene mostrato come il segmentatore opera per ottenere le primitive.

In questo senso, quindi, l'analisi dell'immagine, attraverso le proiezioni, può rappresentare un efficiente metodo di segmentazione e di estrazione delle forme grafiche alla base della simbologia. L'insieme delle primitive che ne deriva, al contrario di quello che si potrebbe pensare, è ben definito e non molto complesso.

4.2.3 Perché rimuovere il pentagramma

In molti dei sistemi OMR sviluppati in letteratura l'identificazione e la rimozione del pentagramma rappresentano dei passi obbligati da compiere nella fase precedente alla segmentazione e alla ricerca dei simboli musicali.

In questo progetto, invece, si ritiene, vista la tecnica di segmentazione utilizzata, che le linee del pentagramma debbano essere parte integrante delle primitive. Ciò significa che durante il processo di frammentazione dello spartito, il peso grafico che è dato dalle linee del pentagramma ha la stessa importanza di quello dato dagli altri simboli[3]. Secondo questa impostazione, faranno parte dell'insieme delle primitive elementi come, per esempio, la testa della nota, toccata sopra e sotto da tratti di linee, oppure le chiavi, attraversate da tutto il pentagramma.

La scelta di questo tipo di segmentazione si rivela molto utile, anche perché, oltre

[3]Il problema è un po' più complesso e per studiarlo a fondo (come verrà fatto nei prossimi capitoli) è necessario distinguere il ruolo delle linee del pentagramma nelle varie fasi della segmentazione. È comunque vero che, nell'analisi dettagliata dello spartito (ultima fase di segmentazione con proiezione X calcolata su piccole strisce verticali), i segmenti delle linee di pentagramma presenti vengono trattati come qualsiasi altra parte della simbologia.

alla facilità di implementazione che essa comporta, porta a risolvere molti dei problemi aperti nello studio dell'OMR: non si producono oggetti spezzati dalla cancellazione del pentagramma; non sono prodotte distorsioni dei simboli provocata da i metodi di identificazione (e rimozione) di forme primitive; il problema della sovrapposizione o la tangenza dei simboli viene implicitamente risolto dal punto di vista grafico.

4.3 Riconoscimento delle primitive

Nella scelta del metodo di segmentazione dell'immagine si deve tenere conto anche di quale metodologia adottare nella successiva fase di identificazione e classificazione delle primitive. Il termine primitive sta ad indicare quegli elementi grafici di base, la cui combinazione permette di ricreare tutti i simboli musicali. Qualsiasi sia la metodologia impiegata per il riconoscimento di questi "mattoni" fondamentali della notazione, essa si basa sempre su un gruppo di modelli di riferimento chiamato, di solito, *insieme delle primitive*. Bisogna fare attenzione al fatto che talvolta il termine **primitive** viene utilizzato sia per indicare gli elementi risultanti dal processo di segmentazione che per descrivere l'insieme dei modelli di riferimento.

In letteratura sono molti casi in cui la semplice conformazione delle primitive (generalmente corrispondenti a parti ben definite di simboli musicali) dà la possibilità di utilizzare tecniche di identificazioni basate su semplici proprietà grafiche: le dimensioni del *bounding box*, l'area, la posizione del baricentro, la forma delle proiezioni X e Y, la lunghezza di alcune paragrafi, ecc. In altri casi vengono utilizzate tecniche più complesse come il *contour tracing* e il *pattern matching*.

4.3.1 Variabilità della notazione

Le primitive, "mattoni" grafici fondamentali della notazione, per loro stessa natura sono meno sensibili alle possibili configurazioni e combinazioni dei simboli musicali. Ciò non toglie che la natura dinamica delle notazioni, che dipende dalla loro naturale evoluzione e dalle novità e le personalizzazioni introdotte dagli autori e dalle case editrici, possa modificare l'insieme delle forme grafiche primitive necessarie per la rappresentazione della musica[4].

Dunque, è auspicabile che gli strumenti utilizzati per l'identificazione e la classificazione delle primitive prevedano la possibilità di aggiungere nuovi elementi e di modificare quelli esistenti. In questo senso è necessario sottolineare che la simbologia musicale è particolarmente soggetta a variazioni, anche lievi, dei profili grafici, sia per le ragioni accennate

[4]Non è possibile fare una valutazione di carattere generale, perché la relazione tra la variazione delle forme della simbologia e la variazione dell'insieme delle primitive dipende da una serie di componenti tra le quali: il tipo di notazione utilizzata, quanti e quali simboli sono considerati scomponibili, quali primitive sono state scelte e così via.

sopra che per motivi tipografici. In altre applicazioni di riconoscimento ottico (come l'OCR) i cambiamenti dei font, delle spaziature, delle dimensioni, possono essere trattati con relativa facilità; nell'OMR, invece, se non vengono impiegate tecniche adeguate, anche leggere modifiche possono provocare gravi peggioramenti delle prestazioni.

4.3.2 Reti neurali

Le *reti neurali artificiali* (Artificial Neural Network) sono il risultato di una serie di ricerche nelle quali si utilizzano modelli matematici per simulare il sistema nervoso dell'uomo. Molte reti neurali artificiali sono ispirate a reti neurali biologiche e una parte della ricerca in questo campo deriva dal desiderio di produrre sistemi artificiali simili a quelli sviluppati nel cervello umano.

Anche se non esiste una definizione universalmente accettata, si può affermare che le reti neurali artificiali sono una rete di molti semplici elementi di calcolo (*neuroni*), ognuno dei quali può essere fornito di una memoria locale. Queste unità sono connesse attraverso canali di comunicazione (*sinapsi*) che trasportano un segnale numerico (non simbolico). Le unità operano solamente sui loro dati locali e sugli ingressi che ricevono dalle connessioni.

Molte reti neurali artificiali hanno alcuni meccanismi di "apprendimento" che lavorano in modo da aggiustare i pesi delle connessioni. In altre parole, esse "imparano" da esempi (provando e sbagliando) e in questo modo dimostrano delle capacità di *generalizzazione* al di là dei dati forniti inizialmente. Infatti, uno dei vantaggi principali che si ha nell'utilizzo di queste reti è che non è necessario conoscere in anticipo quali siano le esatte relazioni tra i dati che vogliamo esaminare; di conseguenza le reti neurali lavorano bene con dati disturbati, complessi, legati da relazioni non lineari. D'altra parte, uno degli svantaggi principali è che esse non forniscono una formula esatta, e nell'elaborazione dei dati e nella scelta degli ingressi, secondo che cosa si vuole modellare, possono essere necessari anche un numero molto alto di errori e di prove.

In pratica, le reti neurali artificiali sono particolarmente utili per la classificazione, la modellazione e l'approssimazione di quei problemi che sono tolleranti a qualche imprecisione e che hanno la disponibilità di un gran numero di esempi, ai quali, però, non possono essere facilmente applicate schematizzazioni e regole precise (come quelle utilizzabili in un sistema esperto).

Per la loro abilità a ricavare informazioni da dati imprecisi e complicati, le reti neurali sono già state utilizzate con successo in molte ricerche ed anche a livello industriale in innumerevoli settori. Inoltre, è stato riscontrato che con le reti neurali si possono ottenere risultati molto soddisfacenti nel *pattern recognition*; per questo motivo esse vengono utilizzate in modo sempre più rilevante nel campo del riconoscimento automatico dei caratteri (OCR).

4.3.3 Riconoscere imparando

Per le considerazioni fatte nei due paragrafi precedenti, le reti neurali sono tra gli strumenti più adeguati all'identificazione e alla classificazione delle primitive che otteniamo dalla segmentazione utilizzata in questo sistema.

Infatti, ognuna delle primitive, ottenuta dall'immagine musicale in base alle informazioni fornite dalle proiezioni, non ha sempre dimensioni ben definite, forma standard e contorni tagliati nello stesso modo. Il metodo di segmentazione utilizzato, però, ci garantisce che, di ogni elemento grafico primitivo, si possano ottenere (in modalità pressochè automatica) un numero molto grande di esempi e di prove. Quindi, in questo senso, la caratteristica di apprendimento delle reti neurali rispecchia esattamente le esigenze del riconoscimento in materia musicale, dove si ha una gran quantità di materiale, con notazioni, stili, scelte tipografiche e grafie mai esattamente identiche.

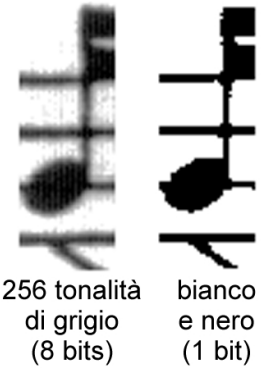


Figura 4.4: Particolare di uno spartito digitalizzato a 256 tonalità di grigio e in bianco e nero.

Dunque, è possibile affermare che, se i parametri di segmentazione (valutazione ed elaborazione dei dati delle proiezioni) sono impostati correttamente e di conseguenza l'insieme delle primitive è composto in maniera ottimale e completa, un sistema di identificazione basato sulle reti neurali può garantire un buon grado di efficienza e flessibilità (adattamento a situazioni nuove).

Dal punto di vista pratico è importante sottolineare che, la possibilità di ottenere risultati utili da un modello di ANN è ai dati che vengono forniti in ingresso. Per questa ragione si è optato per l'utilizzo di immagini digitalizzate a 256 tonalità di grigio (8 bit), in modo tale che i dati di ingresso alle reti neurali siano più ricchi di informazioni e ci siano maggiori possibilità di elaborazione e di confronto. Nella Figura 4.4 viene evidenziato un particolare di uno spartito digitalizzato a 256 tonalità di grigio a (sinistra) e in bianco e nero (a destra).

4.4 Ricostruzione dei simboli

L'ultima fase del sistema OMR è quella relativa alla ricostruzione dei simboli musicali. Per poter comprendere a fondo il modo di operare della fase di ricostruzione è utile osservare la Figura 4.5 nella quale vengono rappresentati gli ingressi e le uscite dei moduli costituenti il sistema OMR. Focalizzando l'attenzione sulla parte relativa al ricostruttore, rettangolo tratteggiato in Figura 4.5, si nota che gli ingressi a tale modulo corrispondono alle uscite della rete neurale. Analizzando la struttura degli ingressi si osserva che questa è formata da un nome che rappresenta l'associazione fatta dalla rete neurale alla primitiva grafica

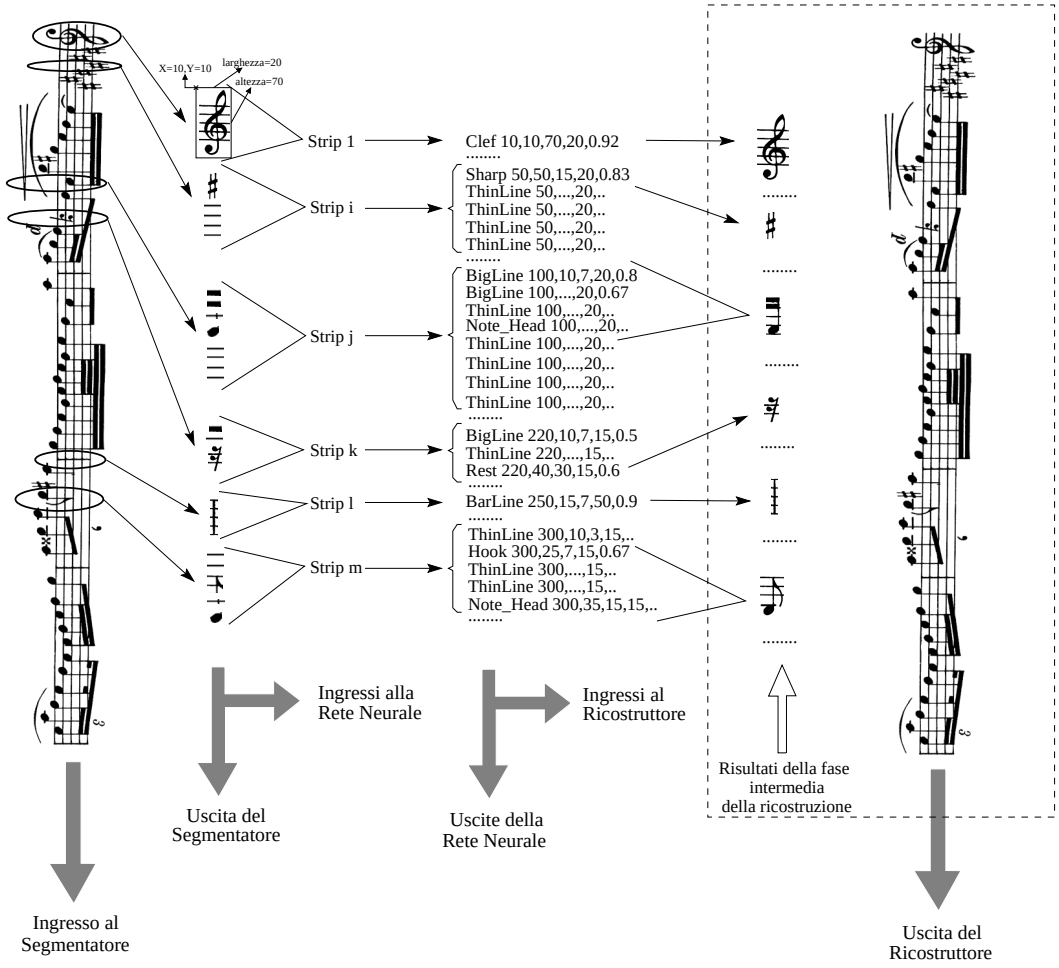


Figura 4.5: Processo evolutivo dallo spartito in ingresso al sistema OMR. Si noti che per ognuno dei moduli operativi (segmentazione, riconoscimento e ricostruzione) vengono riportati gli ingressi e le uscite.

in ingresso, etichetta della primitiva, quattro valori che rappresentano rispettivamente l'ascissa, l'ordinata, l'altezza e la lunghezza del bounding box che racchiudono la primitiva grafica in esame, infine un valore che rappresenta la confidenza con cui la la rete neurale ha riconosciuto la primitiva grafica. Quest'ultimo parametro, che può assumere valori compresi nell'intervallo [0,1], rappresenta la confidenza con la quale la rete neurale ha effettuato la classificazione della primitiva in ingresso: tale associazione è tanto migliore quanto più il livello di confidenza si avvicina ad 1. Quindi si ha che le etichette a cui sono associati valori del livello di confidenza prossimi ad 1 rappresentano quasi delle certezze mentre per valori minori di 0.5-0.6 la classificazione non è molto attendibile. Tornando all'analisi degli ingressi al ricostruttore si nota che questi sono raggruppati in **strip**. Con il termine **strip** si indicano quell'insieme delle primitive grafiche, e di conseguenza le etichette associate, ottenute dalla segmentazione delle strisce verticali del pentagramma

in ingresso.

Il ricostruttore, basandosi sui dati in ingresso precedentemente analizzati, deve riuscire a riprodurre, il più fedelmente possibile, lo spartito in ingresso. Come è lecito aspettarsi l'operazione di ricostruzione avviene procedendo in maniera inversa rispetto al modo di operare del segmentatore. In particolare si ha che tale processo viene effettuato tramite un meccanismo di aggregazione progressiva dei simboli in modo da ottenere, via via, simboli musicali di complessità crescente. Si procede, prima, aggregando i simboli elementari appartenenti alle singole strip componenti lo spartito, fase intermedia della ricostruzione, e poi aggregando tra loro i simboli appartenenti a strip diverse.

In letteratura si trovano diverse soluzione per la risoluzione del problema della ricostruzione. Attualmente si può osservare che sta sempre più prendendo campo l'uso della conoscenza musicale sia per la fase di segmentazione, prevalentemente, sia per quella di ricostruzione.

4.4.1 L'uso della conoscenza

Nella ricerca in campo OMR si rileva un interesse crescente nell'estensione dell'utilizzo della conoscenza musicale a tutto il sistema e non solo nella parte finale.

Per "conoscenza" in campo musicale non si vuole indicare solo le regole sintattiche musicali ma anche l'insieme delle regole che regolano la scrittura della musica, le quali andranno a formare delle grammatiche a livello grafico; sono appunto queste ultime che sempre più vengono considerate nei sistemi OMR. In alcuni casi si cerca di aggiungere anche una dimensione semantica.

Il linguaggio da utilizzare per la descrizione della conoscenza musicale è opportuno che abbia caratteristiche di modularità ed estendibilità. Queste caratteristiche fanno sì che tale grammatica possa essere ampliata o modificata senza che il sistema ne risenta (separazione tra la fase operativa e quella di definizione delle regole musicali).

Un esempio è dato da Couasnon e Camillerap (si veda paragrafo 2.11) per i quali un utilizzo della "conoscenza", nella fase finale del sistema, è riduttivo ed incide solo marginalmente sulle prestazioni del sistema stesso. Infatti secondo quest'ultimi è opportuno introdurre una grammatica per controllare la fase di segmentazione. Un altro esempio è dato dal progetto ideato da Fahmy e Blostein (si veda paragrafo 2.27) nel quale l'utilizzo della conoscenza avviene nella fase di ricostruzione dello spartito.

4.4.2 Verifica e correzione progressive

L'utilizzo esteso della conoscenza musicale è utile nella misura in cui riesce a creare un meccanismo di verifica ed eventuale ripetizione delle operazioni. È interessante studiare questa sorta di **retroazione** non solo applicata alla globalità del sistema (la correttezza o

meno dei risultati finali può implicare la ripetizione, magari locale, del riconoscimento), ma anche alle fasi intermedie, nelle quali, di solito, vengono compiute le scelte più importanti di tutto il processo.

Per esempio, invece di assegnare ad una primitiva un'etichetta definitiva, sarebbe utile, già in questa fase, verificare se le sue caratteristiche sono coerenti con il contesto e con le regole sintattiche che lo descrivono. Se ciò non si verificasse, probabilmente sarebbe opportuno segnare quella primitiva con un etichetta temporanea (modificabile in futuro) oppure ripetere le operazioni di identificazione, alla luce delle informazioni fornite dal contesto. Un comportamento analogo è auspicabile nell'algoritmo di ricomposizione delle primitive, il quale potrebbe, stavolta, con una verifica sintattica di più alto livello (simboli musicali completi), accettare o meno certe configurazioni di primitive oppure richiedere, addirittura, che venga ripetuta l'identificazione di una componente discordante con la struttura del simbolo attualmente in costruzione.

Come traspare dagli esempi precedenti, è chiaro che, il linguaggio che descrive la notazione musicale deve essere strutturato in modo da prevedere vari livelli di controllo sintattico. Nella fattispecie, per verificare la fase di *labelling*, c'è bisogno di un metodo per specificare le aggregazioni legali tra primitive; la verifica della ricostruzione dei simboli implica un livello di conoscenza più generale, quello riferito alle regole classiche alla base della simbologia.

Capitolo 5

Architettura del modulo di segmentazione

L'informazione musicale è espressa attraverso notazioni complesse, con modalità e stili talvolta imprevedibili. L'insieme dei simboli musicali alla base delle notazioni è ricco di elementi grafici combinabili tra loro, spesso seguendo criteri grafici non ben definiti. Per ridurre la complessità del problema è necessario considerare la notazione come composta da elementi grafici di base o primitive, la cui combinazione permette di ricreare tutti i simboli musicali. Per raggiungere questo obiettivo è necessario definire un processo di segmentazione, che partendo dall'immagine dello spartito identifichi i simboli di base in modo ripetibile e uniforme.

5.1 Struttura modulare del processo di estrazione

Un'immagine acquisita attraverso uno scanner è il punto di partenza del processo di segmentazione. Il metodo proposto si basa sulla decomposizione gerarchica dello spartito musicale. L'immagine è analizzata e ricorsivamente decomposta in blocchi di immagine via via sempre più piccoli attraverso la definizione di un insieme di linee di taglio verticali e orizzontali, che consentono l'isolamento e l'estrazione dei simboli di base.

Questo processo si basa su di 3 livelli successivi di elaborazione e una struttura modulare a catena aperta, all'interno della quale vengono definite le fasi di elaborazione che compongono ciascuno stadio della segmentazione. La Figura 5.1 mette in evidenza i componenti e i legami tra i livelli a partire dall'acquisizione dello spartito musicale fino alla sua frammentazione in simboli di base.

In una visione più generale i livelli di segmentazione e le uscite intermedie che essi producono possono essere definiti nel seguente modo:

- **Livello 0**: dato uno spartito musicale (con pentagrammi disposti in orizzontale) la

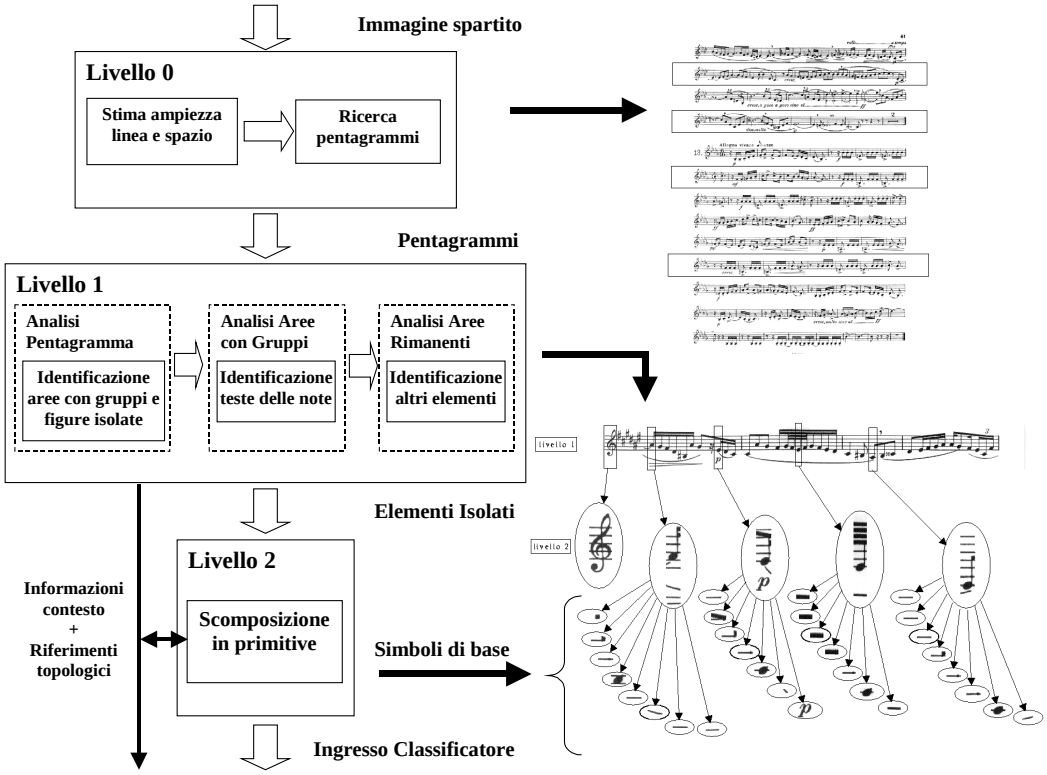


Figura 5.1: Diagramma a blocchi del processo di segmentazione

prima applicazione di tale procedura determina la creazione di nuove immagini che contengono i singoli pentagrammi.

- **Livello 1**: operando su ogni singolo pentagramma, vengono isolate ed estratte delle strisce elementari che contengono tutti gli elementi musicali presenti in una *fetta* verticale, larga generalmente quanto la testa di una nota.

- **Livello 2**: i simboli musicali individuati sono decomposti in simboli di base. In questa fase, sono previsti due metodi di decomposizione: uno per le immagini contenenti le teste delle note e uno per le rimanenti. In questo ultimo caso, l'immagine potrebbe contenere gli altri simboli.

Con riferimento alla Figura 5.1 è stato evidenziato il procedimento di estrazione dal livello 1 al livello 2 (come esempio sono riportate solo alcune strisce elementari). Si può già notare che i simboli di base, i quali possono essere individuati anche al secondo livello, possono corrispondere a simboli musicali completi oppure essere solo una parte di essi.

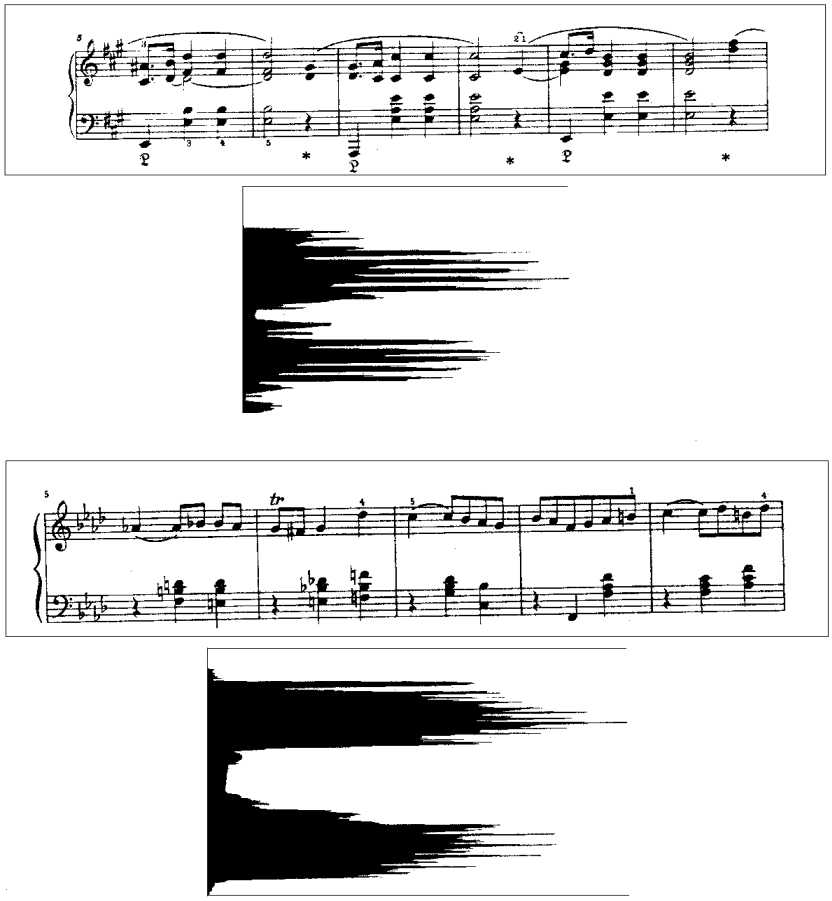


Figura 5.2: L'effetto dell'inclinazione sulla proiezione-Y

5.2 Livello 0 della segmentazione

Il primo passo verso la scomposizione dell'immagine musicale nei suoi elementi di base è associato al processo di estrazione dei pentagrammi (livello 0). Nell'ottica di un sistema automatizzato, tale procedura deve poter prescindere da qualunque forma di intervento umano che non sia, eventualmente, quello di inizializzazione del procedimento. Il metodo della proiezione-Y dello spartito, come punto di partenza per l'estrazione dei pentagrammi non si presenta come un procedimento automatico e non si presta facilmente ad un'estensione in tale direzione. I principali aspetti da risolvere sono legati alla ricerca di un criterio che stabilisca se effettuare o meno il filtraggio della proiezione, e qualora questo sia possibile, è preferibile caratterizzarlo con una parametrizzazione costante valevole per tutti gli spartiti. Oltre a ciò, è necessario individuare un valore di soglia e tolleranza di taglio specifica, in grado di estrarre ciascun pentagramma senza perdere informazioni, in modo da non compromettere i processi che seguono. Un problema imprescindibile, inoltre, è l'inclinazione della pagina, ed è considerato così importante che molti sistemi di

riconoscimento dedicano risorse nella fase di pre-processing dello spartito per ridurre al minimo la pendenza dei pentagrammi e l'inclinazione della pagina. La proiezione sull'asse verticale dell'intero spartito, sebbene sia in grado di mettere in evidenza la presenza delle linee di pentagramma, è però sensibile all'inclinazione dell'immagine ([32]). In questo caso (si veda Figura 5.2), i picchi nella proiezione non sono nitidi, ma si sovrappongono tra loro generando per fusione falsi picchi che possono essere successivamente riconosciuti, erroneamente, come linee di pentagramma.

Nonostante, quindi, queste difficoltà, la proiezione orizzontale è stato il punto di partenza per la ricerca e lo sviluppo di un metodo di identificazione dei pentagrammi, caratterizzato da un insieme minimo di parametri, con i quali modellare direttamente o indirettamente tutta la procedura, e poco sensibile al problema dell'inclinazione, così da avviare lo studio di un processo automatizzabile.

Queste considerazioni hanno condotto alla definizione di processi intermedi e indipendenti dalla tipologia dello spartito. Con riferimento al diagramma a blocchi nella Figura 5.1 il livello 0 è stato suddiviso in due fasi: (i) *Stima automatica delle ampiezze delle linee e degli spazi* e (ii) *Ricerca automatica del pentagramma*. Nella prima fase si ricercano le grandezze di riferimento che caratterizzano l'intero processo di segmentazione, sia per il livello corrente che per quelli successivi; nella seconda si procede all'identificazione e all'estrazione dei singoli pentagrammi, che risulteranno essere l'ingresso del livello 1.

5.2.1 Stima automatica delle ampiezze delle linee e degli spazi

È stata già evidenziata l'importanza che il pentagramma ricopre nel processo di riconoscimento, in questa sezione si considerano alcune informazioni che l'immagine di uno spartito musicale offre, che a prima vista non sembrano ovvie, ma che risultano essere importanti:

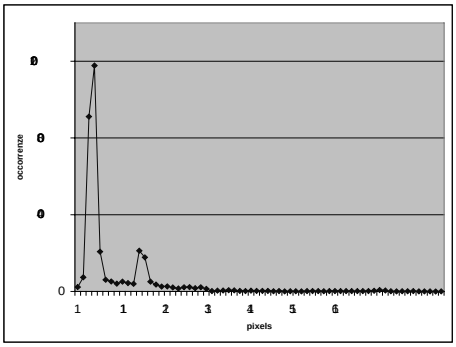
1. Le ampiezze delle linee e degli spazi tra le linee del pentagramma, se espresse in pixel, possono essere usate per definire soglie e valori di tolleranza per misurazioni e comparazioni.

2. La distanza tra i pentagrammi è utile per definire una misura per la normalizzazione di distanze e per conoscere la dimensione dello spartito.

L'attenzione, quindi, è stata rivolta all'esame delle proiezioni orizzontali di alcuni tratti di pentagramma ed è stata osservata la possibilità di descriverlo prendendo in considerazione due parametri grafici:

- l'ampiezza in pixel della singola linea di pentagramma

- l'ampiezza in pixel dello spazio fra due linee del pentagramma

a) 

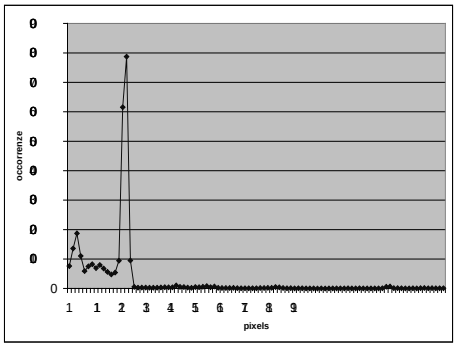
b) 

Figura 5.3: Profili delle sequenze di pixel a) neri e b) bianchi

La proiezione-Y di una linea di pentagramma presenta però un'istogramma variabile sia in forma che in ampiezza, dovuto principalmente al rumore e a quello di risoluzione nell'acquisizione, e non permette di fissare dei valori costanti, ma obbliga l'introduzione di due intervalli di tolleranza in modo da tenere conto delle possibili variazioni dello spessore e della distanza tra le linee. Gli estremi di tali intervalli sono così definiti:

- n_1 e n_2: rispettivamente il minimo e il massimo numero di pixel per lo spessore della linea di pentagramma

- d_1 e d_2: rispettivamente il minimo e il massimo numero di pixel per lo spazio tra due linee del pentagramma

Identificati i parametri descrittivi del pentagramma, occorre un metodo automatico [18] per ricavare una buona stima delle ampiezze prese come riferimento, senza alcuna conoscenza in merito, in modo da rendere il processo il più generale possibile. Resta di fatto che, anche se ottenuta una valida stima dei valori richiesti, sia comunque necessario considerare delle tolleranze di variazione.

Per stimare tali valori, la matrice dell'immagine dello spartito è scandita colonna per colonna per generare due istogrammi nei quali le sequenze di pixel bianchi e neri sono conteggiate in funzione delle loro dimensioni. Posizionando in ascissa il numero di pixel e in ordinata il numero di occorrenze, si ottengono due profili di sequenze verticali contigue relative rispettivamente ai pixel neri e bianchi (si veda Figura 5.3). I valori cercati coincidono con i massimi assoluti dei profili poiché il pentagramma è la struttura principale all'interno di uno spartito e il suo contributo in pixels è dominante. Infine, la determinazione delle tolleranze per lo spessore della linea e lo spazio tra due linee è eseguita prendendo l'approssimazione in difetto e in eccesso rispetto al punto di massimo individuato.

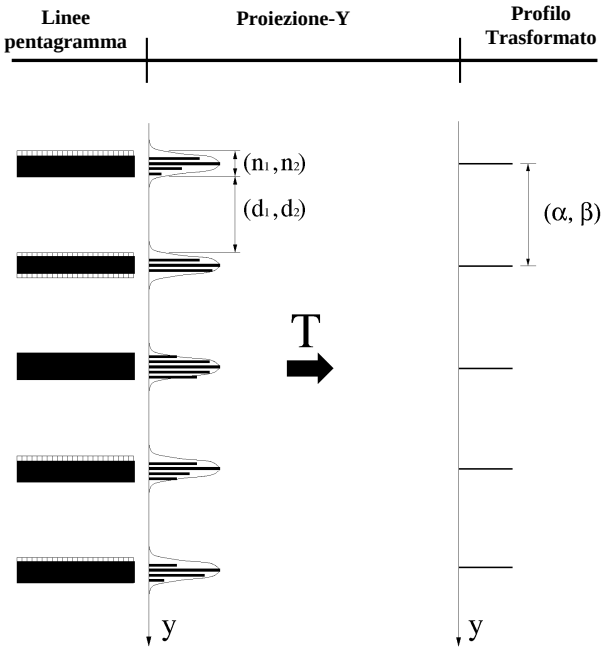


Figura 5.4: Caratterizzazione parametrica del pentagramma e trasformazione della proiezione-Y

5.2.2 Trasformazione della proiezione-Y

In questa sezione viene affrontato il passaggio preliminare che consente di definire in modo minimale i parametri scelti per la caratterizzazione del problema e che permette di costruire un criterio efficiente per la ricerca dei pentagrammi all'interno dell'intero spartito musicale. Con ciò, l'obiettivo di esprimere tutto in funzione di grandezze direttamente legate a proprietà grafiche che caratterizzano tutto lo spartito è raggiunto.

Partendo dall'istogramma associato alla proiezione-Y di una porzione dell'immagine, ottenuta mediante l'estrazione di una finestra rettangolare, si esaminano le sequenze continue di valori non nulli, corrispondenti a zone occupate da simboli con livelli di grigio diversi dal bianco. Dell'insieme delle sequenze si considerano tutte quelle con lunghezza (in pixel) entro i limiti di variazione dello spessore di una linea di pentagramma, e per ciascuna di esse si calcola il baricentro. Quelle che non rispettano la condizione imposta vengono trascurate e sostituite con una sequenza a valori nulli. In questo modo si costruisce il vettore dei baricentri esteso quanto la dimensione verticale massima dell'immagine in questione, contenente tanti 1 quanti sono i baricentri calcolati e posizionati nella coordinata y corrispondente, e 0 nelle restanti posizioni (si veda Figura 5.6).

Dati n_1 e n_2 si introducono le seguenti definizioni:

- Y_{max} la dimensione verticale massima dell'immagine

- $v[i]$ il vettore della proiezione-Y della porzione di immagine associata ad una finestra di scansione di larghezza dx e altezza Y_{max}
- $w[i]$ il vettore dei baricentri
- i l'indice di scorrimento per i vettori v e w corrispondente alla coordinata lungo l'asse y dell'immagine
- l_k la lunghezza della sequenza $k - esima$ a valori non nulli all'interno del vettore della proiezione-Y, della quale si calcola il baricentro
- i_k l'indice o la coordinata verticale del primo elemento della sequenza l_k diverso da zero
- C_k la coordinata del baricentro della sequenza l_k

Si definisce la seguente trasformazione[1]:

$$C_k = \frac{\sum_{j=i_k}^{i_k+l_k-1} v[j](j+1)}{\sum_{j=i_k}^{i_k+l_k-1} v[j]} \tag{5.1}$$

con la condizione:

$$n_1 \leq l_k \leq n_2 \tag{5.2}$$

Mentre il vettore w è definito come segue:

$$w[i] = \begin{cases} 1 \quad \text{per} \quad i = C_k - 1 \\ \\ 0 \quad \text{altrimenti} \end{cases} \tag{5.3}$$

Nel dominio dei baricentri, le condizioni sull'ampiezza di linea e di spazio si riducono alla sola condizione di distanza tra "picchi", ovvero lo spazio (numero di zeri) che intercorre tra due 1 consecutivi nella rappresentazione w. Anche in questo caso, non è possibile definire un unico valore di condizione, ma una coppia, poiché si eredita, per i baricentri, la variabilità sulla posizione, dovuta, sempre, alla diversità degli istogrammi della proiezione-Y. Il nuovo intervallo di tolleranza è direttamente esprimibile in funzione delle condizioni introdotte per lo spessore della linea e dello spazio ed è determinato attraverso

[1]Nella definizione del baricentro l'indice di scorrimento parte dal valore unitario, poiché nelle strutture indicizzate si preferisce partire dal valore nullo, è stato necessario aggiungere l'unità per non perdere il primo elemento. Successivamente, si riporta la coordinata del baricentro al suo valore coerente con l'indicizzazione sottraendo l'unità.

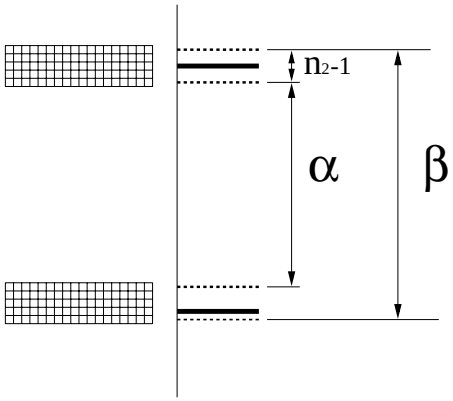


Figura 5.5: Valutazione dell'intervallo (α, β)

la valutazione della posizione del baricentro nel caso peggiore e nel caso migliore (si veda Figura 5.5) che permette di definire gli estremi α e β come segue:

$$\begin{cases} \alpha = d_1 \\ \\ \beta = d_2 + 2(n_2 - 1) \end{cases} \tag{5.4}$$

L'intervallo così definito completa l'insieme di condizioni da utilizzare nella costruzione dell'algoritmo di ricerca. Prima di descriverlo in dettaglio è bene considerare due problemi che dovranno essere affrontati: la presenza delle legature e l'inclinazione del pentagramma.

5.2.3 Le legature e l'inclinazione del pentagramma

La caratterizzazione introdotta e la modalità di costruzione del vettore w consentono di escludere tutto ciò che graficamente non è una linea, infatti, se si verificano le condizioni definite in precedenza, sono escluse tutte le proiezioni che generano sequenze continue di ampiezza maggiore.

Il problema delle legature nasce dal momento che esse, talvolta, hanno le dimensioni di una linea e che pertanto possono essere confuse. Si generano, quindi, dei falsi baricentri e dei picchi non validi che si possono collocare in qualsiasi punto del vettore w, perdendo così il significato per cui è stato costruito. Poiché l'obiettivo, come si vedrà più avanti, è quello di identificare una struttura regolare di cinque linee equispaziate, si può in primo luogo escludere tutte quelle configurazioni di w che non presentano tale caratteristica.

Per quanto concerne il problema dell'inclinazione, il metodo di ricerca che si sta prefigurando non ha bisogno di processare l'intera immagine, ma bensì parte di questa. La differenza principale tra questo e il metodo della proiezione-Y applicata all'intero spartito consiste proprio nello svincolarsi dall'influenza di tutto il pentagramma, andando a cercarlo

in zone dimensionalmente più piccole e che pertanto non risentono dell'inclinazione, poiché in ciascuna area processata si può ritenere costante l'andamento in pixel delle linee di pentagramma. Sotto questo aspetto, il criterio di ricerca sviluppato si presenta *robusto*.

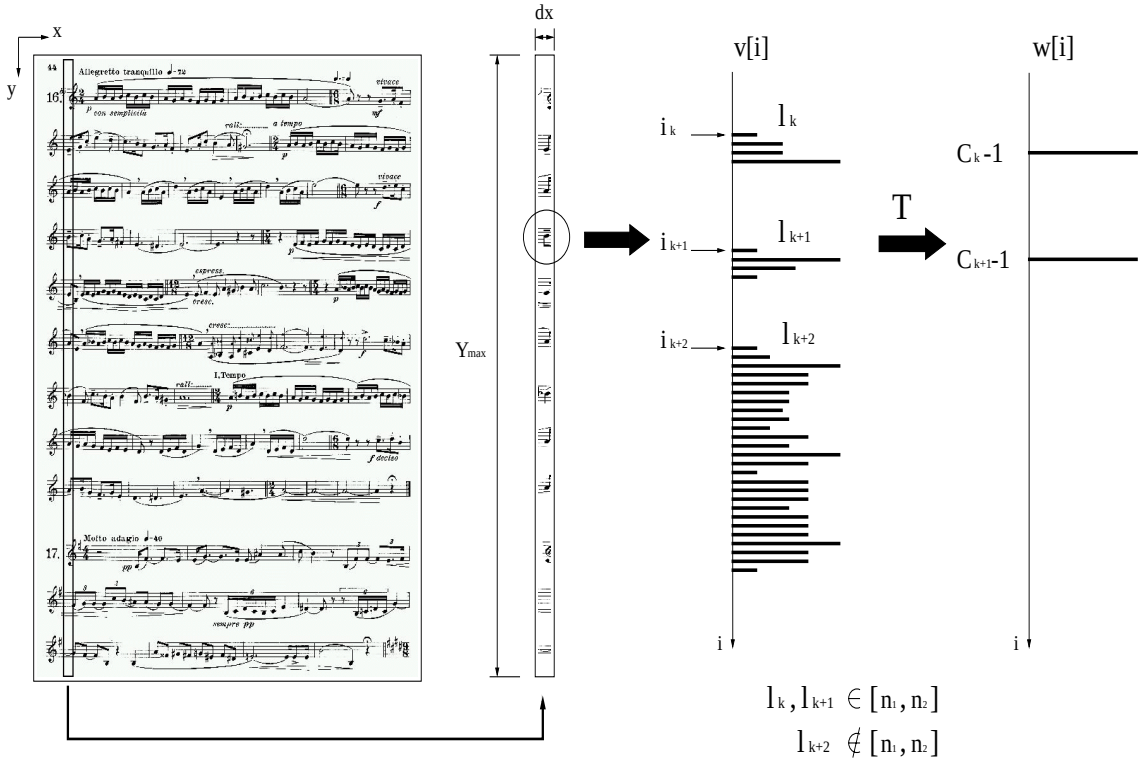


Figura 5.6: Trasformazione della proiezione-Y

5.2.4 Ricerca automatica del pentagramma

Nei precedenti paragrafi sono state introdotte le condizioni che permettono di discriminare e suddividere le proiezioni dei simboli musicali in due insiemi: simboli che graficamente sono linee e simboli dimensionalmente più grandi. È stato anche evidenziato come sia possibile eludere il problema dell'inclinazione se si considerano piccole fette dell'immagine, e quello delle legature, se si trascurano tutte le configurazioni presenti in w che non sono candidate ad essere pentagramma.

La ricerca dei pentagrammi (si veda Figura 5.6) avviene processando delle finestre di immagine di dimensione orizzontale pari a dx_0 e verticale pari a Y_{max}, ottenute da traslazioni lungo la direzione x dello spartito, con la possibilità di definire il passo di scansione ($passo_0$). La scansione orizzontale si arresta quando viene processata un quarto di immagine.

Per mezzo di una seconda finestra sonda (si veda Figura 5.7) con dimensione orizzontale dx_0 e verticale I, non più piccola dell'ampiezza del pentagramma, si scandisce l'immagine estratta. La minima estensione utile I_{min} è facilmente ottenibile dai parametri introdotti andando a considerare i valori massimi ed è espressa dalla seguente relazione:

$$I_{min} = 5n_2 + 4d_2 \tag{5.5}$$

Per cautelarsi e ottenere successivamente la minima centratura utile del pentagramma si considera un valore $I = I_{min} + 20\%$. Quindi, si procede all'analisi del vettore w, limitatamente alla porzione di immagine definita dalla finestra, ricercando cinque baricentri equispaziati, con distanza reciproca entro l'intervallo $[\alpha, \beta]$. La scansione avviene per traslazione a passo unitario finché non si verifica la condizione richiesta.

In caso di esito positivo, viene aggiornata una variabile contatore, preposta al conteggio dei pentagrammi individuati, un vettore delle posizioni, nel quale si memorizza la coordinata y_{sup} della finestra sonda, e si procede al riposizionamento della stessa a partire dalla coordinata y successiva a quella corrispondente al secondo estremo y_{inf}. Si continua fino al raggiungimento della condizione di arresto, coincidente con la copertura completa della *fetta* della sottoimmagine processata. Successivamente, sfruttando l'informazione contenuta nel vettore delle posizioni, si riesaminano le aree (porzioni di w) a partire dalle coordinate memorizzate per centrare il pentagramma all'interno della finestra sonda, verificando, che quello trovato, è effettivamente il rigo musicale[2]. Quindi si riaggiorna la coordinata.

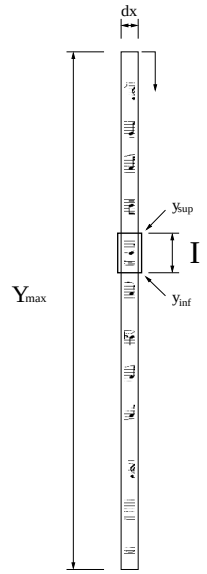


Figura 5.7: Scansione verticale

Nei passi successivi, in presenza di coordinate già acquisite, non si procede all'aggiornamento del vettore delle posizioni. Lo si esegue se la nuova coordinata non rientra entro un intorno di quelle già esistenti, con tolleranza pari all'80%, che equivale a considerare un'area al di sopra e al di sotto del pentagramma pari all'altezza dello stesso. Questa condizione è necessaria per risolvere gli errori di conteggio delle linee, che nascono in presenza di tagli addizionali e in seguito alle modalità di scansione verticale a passo unitario, determinando, in questo modo, la presenza anticipata del pentagramma ed un'errata identificazione della posizione.

Al termine della scansione orizzontale, il vettore delle posizioni fornisce le coordinate verticali $y_{sup}^{(i)}$ delle aree che contengono i pentagrammi. Tali aree, però, definiscono solo la centratura per ciascuno di essi e da sole non sono in grado, generalmente, di contenere

[2]Ad esempio può essere ripetuta localmente la procedura di ricerca

tutta l'informazione musicale, perciò si rende necessaria l'estensione delle coordinate y_{sup} e y_{inf}. Se con n indichiamo il numero dei pentagrammi presenti nello spartito e ($y_{sup}^{(i)}$, $y_{inf}^{(i)}$) la coppia di coordinate della finestra $i-esima$, è possibile ridefinire le nuove coppie di coordinate ($\hat{y}_{sup}^{(i)}$, $\hat{y}_{inf}^{(i)}$) nel modo seguente:

$$\begin{cases} \hat{y}_{sup}^{(i)} = \dfrac{y_{sup}^{(i)} + y_{inf}^{(i-1)}}{2} - \epsilon \\ \\ \hat{y}_{inf}^{(i)} = \dfrac{y_{sup}^{(i+1)} + y_{inf}^{(i)}}{2} + \epsilon \end{cases} \tag{5.6}$$

Con:

- $i = 2, ..., n-1$
- $\epsilon \geq 0$ definito dall'operatore o fissato a priori

La definizione esclude le coordinate del primo e dell'ultimo pentagramma, tuttavia per entrambi è sempre possibile definire, come sopra, sia la coordinata $\hat{y}_{inf}^{(1)}$ che la $\hat{y}_{sup}^{(n)}$, le rimanenti possono essere fatte coincidere con i margini superiore ed inferiore della pagina:

- $\hat{y}_{sup}^{(1)} = 0 \quad$ e $\quad \hat{y}_{inf}^{(1)} = \dfrac{y_{sup}^{(2)} + y_{inf}^{(1)}}{2} + \epsilon$
- $\hat{y}_{sup}^{(n)} = \dfrac{y_{sup}^{(n)} + y_{inf}^{(n-1)}}{2} - \epsilon \quad$ e $\quad \hat{y}_{inf}^{(n)} = Ymax$

L'introduzione di una tolleranza ϵ consente di ottenere aree adiacenti o parzialmente sovrapposte, in questo modo anche inglobando una parte di simboli dei pentagrammi vicini, non si perde informazione, rimandando l'interpretazione degli elementi estranei al contesto corrente alla fase di ricostruzione.

5.3 Livello 1 della segmentazione

Una volta identificato ed estratto il pentagramma, l'obiettivo adesso è isolare ed estrarre delle strisce verticali di immagine che contengano tutti gli elementi musicali e che siano larghe, generalmente, quanto la testa di una nota. Il livello corrente è stato suddiviso in tre stadi (si veda Figura 5.8):

1. Analisi del pentagramma: identificazione di aree con gruppi e figure isolate.
2. Analisi aree con gruppi: identificazione teste delle note.
3. Analisi aree rimanenti: identificazione altri elementi.

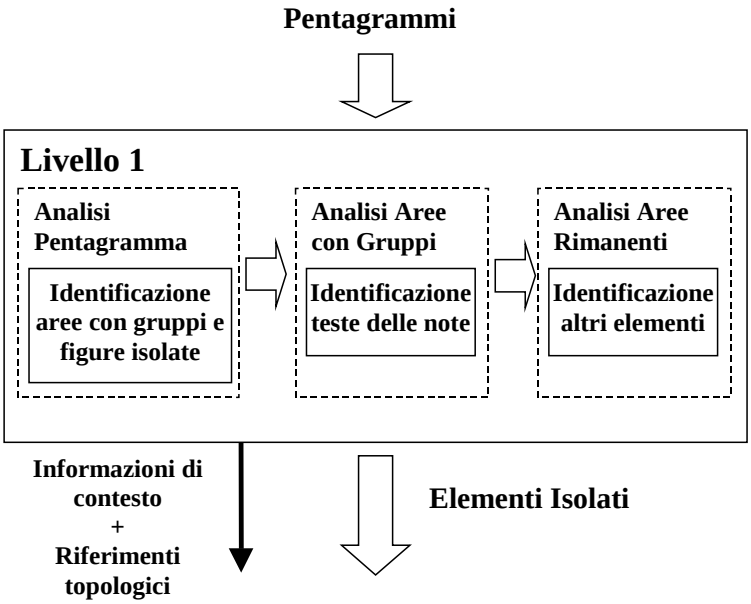


Figura 5.8: Diagramma a blocchi del Livello 1

Il pentagramma in ingresso viene inizialmente processato con l'obiettivo di effettuare un primo isolamento delle configurazioni musicali, gruppi o simboli isolati[3], e determinare un primo insieme di coordinate in corrispondenza delle quali effettuare i tagli per l'estrazione delle immagini che costituiranno l'ingresso delle due fasi successive.

Quest'ultime, generalmente, sono sequenziali anche se talvolta può non essere necessario ricercare altri elementi musicali, se in precedenza si è ottenuto un isolamento totale, permettendo in questo modo di passare direttamente al livello susseguente. Un ruolo rilevante in questo contesto ricopre la complessità del brano e la presenza di configurazioni musicali complesse, sono infatti questi due aspetti che possono richiedere un'ulteriore fase di scomposizione.

5.3.1 Analisi del pentagramma: identificazione di aree con gruppi e figure isolate

È il primo stadio del livello 1 e l'ingresso è costituito dall'immagine del pentagramma. Viene effettuata la ricerca delle aree che denotano la presenza delle sole cinque linee con l'obiettivo di circoscrivere le parti che al loro interno contengono elementi grafici dimensionalmente più grandi di una linea. A tal proposito, è stato utilizzato un'evoluzione del

[3]La distinzione tra gruppi di simboli e simbolo isolato avviene prendendo come riferimento la larghezza della testa di nota, che mediamente risulta essere il doppio dell'altezza. In questi termini si discriminano le immagini identificate confrontando la loro larghezza con quella presa a campione, avviando se necessario la fase di analisi per la ulteriore scomposizione.

metodo di ricerca del livello 0, che nella formulazione originale non è in grado di produrre una scomposizione efficace. Nonostante riesca a identificare le aree nelle quali è presente il solo rigo musicale, in presenza delle legature, posizionate dentro e fuori il pentagramma, non è in grado di trovarlo, in più i tagli addizionali introducono un'errata identificazione, in quanto si riconosce come pentagramma ciò che sta sotto/sopra una testa di nota, non consentendo, quindi, l'isolamento corretto dei simboli.

Variante del metodo di ricerca dei pentagrammi

È stata definita una variante del metodo, nella quale è stata apportata una modifica nella trasformazione e nella costruzione del vettore w. Nella versione precedente è esaminata soltanto la presenza di simboli con ampiezza della proiezione entro l'intervallo stabilito per la linea, in questo caso si estende l'intervallo di variazione della proiezione suddividendo i simboli in due insiemi, uno contenente linee e legature, l'altro tutto ciò che è più grande. La nuova definizione di w diventa:

$$w[i] = \begin{cases} 1 & \text{per} \quad i = C_k - 1 \quad \text{e} \quad n_1 \leq l_k \leq (2n_2 - n_1) \\ -1 & \text{per} \quad i = C_k - 1 \quad \text{e} \quad l_k > (2n_2 - n_1) \\ 0 & \text{altrimenti} \end{cases} \tag{5.7}$$

L'informazione che adesso viene estratta dall'analisi della proiezione è maggiore così come le possibili configurazioni di w e questo ha reso necessario la costruzione di una funzione F che esaminando la struttura di w stabilisca se il pentagramma si trova o meno nella finestra corrente. In altri termini, si tratta di una funzione binaria che indica quando inizia (1) e termina (0) l'area nella quale non ci sono simboli e operativamente fornisce un criterio di segmentazione per isolare gruppi o simboli isolati.

È necessario, anzitutto, associare le configurazioni ai due insiemi introdotti e ricavare le regole che definiscono la funzione. Tali regole possono essere così espresse:

- w contiene una successione con al più cinque 1 equispaziati e nessun -1
- w contiene almeno un -1 o le configurazioni non considerate nel caso precedente

Associando a ciascuna regola un valore binario si può costruire la funzione. La presenza del vettore nullo, che generalmente si genera nei primi passi della scansione del pentagramma, è stata gestita considerandolo come pentagramma[4]. Tale scelta è stata dettata dalla

[4]Il vettore nullo nasce in corrispondenza del margine della pagina contenente lo spartito e può risultare variabile in presenza di indentazioni.

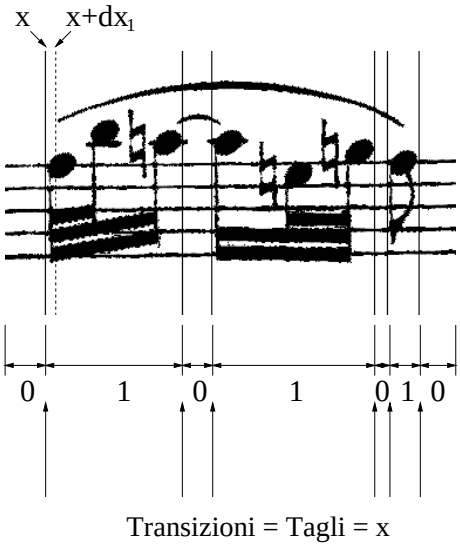


Figura 5.9: Ricerca delle coordinate di taglio per l'isolamento dei gruppi e dei simboli isolati: valori assunti e transizioni della funzione binaria F

necessità di fissare il primo taglio in corrispondenza del primo elemento grafico consistente che si incontra e che generalmente viene a coincidere con la chiave.

L'isolamento

Una volta introdotta una funzione che permette di stabilire, sulla base dell'analisi delle configurazioni di w, la presenza di aree prive di simboli musicali con proiezione maggiore del riferimento scelto, occorre un criterio che consenta di effettuare i *tagli* dentro il pentagramma per isolare l'informazione musicale.

Il processo di scansione del pentagramma avviene con le stesse modalità descritte per la ricerca nell'intero spartito, con la differenza che si scandisce per mezzo di una sola finestra in direzione orizzontale di ampiezza costante pari al valore della massima dimensione verticale dell'immagine estratta al livello 0 ($y_{inf} - y_{sup}$).

La procedura di isolamento è legata al valore di uscita della funzione e può essere descritta nel modo seguente: *in corrispondenza del cambio di valore*[5] *della funzione F si esegue un taglio. La coordinata grafica x in corrispondenza del taglio coincide con quella minore della finestra di scansione e poichè la traslazione della finestra avviene orizzontalmente, le coordinate di taglio ottenute risultano essere ordinate* (si veda Figura 5.9). In questa fase di segmentazione, si ottiene già l'isolamento di quei simboli che per il loro significato musicale sono per definizione isolati, come ad esempio le chiavi, le semibrevi,

[5] *Il procedimento può essere visto come la ricerca dei cambi di stato descritto per mezzo di una variabile binaria il valore della quale è legato al verificarsi di una delle regole citate.*

minime e semiminime con i corrispondenti simboli di pausa, la frazione, le barre di inizio e fine battuta, ecc... Rimangono fuori da questo insieme tutti quelle strutture musicali costituite dall'unione di figure, come ad esempio note unite da travi singole, multiple o miste, e i gruppi irregolari (quintine, settimine ed altri), e quelle in cui l'adiacenza degli elementi grafici è molto ravvicinata (note con alterazioni e armatura di chiave). In quest'ultime, la scelta delle modalità di scansione legate alla scelta dei parametri dx_1 e $passo_1$ ricopre un ruolo importante. Variando il loro valore è possibile controllare finemente il processo in modo da analizzare aree anche piccole, di ampiezza dx, e ottenere in questo modo un isolamento maggiore, in quanto si cerca di far entrare la finestra di scansione negli spazi di adiacenza tra i simboli. Il limite inferiore di tale ampiezza, però, non deve essere troppo piccolo, per non far perdere di significato alla proiezione e considerare l'informazione correttamente.

I gruppi determinati in questa fase, vengono quindi inviati ai due stadi successivi, affinché si esegua la scomposizione definitiva.

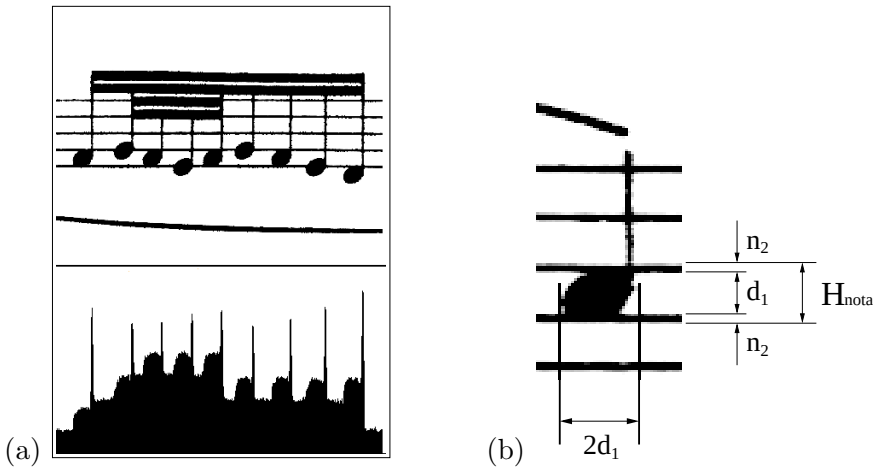


Figura 5.10: (a): Proiezione-X di un gruppo - (b): Parametrizzazione della testa di nota

5.3.2 Analisi aree con gruppi: identificazione teste delle note

Obiettivo principale di questa fase è quello di individuare e determinare le coordinate delle aree nelle quali sono presenti le teste delle note e considerando quanto detto in merito alla distinzione tra gruppi e simboli isolati, la ricerca si riduce a considerare quelle *piene*. Poiché queste sono presenti con maggiore frequenza rispetto ad altri elementi grafici, la loro individuazione determina un notevole risultato in termini di segmentazione al punto che già in questa fase è possibile ottenere una scomposizione completa dell'intero pentagramma e quindi consente di passare direttamente al livello 2. In caso contrario, i

gruppi rimasti integri, che risultano comunque essere in numero ridotto, vengono inviati allo stadio successivo.

La procedura di identificazione delle teste di nota segue la modalità di scansione del metodo di ricerca dello stadio precedente, ma si differenzia per l'introduzione di un'ulteriore livello di analisi necessario a definire una discriminazione corretta tra gli elementi grafici, e perché la ricerca dei tagli avviene sulla proiezione orizzontale debitamente filtrata.

La presenza delle teste (si veda Figura 5.10.a) introduce nella proiezione orizzontale un profilo caratterizzato da un andamento a bassa frequenza. È necessario, pertanto, effettuare un'elaborazione di filtraggio che consenta di eliminare la componente continua introdotta dalla proiezione del pentagramma e le travature, e le componenti ad alta frequenza (picchi), legate alla proiezione degli elementi verticali, tra cui il gambo. Operando su entrambe le proiezioni è possibile combinare l'azione filtrante richiesta. Successivamente si determinano le coordinate di taglio ed in questo caso si sfruttano alcune proprietà grafiche (si veda Figura 5.10.b) che caratterizzano la testa di nota, come ad esempio il rapporto esistente tra altezza e larghezza, che risulta essere pari ad 1/2. Queste due grandezze sono riconducibili a quelle introdotte per descrivere il pentagramma.

Costruzione proiezione-X

L'azione filtrante viene ottenuta analizzando le sequenze non nulle della proiezione-Y con ampiezza maggiore di quella della linea e minore dell'altezza della nota, in questo modo si escludono gli elementi grafici che incidono sulla proiezione verticale determinando la presenza di picchi ad alta frequenza. La sola definizione di un intervallo di variazione per la discriminazione delle sequenze non è sufficiente e si associa, quindi, l'informazione relativa all'incidenza dell'istogramma, ovvero la consistenza del numero e del livello di grigio dei pixel, che costituiscono l'elemento all'interno della proiezione corrente, legata alla sequenza, tra quelle valide, che presenta ampiezza maggiore. Questa informazione viene usata per costruire successivamente la proiezione-X, dalla quale ricavare i tagli per isolare le note.

Il valore massimo dell'altezza della testa di nota, è legato ai parametri che caratterizzano il pentagramma dalla seguente relazione[6]:

$$H_{nota} = 2n_2 + d_1 \tag{5.8}$$

Si considerino le seguenti definizioni:

- Data la proiezione-Y della finestra di scansione e l'insieme H delle sequenze a valori non nulli dell'istogramma ad essa associata, si definisce *sequenza valida* la successione

[6]Si considera un valore maggiorato per tale larghezza, considerando l'ampiezza massima di una linea e quella minima dello spazio.

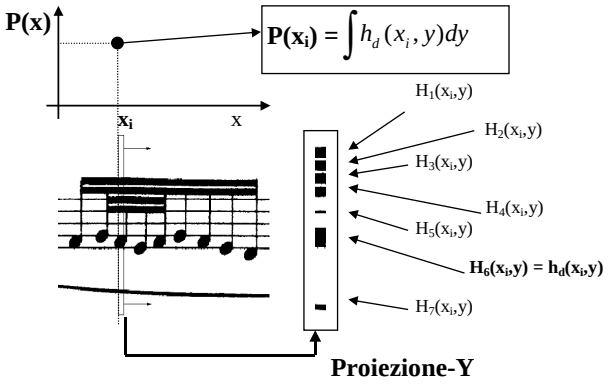


Figura 5.11: Costruzione della proiezione-X

di pixel $h \in H$ tale che:

$$(2n_2 - n_1) < length(h) \leq H_{nota} \tag{5.9}$$

- Sia $H_v \subset H$ l'insieme delle sequenze valide, si definisce *sequenza dominante* la sequenza $h_d \in H_v$ di lunghezza massima.
- Data una sequenza dominante si definisce *incidenza* P della sequenza la somma dei valori contenuti in essa.

La proiezione verticale passo passo, parallelamente alla scansione dell'immagine, viene costruita considerando in successione le *incidenze* P per ciascuna *sequenza dominante* ottenuta ad ogni traslazione della finestra di elaborazione (si veda Figura 5.11). Poiché la scansione avviene con sovrapposizione delle finestre e traslazione a passo unitario, l'effetto globale è quello di introdurre un'azione filtrante di tipo passa basso (si veda Figura 5.12).

Calcolo dei baricentri

Ottenuta la proiezione-X, si procede all'isolamento dei profili utili in corrispondenza delle teste di note, mediante un'operazione di soglia dei valori della proiezione e delle ampiezze dei profili, quest'ultimo con l'obiettivo di eliminare gli eventuali picchi residui di breve durata spaziale. Per quanto concerne il primo valore di soglia si considera la condizione generata dalla proiezione di una testa di nota quando la finestra di scansione si colloca nel punto di massima incidenza in corrispondenza della sequenza dominante maggiore, all'incirca in un intorno del punto di mezzo della testa. Il valore di massima incidenza P_{max} si può facilmente ottenere considerando la porzione di finestra che racchiude interamente la testa di nota con il massimo valore per i pixel, in termini di livelli di grigio.

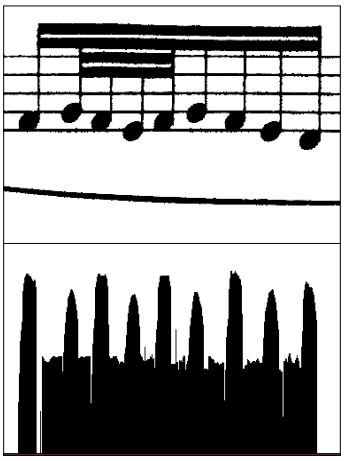


Figura 5.12: Proiezione-X intermedia

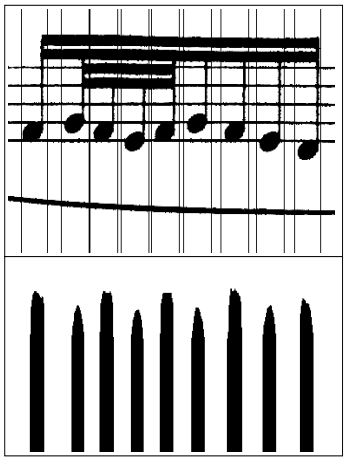


Figura 5.13: Proiezione-X elaborata e isolamento delle teste di nota

Definizione: Data l'ampiezza della finestra di scansione dx e l'altezza della testa di nota (piena) H_{nota}, la *massima incidenza* P_{max} è data da:

$$P_{max} = 255\ dx\ H_{nota} \tag{5.10}$$

Il primo valore di soglia viene fissato al 90%, mentre per quanto riguarda il secondo, quello relativa all'ampiezza dei profili, viene fissato nel valore di $d_1/2$.

Determinata la nuova proiezione-X (si veda Figura 5.13), si procede con il calcolo della coordinata del baricentro delle sequenze di istogramma in corrispondenza dei profili, successivamente si calcolano i punti di taglio. Indicata con C_x la coordinata del baricentro lungo la direzione x dell'immagine, la coppia dove effettuare il taglio è data da:

$$(C_x - d_1, C_x + d_1) \tag{5.11}$$

In corrispondenza dei baricentri periferici nell'immagine del gruppo, è sufficiente definire solo le coordinate interne per i tagli, in quanto le esterne coincidono con quelle individuate nella fase di definizione dello stesso. Per ogni gruppo esaminato si procede, quindi, all'aggiornamento della lista delle coordinate di taglio mantenendo la struttura ordinata.

5.3.3 Analisi aree rimanenti: identificazione altri elementi

È la fase finale del livello 1 ed è preposta a risolvere quelle situazioni di raggruppamento che non sono state risolte nei passi precedenti. La ricerca non si basa su particolari simboli, ma si propone come un metodo di carattere generale.

Definiti i tagli dello stadio precedente, si effettua la ricerca di eventuali gruppi con estensione grafica maggiore della larghezza della testa di nota, presa anche in questo caso

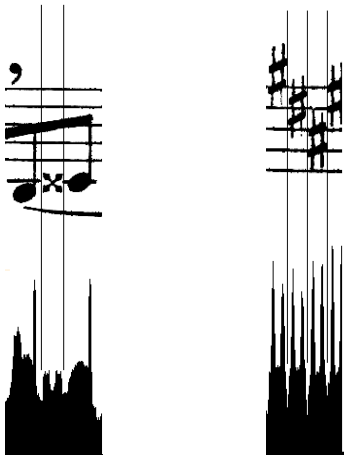


Figura 5.14: Corrispondenza tra i tagli ricercati e i minimi della proiezione-X

come riferimento, se non esistono configurazioni da scomporre si procede con il livello successivo. Il valore di confronto viene fissato a $(5/2)d_1$ e dall'insieme ordinato che si viene a definire, si esclude il primo gruppo che generalmente racchiude la chiave. I gruppi in ingresso (si veda Figura 5.14) a questo stadio, sono caratterizzati da figure ravvicinate tra loro e tali che la finestra di scansione non è in grado di rilevare l'interspazio, che esiste comunque, anche se minimo. La presenza delle aree di separazione tra simboli adiacenti, produce nel profilo della proiezione-X dei minimi relativi, poiché il peso grafico degli elementi grafici si riduce, venendo a mancare un contributo alla proiezione. Da questa osservazione, si intuisce la necessità di considerare la sola proiezione-X per ricercare le coordinate dei tagli necessari alla separazione definitiva dei simboli e che quindi l'obiettivo è quello di determinare i minimi corrispondenti all'interspazio.

Ricerca dei minimi

Data la proiezione-X del gruppo (si veda Figura 5.15) (a, b), se ne considera una regione con estremi distanti d_1 dai margini e si opera un filtraggio di tipo passo basso, mediante un filtro a media mobile su tre punti e passo unitario, per "addolcire" il profilo della proiezione.

Si prosegue con la ricerca del minimo assoluto che genera due nuove regioni I_1 e I_2 e in corrispondenza del quale si definisce un taglio. Successivamente, si esaminano le ampiezze delle porzioni ottenute: se entrambe o una delle due risultano avere un'estensione maggiore di $d_1/2$, si riapplica la procedura. Si crea così un procedimento ricorsivo, che termina quando le aree non rispettano le condizioni fissate sull'ampiezza, e in corrispondenza dei minimi assoluti locali, ottenuti prima dell'arresto del procedimento, si definisce

la coordinata di taglio. In presenza di un profilo della proiezione costante, ad esempio in corrispondenza di tratti di pentagrammi all'interno dei quali non sono presenti simboli musicali, il risultato della ricerca dei minimi, produce una scomposizione dell'immagine in porzioni di larghezza poco più grande di quella della testa di nota.

Il criterio usato, si presta a risolvere i problemi connessi alla presenza dell'armatura di chiave nella quale sono presenti i simboli di alterazione in configurazione ravvicinata e le alterazioni temporanee nella linea melodica.

La generalità del criterio, infine, permette di risolvere le scomposizioni fallite allo stadio precedente, nonostante possano essere presenti delle teste di nota, i minimi si posizionano nelle regioni in cui esse non sono presenti, consentendo così di determinare i tagli mancanti.

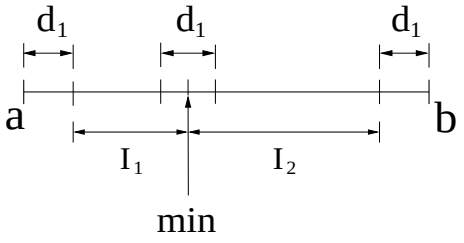


Figura 5.15: Procedimento di ricerca dei minimi

5.3.4 Identificazione di informazioni relative al contesto

La procedura illustrata non consente soltanto di ottenere i segmenti di immagine da inviare al livello 2, ma permette di associare a ciascuna di esse informazioni relative al contesto che risultano utili sia nel livello 2 per guidare la scelta della modalità di segmentazione finale, sia in sede di riconoscimento per agevolare il recupero dell'errore di classificazione da parte della rete neurale, ed, infine, nella fase di ricostruzione delle figure musicali:

- L'algoritmo di ricerca dei gruppi e delle figure isolate permette di etichettare e discriminare i segmenti di immagine che fanno parte di un gruppo o sono isolati.

- Per mezzo dell'algoritmo di ricerca delle teste di nota piene, i segmenti contenenti la nota sono marcati in modo da fissare l'informazione legata alla presenza della testa di nota (piena) all'interno di essi.

- Combinando le informazioni sopra descritte si stabiliscono i segmenti contenenti la nota d'inizio e quella di fine appartenenti ad un gruppo di note (note connesse orizzontalmente mediante travi o beam); tale informazione permette di tenere traccia delle configurazioni e delle relazioni orizzontali tra le note durante la fase di ricostruzione.

Infine, poiché il pentagramma viene esaminato completamente, si esegue il tracciamento continuo delle coordinate verticali delle linee. La conoscenza dell'andamento del pentagramma consente di risolvere il problema dell'inclinazione, di poter usufruire di un riferimento continuamente aggiornato durante la fase di ricostruzione e di conoscere la fine del pentagramma.

5.3.5 Selezione delle aree di estrazione e considerazioni

In questa sezione si analizzano le problematiche introdotte dagli stadi del livello 1. La non perfetta rappresentazione grafica degli elementi grafici dovuta al rumore che si sovrappone con l'acquisizione e alle imprecisioni tipografiche (si veda paragrafo 1.6), influenzano il processo di ricerca del pentagramma che può generare dei tagli indesiderati. Può succedere, ad esempio, che in una zona dove è presente il solo pentagramma, una delle cinque linee sia troppo ampia, questa situazione comporta l'identificazione di un simbolo, determinando il cambio di valore della funzione binaria e l'introduzione del taglio.

Per quanto riguarda i due stadi successivi, entrambi possono generare dei tagli ravvicinati, ma differentemente dal problema precedente, questi sono insiti nel processo di identificazione che essi operano.

In definitiva, pur ottenendo un insieme di segmenti di immagini utili, si aggiungono alcune aree con uno spessore troppo piccolo. In virtù di quanto detto, prima di passare al livello 2 si esegue una selezione delle zone di estrazione, escludendo quelle con larghezza, lungo la direzione x, che risultano essere minori di una certa soglia, valutata in pixel[7]. Tale operazione è necessaria in prospettiva del processo di *classificazione*, in quanto si deve operare una normalizzazione delle dimensioni grafiche di ciascun elemento di base estratto, che costituisce l'ingresso della rete neurale.

La struttura ordinata della lista delle coordinate di taglio facilita la selezione delle aree di interesse, infatti è sufficiente scandire la lista a passo unitario e confrontare la differenza tra la coordinata $i - esima$ e la precedente con il valore di riferimento scelto.

Il livello 1 consente di etichettare sia i segmenti di immagine all'interno dei quali è stata riscontrata la presenza di una testa di nota nera, sia quelli che appartengono ad aree di immagini etichettate come gruppo di note. Prendendo in esame le regole di scrittura musicale, all'interno di un gruppo di note sono presenti tutte le figure di nota con valore inferiore all'ottavo; tali figure hanno tutte la testa nera. Nel caso però di note non raggruppate, esse non vengono rilevate dal livello 1, in quanto operativamente la ricerca avviene solo in aree identificate come gruppi. Pertanto occorre recuperare tale informazione, di questo verrà discusso in seguito come fase di elaborazione all'interno Livello 2. Tale fase permetterà di etichettare il segmento come contenente una testa di

[7]Come valore di riferimento può essere presa la dimensione del simbolo musicale più piccolo: il punto di valore.

nota nera. Nessuna ricerca specifica è stata prevista a questo livello per le note come le minime e la semibreve che sono state considerate alla stessa stregua di un simbolo notazionale generico e che per la loro natura grafica sono da considerarsi isolate. Lo sforzo nella ricerca mirata delle note con la testa nera è giustificato dal fatto che esse sono le figure che ricorrono con più frequenza negli spartiti di musica classica e che presentano una vasta tipologia di rappresentazione.

5.4 Un esempio: dal livello 0 al livello 1

In questa sezione, sono riportati a titolo di esempio gli effetti delle elaborazioni introdotte dai livelli 0 e 1. È stata presa in considerazione una pagina di musica con linea melodica monofonica ad una voce, digitalizzata a 300 d.p.i. e 256 tonalità di grigio.

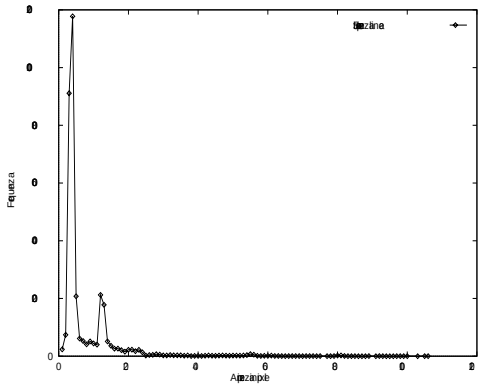
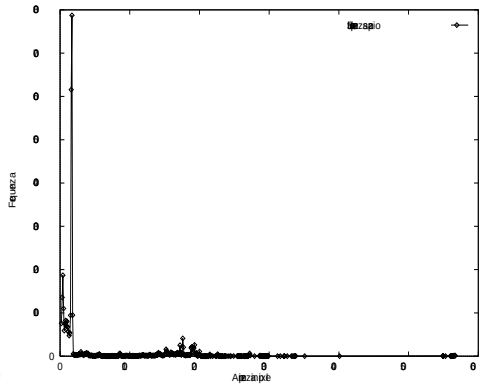
(a)  (b) 

Figura 5.16: Stima dell'ampiezza della linea (a) e dello spazio (b)

La Figura 5.16 fornisce una stima[8] dei parametri di inizializziazione del processo di segmentazione, in corrispondenza dei due massimi assoluti si determinano rispettivamente il valore dell'ampiezza della linea e dello spazio. Nel caso specifico dai valori riportati nella Tabella 5.1 si determina:

- ampiezza linea=4
- ampiezza spazio=17

Per fissare la tolleranza sull'ampiezza della linea è sufficiente osservare la presenza di valori di frequenza non trascurabili nell'intorno di quella massima e questo permette di fissare una variazione di ± 1 rispetto al valore massimo. Nel caso dello spazio si possono considerare il primo e il secondo massimo che risultano essere contigui. Con questa considerazione si ottengono i parametri che inizializzano il processo di segmentazione:

- $n_1 = 3$ e $n_2 = 5$ per l'ampiezza di linea

[8]Nel caso specifico la stima è stata eseguita considerando il primo quarto di pagina.

Ampiezza	Frequenza
1	236
2	740
3	9111
4	**11777**
5	2076
6	612
7	526
8	410
9	513
10	443
11	400
12	2124
13	1779
14	511
15	362
16	260
17	262
18	211
19	155
20	220
21	227
22	173
23	221
24	136

(a)

Ampiezza	Frequenza
2	760
3	1358
4	1869
5	1102
6	590
7	749
8	824
9	693
10	800
11	677
12	562
13	476
14	539
15	944
16	**6157**
17	**7874**
18	949
19	54
20	26
21	32
22	31
23	28
24	28
25	27

(b)

Tabella 5.1: Valori assunti nell'intorno della frequenza massima per le sequenze di pixel neri (a) e pixel bianchi (b).

- $d_1 = 16$ e $d_2 = 17$ per l'ampiezza dello spazio

La Figura 5.17 mette in evidenza l'uscita del livello 0 con l'identificazione e circoscrizione dei singoli pentagrammi. I parametri di scansione di questa fase sono stati fissati nel seguente modo:

- apertura finestra $dx_0 = 4$
- passo di scansione $passo_0 = 2$
- tolleranza di taglio $\epsilon = 0$

Dell'insieme dei pentagrammi che si vengono a determinare, si riporta, la segmentazione del rigo evidenziato dalle frecce, mettendo in evidenza gli effetti dell'elaborazione effettuata dagli stadi che compongono il livello 1, usando per la scansione orizzontale i seguenti valori:

- apertura finestra $dx_1 = 3$

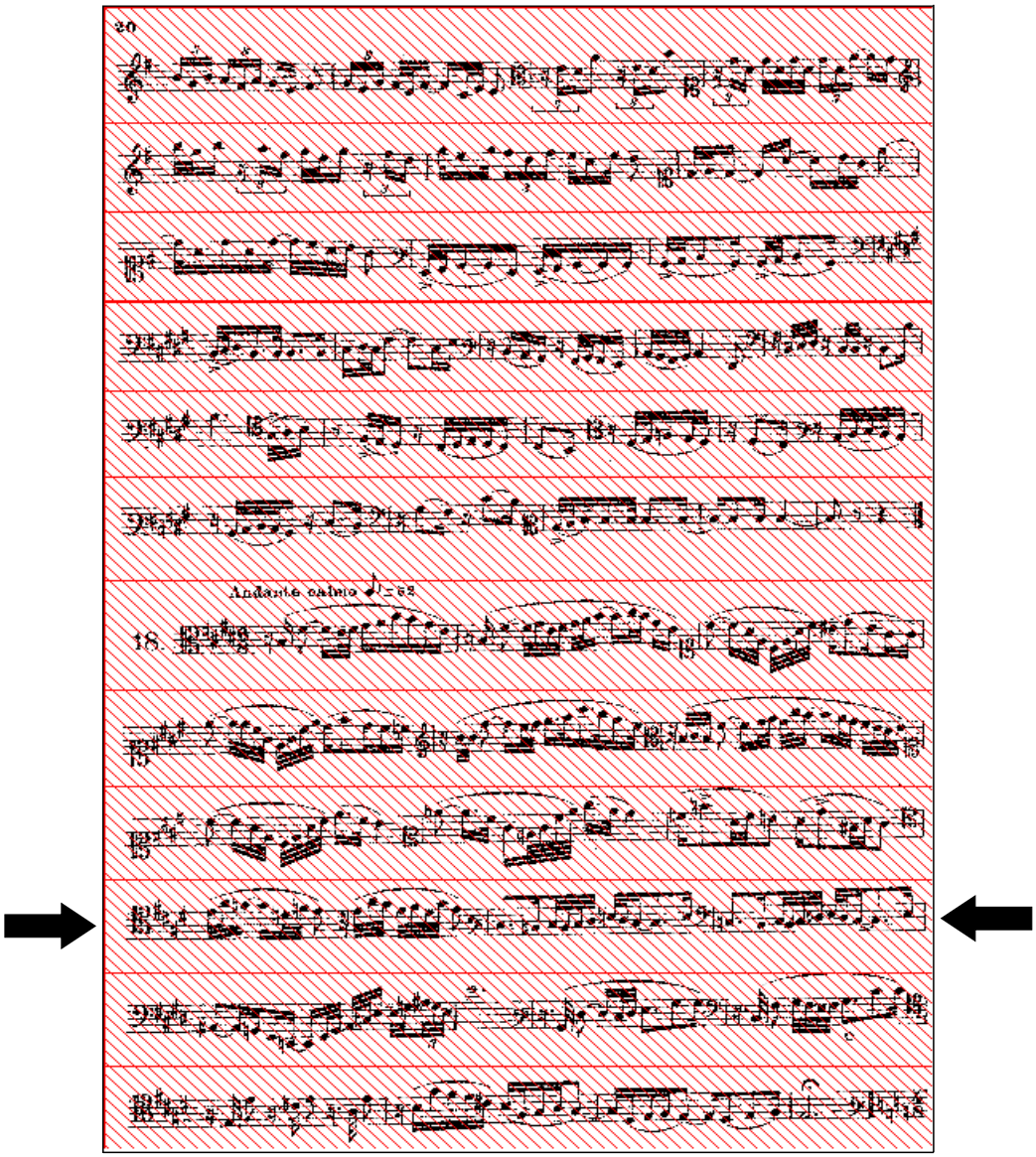


Figura 5.17: Uscita livello 0: identificazione e circoscrizione dei pentagrammi

- passo di scansione $passo_1 = 1$

La Figura 5.18 riporta in ordine di tempo i passi legati alla scomposizione che produce in uscite le fette che verranno mandate in ingresso al livello 2 per la suddivisione finale negli elementi di base.

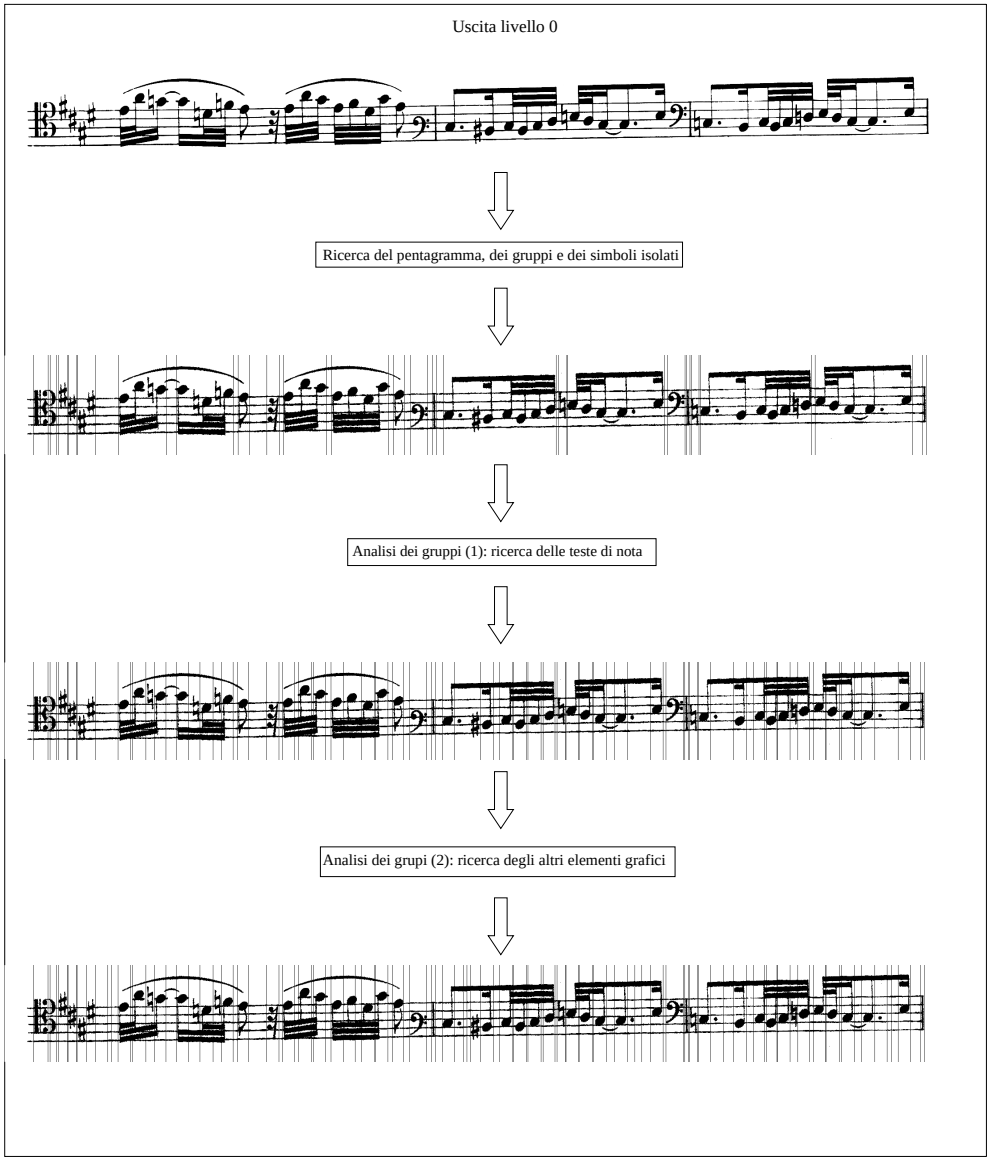


Figura 5.18: Procedura di segmentazione del livello 1

5.5 Livello 2 della segmentazione

Il Livello 2 è l'ultima fase del processo di segmentazione. I segmenti di immagine provenienti dal Livello 1 sono definitivamente decomposti in simboli di base. Questa fase ricopre un ruolo importante all'interno di tutto il processo di segmentazione in quanto da essa dipendono in modo diretto sia la fase di riconoscimento che di ricostruzione. Per questa ragione, i segmenti di immagine che si ottengono al termine della decomposizione devono avere le seguenti proprietà:

- appartenere all'insieme dei simboli di base definiti dal processo di segmentazione;
- essere ottenuti in modo ripetibile e con proprietà grafiche il più uniformi possibile;
- avere associate informazioni necessarie per il riconoscimento.

I primi due aspetti sono fondamentali per l'affidabilità del processo di riconoscimento, mentre l'ultimo è importante nell'applicazione delle regole per la ricostruzione dei simboli musicali. Le informazioni ottenute dalle precedenti fasi di segmentazione consentono di suddividere i segmenti di immagine in due insiemi, questa ripartizione ha condotto alla definizione di due procedure di decomposizione da applicare rispettivamente alle immagini contenti le note e alle altre immagini. Entrambe le procedure sono basate sulla tecnica delle proiezioni X/Y e generano coppie di coordinate verticali che permettono di estrarre i simboli di base.

5.5.1 Ricerca delle note con testa nera in immagini etichettate come isolate

Nella sezione 5.3.5 sono stati discussi i limiti della ricerca delle teste di nota operata nel Livello 1 e sottolineata l'importanza di recuperare l'informazione sulla presenza della testa di nota nera all'interno di un segmento che è stato etichettato come isolato e che potenzialmente potrebbe contenere una figura di nota con tali caratteristiche. Le uniche informazioni a disposizione in questa fase sono solo i riferimenti delle coordinate e le dimensioni del bounding box, associate all'etichetta di figura isolata. Non tutti i segmenti devono essere considerati, ma solo quelli che presentano delle caratteristiche grafiche che denotano la possibilità di contenere una figura di nota, per ridurre lo spazio di analisi e non appesantire il processo di elaborazione. Ricordando che le fasi dei livelli precedenti sono state progettate per estrarre segmenti che abbiamo un'ampiezza comparabile a quella della testa della nota ($2d_1$), tutti i segmenti isolati che presentano questa caratteristica topologica saranno oggetto di analisi per la ricerca della presenza della testa nera. La modalità di ricerca usata è il metodo utilizzato (si veda il paragrafo 5.3.2) per le aree etichettate come gruppi. Individuata la presenza della testa il segmento viene quindi

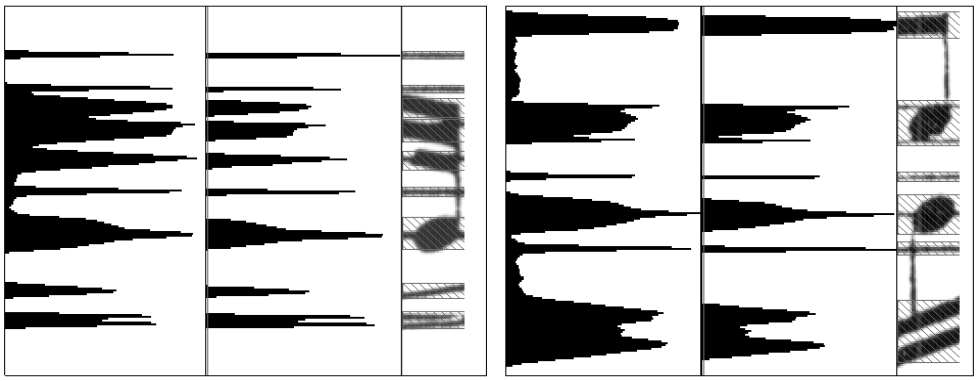


Figura 5.19: Le proiezioni sulla sinistra delle figure sono quelle originali; al centro sono riportate le proiezioni elaborate con un filtraggio passa alto.

etichettato come contenente la testa di nota e successivamente decomposto in simboli di base.

5.5.2 Decomposizione immagini contenenti le teste di nota nere

Le componenti grafiche di una nota sono principalmente due: (i) la testa e (ii) le travi (in caso di gruppi di note) oppure gli uncini (nel caso di nota singola/isolata). Queste componenti sono connesse tra loro verticalmente attraverso il gambo. Gli abbellimenti (mordenti, gruppetti, trilli, accenti, ecc.) e i simboli orizzontali (legature, crescendo, ecc.), che possono essere associati alla nota (si veda Capitolo 3), non sono graficamente connesse alla nota e sono allineati verticalmente rispetto ad essa. Pertanto l'estrazione della testa, delle travi e degli uncini è realizzabile disconnettendoli dal gambo. Osservando la proiezione-Y ottenuta dalle note (si veda Figura 5.19), è possibile identificare il contributo imputabile al gambo. Esso aggiunge un contributo continuo (offset) al profilo della proiezione che si estende dalla testa della nota alle travi o uncini. Il contributo del gambo, quindi, può essere preso come riferimento per fissare un'eventuale soglia di taglio da applicare all'estrazione dei simboli di base operando direttamente col profilo della proiezione-Y. Con riferimento alla Figura 5.20, si osserva però una certa difficoltà nel definire un valore da attribuire alla soglia che abbia una valenza generica, poiché si potrebbero avere estrazioni non corrette e generazione di simboli di base diversi. Difatti, mentre per il primo caso il valore di soglia usato consente di isolare la testa della nota e l'insieme delle tre travi, nel secondo tale valore è troppo basso e si ottiene l'intera nota (testa e gambo) che diventerebbe elemento primitivo di base; e così in molti altri casi. Un'impostazione di questo tipo renderebbe l'insieme dei simboli di base molto grande e composto da elementi simili, ma non uguali (note con gambo più o meno lungo).

È molto importante, dunque, che l'insieme dei simboli di base abbia componenti elementari, semplici e molto comuni: questo assicura una corretta scomposizione della

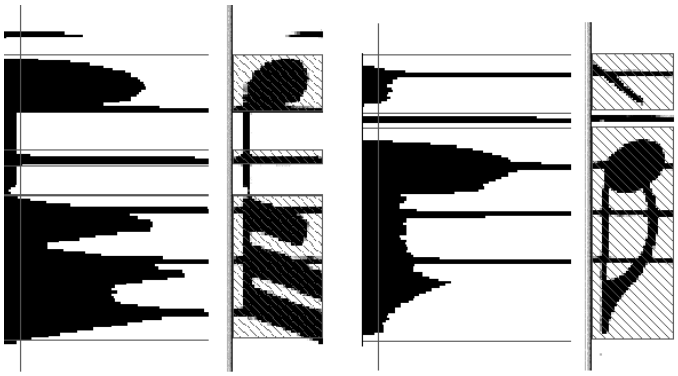


Figura 5.20: Esempio di segmentazione a soglia costante con generazione di simboli di base differenti

simbologia e un numero abbastanza limitato di elementi. Per ottenere una segmentazione che prescinda dal gambo si fa uso di un filtraggio passa alto che elimini tutti i livelli costanti del profilo. Nella Figura 5.19 sono riportati due esempi che evidenziano la semplificazione ottenuta con un filtraggio di tipo *unsharp gaussiano* con finestra temporale di 101 pixel. Sulla base di queste considerazioni la procedura di segmentazione è costituita dai seguenti passi:

1. Calcolo della proiezione-Y del segmento di immagine contenente almeno una testa di nota nera.

2. Filtraggio passa-alto della proiezione per mezzo di un filtro *unsharp gaussiano* con finestra temporale di 101 pixel per l'eliminazione del gambo.

3. Identificazione delle coordinate dei tagli per l'estrazione dei segmenti contenenti i simboli di base. Le coordinate sono individuate sul profilo della proiezione filtrata, mediante il meccanismo di estrazione con soglia. Sono utilizzate due particolari, una per le note all'interno di un gruppo e l'altra per le note isolate.

4. Estrazione segmenti e aggiustamento aeree estratte: quando la distanza di due segmenti successivi è inferiore allo spessore massimo della linea di pentagramma (n_2), essi sono fusi in un unico segmento.

I valori delle soglie per l'estrazione dei segmenti relativi ai simboli di base sono stati espressi in funzioni dello spessore delle linee del pentagramma come segue:

$$soglia = \begin{cases} \frac{3}{4}\frac{(n_1+n_2)B}{MaxPy}, & \text{Note in gruppo} \\ \\ \frac{n_2 B}{MaxPy}, & \text{Note isolate} \end{cases} \tag{5.12}$$

dove:

- **B** è il valore massimo attribuito al pixel nero e posto pari a 255.
- **MaxPy** è il valore massimo della proiezione-Y relativa al segmento di immagine.

La necessità di distinguere due soglie è legata alla diversa conformazione delle travi e degli uncini e alla dimensione che si desidera ottenere per i simboli di base che essi dovranno rappresentare.

5.5.3 Decomposizione delle immagini isolate e contenenti altri simboli musicali

I segmenti di immagine oggetto della decomposizione sono tutti i simboli indipendenti e quelli che pur facendo parte di un gruppo di note non contengono al loro interno una nota. Le figure musicali gestite in questa fase sono prevalentemente le pause, le chiavi, le alterazioni, i punti di valore, le porzioni di travi interne a gruppi di note, porzioni di pentagramma, porzioni di legature, le frazioni, le barre, le note semibreve e minima. Tutti questi simboli ad eccezione della nota da 1/2 (minima) sono estratti direttamente valutando la proiezione-Y dell'immagine e operando un'estrazione mediante un valore di soglia fissato all'1% del valore massimo della proiezione. Dall'insieme dei segmenti ottenuti vengono esclusi tutti quelli con altezza al più uguale a quello dello spessore della linea di pentagramma e centrati su una delle cinque coordinate relative alle linee del pentagramma. La ragione della scelta di un valore di soglia basso è legata al fatto che le figure oggetto della decomposizione per loro natura non sono connesse ad altri simboli e quindi tutti i contributi alla proiezione sono dovuti alle singole figure comprese le linee di pentagramma non sovrapposte a simboli grafici.

Nel caso in cui il segmento contenga una figura di minima, tale valore di soglia risulterebbe troppo basso. Infatti, la presenza del gambo connesso alla nota, ripropone il problema della scelta di un valore della soglia che consenta di escludere lo scalino che esso introduce nella proiezione-Y e di poter estrarre la testa della nota come simbolo di base. In questo caso, occorre un valore di soglia più alto e un metodo per valutarlo.

Identificazione della note da 1/2 (minima)

Il punto di partenza, per la risoluzione del problema della determinazione della soglia nel caso della minima, è la necessità di individuarne la presenza all'interno del segmento di immagine. Il metodo sviluppato prevede di analizzare tutti e solo quei segmenti etichettati come isolati e la cui ampiezza è comparabile con quella di una testa di nota. Solo nel caso in cui il metodo di ricerca fornisca un esito positivo allora per tale segmento viene valutato il valore di soglia da applicare alla proiezione-Y. Con riferimento alla Figura 5.21,

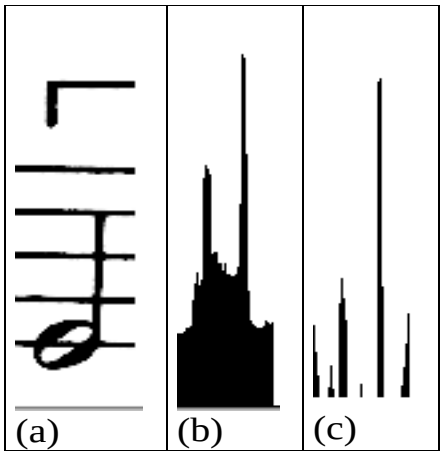


Figura 5.21: Segmento contenente una minima (a), proiezione-X (b), proiezione-X ottenuta con un filtro passa-alto

si osserva che la presenza della nota può essere stabilita individuando il valore massimo della proiezione-X (si veda Figura 5.21.b): tale valore è generato dal gambo. Il picco può trovarsi a destra rispetto al centro del segmento se la nota ha il gambo rivolto verso l'alto oppure a sinistra se è rivolto verso il basso. La ricerca del picco e della sua ampiezza consentono di fissare la soglia da applicare nella proiezione-Y per oltrepassare il gradino continuo dovuto proprio al gambo. Per effettuare l'estrazione si osserva che a seguito di un filtraggio passa-alto della proiezione-X (si veda Figura 5.21.c), il contributo del gambo è ancora presente ed è dominante; questo consente una facile estrazione e una valutazione della sua ampiezza (A_{max}). Il valore della soglia da applicare successivamente sul profilo della proiezione-Y è definito come segue:

$$soglia \quad = \quad \frac{B \max(n_2,\ 1.2\ A_{max})}{MaxPy} \tag{5.13}$$

dove:

- **B** è il valore massimo attribuito al pixel nero e posto pari a 255.
- **MaxPy** è il valore massimo della proiezione-Y relativa al segmento di immagine.

5.6 Uscita del livello 2

Le procedure descritte in precedenza hanno lo scopo di decomporre l'immagine di partenza in un insieme di segmenti di immagini contenenti i simboli di base. Tale insieme costituisce l'uscita del segmentatore; il passaggio da immagini a significato dell'immagine

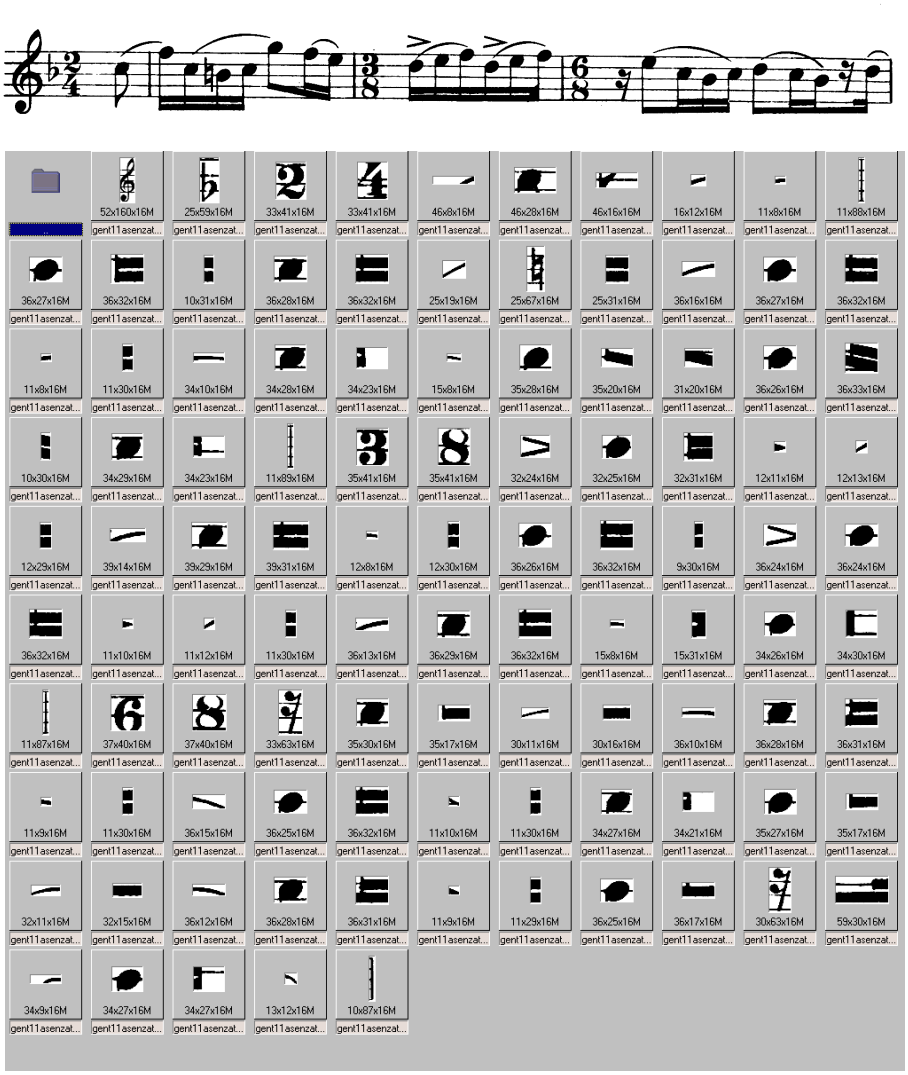


Figura 5.22: Decomposizione finale in simboli di base

sarà l'obiettivo del modulo di classificazione. Nella Figura 5.22, è riportato un pentagramma e la sua decomposizione in simboli di base, mentre nella Figura 5.23 è evidenziata una porzione dello stesso pentagramma nella quale sono state messe in evidenza i segmenti verticali risultato della fase relativa al Livello 1 e il relativo insieme dei simboli estratti. La porzione di immagine riportata negli esempi è stata estratta da uno spartito monofonico acquisito con risoluzione di 300 d.p.i. e in 256 tonalità di grigio. I simboli di base ottenuti costituiscono a questo livello ancora un'informazione grafica. Il modulo relativo alla classificazione, una volta attribuito un valore semantico e le informazioni ricavate nella fase di segmentazione, permetterà di passare nel dominio della semantica e di definire la struttura dell'ingresso per la fase finale di assemblaggio e ricostruzione delle figure musicali.

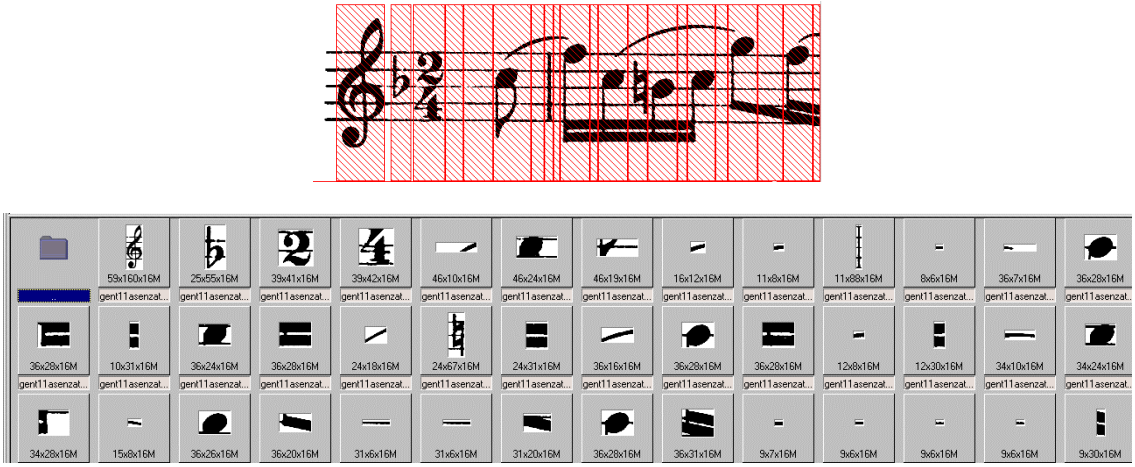


Figura 5.23: Particolare di pentagramma: segmenti verticali e relativi simboli di base

5.7 Alcune considerazioni sulla risoluzione di acquisizione

Tutte le prove di verifica del processo di segmentazione sono state realizzate con spartiti musicali acquisiti con una risoluzione di 300 d.p.i. Di seguito è stato riportato un esempio di valori che i parametri hanno assunto alla risoluzione di acquisizione usata:

1. $n_1 = 3$ e $n_2 = 5$
2. $d_1 = 16$ e $d_2 = 17$
3. $dx_0 = 4$ e $dx_1 = 3$

I valori delle coppie di tolleranza sono strettamente dipendenti dalla risoluzione di scansione e si può pensare che verosimilmente essi assumano valori differenti in corrispondenza di altre risoluzioni, in particolare passando a 600 d.p.i. raddoppino.

Non si può dire altrettanto per le ampiezza delle finestre di scansione dx_0 e dx_1, poiché sono già state scelte cercando il giusto compromesso tra la finezza della scansione e la coerenza dell'informazione prodotta dalla proiezione-Y. Pertanto, in caso di diminuizione della risoluzione queste ed in particolare dx_1 non sarebbero più valide o quantomeno non garantirebbero quella finezza necessaria, poichè gli interspazi tra i simboli grafici si riducono. Si può ipotizzare che un aumento della risoluzione comporti dei benefici, in questo caso si avrebbe l'effetto contrario e conseguentemente si otterrebbe un'ulteriore grado di finezza pur fissando gli stessi valori, tuttavia il tempo di elaborazione aumenterebbe con l'aumentare della risoluzione grafica (aumentano le dimensioni delle immagini). Si può ritenere che la risoluzione di 300 d.p.i sia il giusto valore di compromesso tra la finezza e il tempo di elaborazione.

Capitolo 6

Architettura del modulo di riconoscimento

In questo capitolo è affrontata la fase di riconoscimento. In questa fase gli ingressi al riconoscitore sono costituiti dalle immagini dei simboli di base e dalle informazioni ad esse associate dal processo di segmentazione. È in questa fase che avviene il passaggio da simbolo grafico ad informazione primitiva o più semplicemente primitiva. Formalmente si realizza il processo di assegnazione di un *individuo*, descritto da un insieme di parametri (*feature*), ad una *famiglia* (classe) attraverso una regola di decisione. Alla base della classificazione si trova l'operazione di apprendimento (*learning*) eseguita su un insieme di dati *annotati*, dei quali si conosce l'attribuzione e che consente di definire la funzione di *mapping* con la quale effettuare l'associazione dei dati da classificare. La funzione di mapping o classificatore è stata realizzata mediante una rete neurale, in particolare è stata usata una rete *Multi Layer Perceptons Backpropagation*.

6.1 L'architettura del riconoscitore di simboli

L'architettura del riconoscitore di simboli è costituita da due componenti: il modulo di classificazione delle immagini e un modulo per il recupero degli eventuali errori di riconoscimento. Il primo è stato implementato per mezzo di una rete neurale, mentre il secondo si avvale di un insieme di euristiche definite sulla base delle informazioni di contesto e topologiche provenienti dal segmentatore. Gli ingressi sono costituiti dalle immagini dei simboli di base prodotte dalla segmentazione e da una serie di informazioni relative al contesto in cui esse si trovano e la loro topologia. Per quanto concerne le informazioni di contesto, esse stabiliscono se:

- all'interno del segmento verticale dell'immagine cui fa parte il simbolo di base da riconoscere è segnalata la presenza della testa di nota (nera)

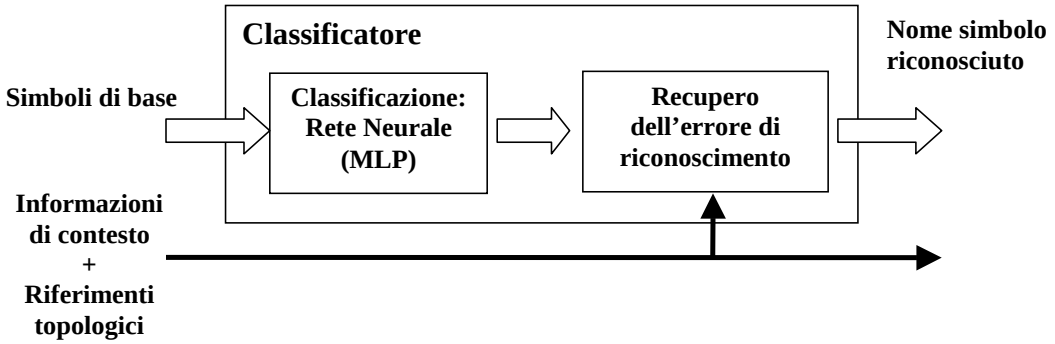


Figura 6.1: Struttura del modulo di riconoscimento

- il simbolo di base è contenuto in un segmento verticale appartenente ad un'area individuata come gruppo di note. In questo caso, sono possibili tre tipi di informazione: (i) il segmento rappresenta l'inizio di un gruppo di note, (ii) il segmento si trova all'interno, (iii) il segmento è la terminazione del gruppo di note.

Per quanto riguarda le informazioni di carattere topologico, esse riguardano:

- le dimensioni del bounding box dell'immagine contente il simbolo di base
- i parametri caratteristici del pentagramma: spessore della linea e distanza tra due linee

L'uscita del riconoscitore è costituita dal nome del simbolo associata all'immagine, le coordinate assolute, le sue dimensioni e la percentuale di confidenza sul riconoscimento.

6.2 Struttutura del modulo di riconoscimento

La realizzazione del classificatore ha richiesto la costruzione di un database delle immagini dei simboli di base ottenuti col processo di segmentazione. Le immagini costituiscono l'insieme dei dati con i quali effettuare la procedura di addestramento del classificatore. Le immagini sono state raggruppate per classi il cui identificativo esprime il significato del simbolo che includono. Prima di descrivere la struttura del classificatore, viene descritto il database usato e l'insieme dei simboli di base presi in considerazione.

6.3 Database dei simboli di base

Le immagini prodotte dalla scomposizione devono essere ben definite per costruire un archivio di elementi grafici con il quale operare l'addestramento della rete neurale preposta al riconoscimento. La scelta delle classi che costituiscono l'insieme dei simboli di base

(si veda Tabella 6.1) è stata indirizzata verso la copertura completa dei simboli grafici musicali, testuali e numerici ricorrenti nella musica classica. L'archivio delle immagini di spariti monofonici a disposizione, ha permesso di collezionare oltre 10000 immagini e di definire un insieme di 48 classi di simboli di base.

Alcune di queste classi costituiscono già dei simboli musicali completi, mentre le altre rappresentano i componenti elementari, la combinazione dei quali consente di ricostruire simboli composti e complessi. L'elenco ed una descrizione delle classi sono riportati di seguito.

ACCENTO Sono stati raggruppati tutti gli accenti.

BIGLINEx collezionano tutte le configurazioni delle travi che connettono le note all'interno dei gruppi. Tre tipologie di immagini sono state collezionate per ogni classe BIGLINE: travi orizzontali, travi diagonali ascendenti e travi diagonali discendenti. Il numero x è legato al numero di travi, che a loro volta definiscono la durata della nota:

- BIGLINE1: trave singola e definisce la durata di 1/8.
- BIGLINE2: trave doppia e definisce la durata di 1/16.
- BIGLINE3: trave tripla e definisce la durata di 1/32.
- BIGLINE4: trave quadrupla e definisce la durata di 1/64.

CLEFTREBLE contiene le chiavi di violino

CLEFBASS contiene le chiavi di basso.

CLEFALTO contiene le chiavi di DO.

CHORDF2V contiene accordi di due note nere connesse tra loro; sono le configurazioni di bicordi che realizzano un'intervallo di terza.

CHORDF3V contiene accordi di tre note nere connesse tra loro, si tratta delle configurazioni di tricordi che realizzano un'intervallo di terza e uno di seconda.

DBARLINE contiene i tratti continui che attraversano il pentagramma e che si presentano come la composizione di due barre verticali. Queste barre sono generate dalle linee di fine battuta, di inizio o di fine ritornello.

DIGITx collezionano i numeri usati (espressi dal valore di x) nelle frazioni per indicare il tempo in chiave: DIGIT1, DIGIT12, DIGIT16, DIGIT2, DIGIT3, DIGIT4, DIGIT5, DIGIT6, DIGIT8, DIGIT9.

DOT contiene immagini di punti, principalmente ottenuti dai punti di valore, ma allo stesso tempo sono rappresentativi dell'accento di tipo *staccato*.

DSHARP contiene i doppi diesis.

ENOTEHEAD contiene le teste di note vuote relative alle minime, pur essendo simile alla semibreve, si è ritenuto separarle in quanto un'attenta osservazione mette in evidenza alcune differenze grafiche, ma soprattutto introdurrebbero delle indeterminazioni nella fase di ricostruzione nel momento in cui si deve stabilire la corretta durata della nota.

FLAT contiene i bemolli.

FNOTEHEAD contiene le teste di note piene/nere relative alle note con durata inferiore ad 1/4.

HALF_REST8 contiene la metà superiore della pausa da 1/8, è stato necessario introdurre questa particolare classe per far fronte ad una frammentazione delle pause, relative alle note con valore inferiore all'ottavo, dovuta ad una segmentazione non corretta. In questo modo la pausa potrà essere assemblata perchè vista come la composizione di simboli di tipo HALFREST8.

HOOK1DWN contiene le partenze dell'uncino singolo discendente per le note da 1/8.

HOOK2DWN contiene le partenze dell'uncino doppio discendente per le note da 1/16.

HOOK3DWN contiene le partenze dell'uncino triplo discendente per le note da 1/32.

HOOK1UP contiene le partenze dell'uncino singolo ascendente per le note da 1/8.

HOOK2UP contiene le partenze dell'uncino doppio ascendente per le note da 1/16.

HOOK3UP contiene le partenze dell'uncino doppio ascendente per le note da 1/32.

LOWER e GREATER sono state introdotte per non perdere le informazioni sulle legature e sui segni di dinamica quali il crescendo e il diminuendo. In particolare, esse rappresentano le configurazioni di partenza e di fine generate dalla legatura nelle situazioni in cui essa attraversa una linea di pentagramma, oppure la parte iniziale o terminale di un crescendo o un decrescendo.

NATURAL contiene i bequadri.

REST4 contiene le pause da 1/4.

REST8 contiene le pause da 1/8.

REST16 contiene le pause da 1/16.

REST32 contiene le pause da 1/32.

REST64 contiene le pause da 1/64.

SBARLINE contiene le barre di suddivisione delle battute.

SHARP contiene i diesis.

TDF contiene il simbolo dinamico di *forte*

TDP contiene il simbolo dinamico di *piano*

THINLINE è rappresentativa di tutte le porzioni di linee con spessore comparabile con quello delle linee di pentagramma. Essa racchiude tutte le immagini relative a parti di legature, crescendo, diminuendo e linee di pentagramma.

WHOLENOTE contiene le note da un intero (semibrevi).

6.4 Classificatore Neurale MLP

Visti i risultati ottenuti dalla classificazione basata su reti neurali nei sistemi OCR, la stessa tecnica è stata adottata per realizzare il classificatore dei simboli di base. La scelta è stata dettata anche dai vantaggi che offre la tecnica neurale. Le reti:

- riproducono comportamenti complessi (superfici di separazione non lineari)
- hanno una buona immunità al rumore, permettendo di apprendere da dati rumorosi e/o incompleti
- permettono una processazione parallela e quindi maggiore velocità
- sono indipendenti dal campo di applicazione

Per contro, però, si possono citare alcuni svantaggi:

- limiti intrinseci delle diverse forme di apprendimento (convergenza, stabilità, precisione, ecc...)
- il processo di apprendimento è progressivamente più lento all'aumentare delle dimensioni della rete e del numero di dati impegnati
- non si ottengono riposte precise e decise al 100%

Il modello neurale adottato è quello delle reti MLP (*Multi Layer Perceptons Back-propagation*). La rete MLP ([53]) è uno dei modelli connessionistici principali costituito da una rete di tipo feedforward arbitraria, con ingressi ed attivazioni continue. Le attivazioni sono calcolate con la funzione sigmoidale definita dall'equazione:

$$\phi(S_k) = \begin{cases} \frac{1}{1+e^{-S_k}}, & \text{se è usata un'attivazione in } [0,1] \\ \\ -1 + \frac{2}{1+e^{-S_k}}, & \text{se è usata un'attivazione in } [-1,1] \end{cases} \tag{6.1}$$

L'algoritmo di apprendimento utilizzato è il back propagation. Le celle sono ordinate, calcolano la loro attivazione e la pongono in ingresso alla cella successiva prima che questa venga a sua volta esaminata. La rete MLP, utilizzata come classificatore presenta, inoltre, le seguenti caratteristiche:

- Gli oggetti da classificare (nel caso specifico gli elementi grafici ricavati dalla segmentazione dello spartito musicale) possono essere posti direttamente in ingresso alla rete, dopo un'opportuna normalizzazione sia della dimensione dell'oggetto, sia dell'intervallo di variazione dell'ingresso. Infatti, la dimensione dovrà essere compatibile con il numero di neuroni in ingresso alla rete, mentre i valori in ingresso devono essere compatibili con la funzione di attivazione utilizzata. Nel nostro caso, le immagini vengono normalizzate ad una dimensione uniforme per tutte e il livello di grigio viene portato nell'intervallo di variazione $[0,1]$, visto che la funzione di attivazione utilizzata è una sigmoide il cui dominio e codominio corrispondono a questo intervallo.

- Le uscite della rete sono pari al numero di classi, ed i valori corretti di uscita per la classe k, impostati dall'apprendimento, sono definiti dall'equazione 6.2:

$$C_i^k = \begin{cases} 1 & \text{se } i = k \\ 0 & \text{se } i \neq k \end{cases} \tag{6.2}$$

- La scelta del numero di ingressi e del numero di neuroni dello strato nascosto viene fatta valutando la convergenza e la generalizzazione della rete a nuovi esempi.

In particolare, nel caso dei simboli grafici musicali elementari, la riduzione dell'immagine originale alla dimensione del pattern, porta ad una certa perdita di informazione che deve essere trascurabile. Questo significa che, il rapporto di compressione deve essere tale da poter rilevare anche nell'immagine ridotta, i dettagli che servono nel discriminare le singole classi. Quindi, nel caso di immagini di piccole dimensioni è necessario non utilizzare una compressione troppo elevata, mentre nel caso di immagini di dimensioni maggiori, occorre ridurre in modo notevole l'immagine per i seguenti motivi:

- La presenza di un numero elevato di neuroni in ingresso contribuisce ad aumentare notevolmente la complessità computazionale della rete sia nella fase di apprendimento che in quella di riconoscimento.
- L'incremento della compressione genera un effetto di filtraggio di tipo *passa-basso* dell'immagine che ha effetti positivi per l'eliminazione del rumore.

Il numero dei neuroni dello strato nascosto viene fissato tenendo conto di alcune considerazioni di tipo empirico. È possibile rilevare, infatti, un miglioramento dell'apprendimento all'aumentare di tale numero, ma anche una minore generalizzazione. Incrementando il numero di questi neuroni, vengono aumentati i gradi di libertà della rete che riesce ad adattarsi in modo peggiore il *trend* degli stessi.

Specifiche del classificatore adottato – La struttura usata è quella *full connected* costituita da 128 ingressi, un livello nascosto di 256 neuroni e 48 neuroni di uscita, corrispondente al numero delle classi definite per i simboli di base. Il numero degli ingressi corrisponde al numero di pixels di un'immagine di 8x16 pixels. I patterns di ingresso sono calcolati vettorizzando l'immagine e normalizzando i valori di detti pixels (si veda Figura 6.2). La normalizzazione avviene considerando il valore massimo della scala dei grigi invertita in modo da associare il valore di 255 al nero. In questo modo i valori sono riportati entro l'intervallo [0,1]. Infine la funzione di attivazione sigmoidale mappa valori nell'intervallo [0,1].

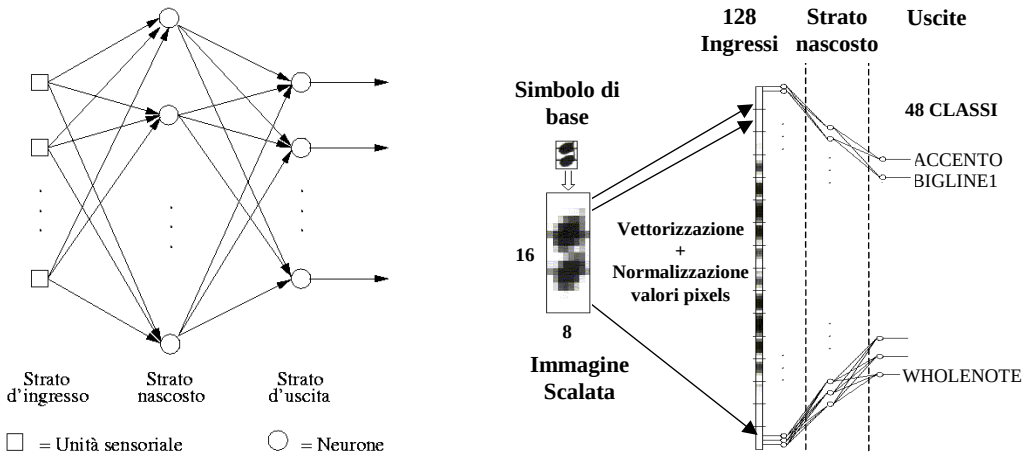


Figura 6.2: Struttura classificatore (a) e vettorizzazione (b)

La scalatura delle immagini: minimizzazione del bounding box – Prima di effettuare la scalatura delle dimensioni delle immagini da classificare, si effettua un'operazione di ritaglio escludendo (*cropping*) il pentagramma (eliminazione delle aree periferiche

CLASSE	LEARNING	TEST	CLASSE	LEARNING	TEST
ACCENTO	126	70	ENOTEHEAD	97	64
BIGLINE1	246	219	FLAT	86	61
BIGLINE2	246	126	FNOTEHEAD	346	171
BIGLINE3	309	190	GREATER	132	81
BIGLINE4	247	134	HALF_REST8	167	48
CHORDF2V	189	105	HOOK1DWN	206	122
CHORDF3V	252	102	HOOK1UP	167	105
CLEFALTO	131	88	HOOK2DWN	59	59
CLEFBASS	125	88	HOOK2UP	70	61
CLEFTREBLE	234	82	HOOK3DWN	50	28
COMMA	132	81	HOOK3UP	42	42
DBARLINE	124	72	LOWER	104	90
DIGIT1	32	32	NATURAL	105	68
DIGIT12	34	34	REST16	289	106
DIGIT16	40	40	REST32	274	112
DIGIT2	123	67	REST4	83	56
DIGIT3	100	72	REST64	54	45
DIGIT4	143	73	REST8	110	83
DIGIT5	57	53	SBARLINE	124	72
DIGIT6	128	72	SHARP	148	83
DIGIT8	244	69	TDF	66	48
DIGIT9	32	32	TDP	70	63
DOT	250	99	THINLINE	181	144
DSHARP	42	42	WHOLENOTE	120	81

Figura 6.3: Numero simboli usati per l'addestramento

contententi solo linee di pentagramma) ed elimindando i bordi bianchi che circondano il simbolo grafico. Questa procedura è necessaria per ridurre le dimensioni del bounding box e uniformare le dimensioni delle immagini. Più in dettaglio, l'eliminazione del pentagramma viene eseguita con la tecnica delle proiezioni, in particolare la proiezione-X. Si effettua, quindi, un filtraggio passa-alto (filtro *unsharp gaussian*) per eliminare il contributo costante delle linee di pentagramma, quindi si estrae l'immagine per mezzo di una soglia. Il valore di tale soglia è definito come l'1% del valore massimo della proiezione. Infine, le dimensioni ottenute al termine dell'operazione di ritaglio diventano i nuovi riferimenti topologici.

6.4.1 L'insieme d'addestramento e di test

La rete è stata addestrata avvalendosi di due insiemi di simboli di base, uno per la fase di training e l'altro per la fase di testing. Complessivamente l'insieme di addestramento è costituito da 6736 elementi, mentre quello di test da 3935. Il numero di elementi per classe è riportato nella Figura 6.3.

6.4.2 Prestazioni e risultati ottenuti nell'addestramento

Al termine dell'addestramento la percentuale di riconoscimento ha raggiunto il valore intorno all'88%. Tale valore non è un parametro assoluto per la valutazione del riconoscimento

e sul quale non può essere fatta alcuna considerazione sul rendimento effettivo di un riconoscitore ottico di caratteri musicali, la vera misura della prestazione può essere ottenuta considerando l'intero sistema, e quindi al termine della fase di ricostruzione. Di seguito è riportata la matrice di confusione e il risultato conseguito dalla rete sopra definita.

Class	N.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
accento	0	61	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
bigline1	1	.	202	.	.	.	.	.	.	.	.	.	.	.	1	.	.
bigline2	2	1	5	74	3	8	2	.	.	.	.	.	.	.	3	1	1
bigline3	3	.	1	.	145	11	.	7	.	.	.	.	.	.	.	.	.
bigline4	4	.	2	6	5	111	.	2	2	.	.	.	1	.	.	.	.
chordf2v	5	.	.	.	.	.	105	.	.	.	.	.	.	.	.	.	.
chordf3v	6	.	.	.	.	.	.	102	.	.	.	.	.	.	.	.	.
clefalto	7	.	.	.	.	.	.	.	82	.	.	.	.	1	.	.	.
clefbass	8	.	.	.	.	.	.	.	.	82	.	1	.	.	.	.	.
cleftreble	9	.	.	.	.	.	.	.	.	.	82	.	.	.	.	.	.
comma	10	.	.	.	2	.	.	.	1	1	.	52	.	.	.	.	.
dbarline	11	.	.	.	.	.	.	.	.	.	.	.	72	.	.	.	.
digit1	12	.	.	.	.	.	.	.	.	.	.	.	.	32	.	.	.
digit12	13	.	.	.	.	.	.	.	.	.	.	.	.	.	34	.	.
digit16	14	.	.	.	.	.	.	.	.	.	.	.	.	.	.	38	.
digit2	15	.	.	.	2	.	.	.	.	.	.	.	.	.	1	.	55
digit3	16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit4	17	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit5	18	.	2	2	.	.	.	.	.	.	.	.	.	.	.	.	.
digit6	19	.	.	2	3	.	.	.	.	.	.	.	.	.	.	.	.
digit8	20	.	1	.	.	.	.	.	.	.	.	.	.	.	1	.	.
digit9	21	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	.
dot	22	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
dsharp	23	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.
enotehead	24	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
flat	25	.	.	.	.	.	.	.	.	.	.	.	1	.	.	2	.
fnotehead	26	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
greater	27	4	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.
half_rest8	28	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.
hook1dwn	29	.	2	.	.	.	.	.	.	.	.	1	.	.	.	.	.
hook1up	30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
hook2dwn	31	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	1
hook2up	32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
hook3dwn	33	.	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.
hook3up	34	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
lower	35	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
natural	36	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.
rest16	37	1	.	.	.	1	.	.	.	.	.	.	.	.	.	1	.
rest32	38	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.
rest4	39	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
rest64	40	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
rest8	41	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
sbarline	42	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
sharp	43	.	.	1	.	.	1	.	.	.	.	.	.	.	.	.	.
tdf	44	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
tdp	45	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	1
thinline	46	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.
wholenote	47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Class	N.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
accento	0	.	.	.	.	.	.	.	.	.	1	.	2	.	2	.	1
bigline1	1	.	.	.	.	.	.	.	.	.	.	2	1	.	.	3	.
bigline2	2	.	.	.	.	.	.	.	.	.	.	1	.	.	9	2	7
bigline3	3	.	.	.	1	.	.	1	.	1	.	.	.	.	.	.	1
bigline4	4	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.
chordf2v	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
chordf3v	6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
clefalto	7	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.
clefbass	8	.	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.
cleftreble	9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
comma	10	.	.	.	4	.	.	.	.	.	2	.	13	.	1	2	.
dbarline	11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit1	12	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit12	13	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit16	14	.	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.
digit2	15	2	.	.	.	.	.	2	.	.	.	.	.	4	.	.	.
digit3	16	68	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.
digit4	17	.	70	.	1	.	.	.	.	.	.	.	.	.	.	2	.
digit5	18	.	.	41	.	.	.	.	.	.	.	4	.	2	.	.	.
digit6	19	.	.	.	57	.	.	.	1	4	.	2	.	1	.	.	.
digit8	20	.	.	.	3	49	.	1	.	5	.	3	.	2	.	3	.
digit9	21	.	.	.	.	.	24	.	.	2	.	.	.	.	.	2	.
dot	22	.	1	.	.	.	.	90	.	3	.	.	3	.	.	.	.
dsharp	23	.	.	.	.	.	.	.	38	.	.	2	.	.	.	.	.
enotehead	24	.	.	.	.	.	.	.	.	64	.	.	.	.	.	.	.
flat	25	.	1	.	.	.	.	.	.	1	49	.	.	.	.	.	.
fnotehead	26	.	.	.	1	.	.	5	.	.	.	164	.	1	.	.	.
greater	27	.	.	.	.	.	.	.	.	.	.	.	63	2	.	1	.
half_rest8	28	.	2	.	7	.	.	.	.	.	.	1	.	35	.	.	.
hook1dwn	29	.	1	.	.	.	.	2	1	8	.	2	.	1	87	.	.
hook1up	30	.	.	.	.	.	.	1	.	.	.	.	.	.	.	103	.
hook2dwn	31	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	55
hook2up	32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
hook3dwn	33	.	4	.	.	.	.	2	.	.	.	.	.	.	.	.	2
hook3up	34	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
lower	35	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
natural	36	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.
rest16	37	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
rest32	38	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	1
rest4	39	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2
rest64	40	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
rest8	41	.	.	.	.	3	.	.	.	.	.	.	.	1	.	.	.
sbarline	42	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
sharp	43	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1
tdf	44	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.
tdp	45	.	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.
thinline	46	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
wholenote	47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Class	N.	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
accento	0	.	.	.	.	2	.	.	.	.	.	.	.	.	.	1	.
bigline1	1	.	.	.	1	.	1	.	.	.	.	.	.	.	.	6	.
bigline2	2	.	.	.	.	1	1	1	.	2	.	.	4	.	.	.	.
bigline3	3	1	17	.	.	.	.	2	1	.	.	1	.	.	.	.	.
bigline4	4	.	.	.	.	2	.	.	.	1	.	.	.	.	.	.	.
chordf2v	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
chordf3v	6	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
clefalto	7	.	.	1	.	.	.	.	.	.	.	.	.	.	.	2	.
clefbass	8	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.
cleftreble	9	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
comma	10	.	2	.	.	.	.	.	.	.	.	1	.	.	.	.	.
dbarline	11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit1	12	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit12	13	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit16	14	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit2	15	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1
digit3	16	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit4	17	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
digit5	18	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2
digit6	19	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2
digit8	20	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.
digit9	21	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2
dot	22	.	.	.	1	.	.	.	.	.	.	1	.	.	.	.	.
dsharp	23	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
enotehead	24	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
flat	25	.	3	.	2	.	.	.	.	.	.	2	.	.	.	.	.
fnotehead	26	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
greater	27	.	.	.	1	.	.	.	.	.	.	.	.	2	3	4	.
half_rest8	28	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2
hook1dwn	29	.	2	.	3	.	1	1	.	.	1	.	.	.	.	8	1
hook1up	30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.
hook2dwn	31	.	.	.	.	1	.	.	1	.	.	.	.	.	.	.	.
hook2up	32	61	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
hook3dwn	33	.	12	.	.	.	.	4	.	.	.	.	2	.	.	.	.
hook3up	34	.	.	42	.	.	.	.	.	.	.	.	.	.	.	.	.
lower	35	.	.	.	88	.	.	.	.	.	.	.	.	.	.	2	.
natural	36	.	1	.	.	63	2	.	.	.	.	.	.	.	.	.	.
rest16	37	.	1	.	.	.	101	.	1	.	.	.	.	.	.	.	.
rest32	38	.	.	.	.	.	1	101	1	.	1	.	.	.	.	.	.
rest4	39	.	.	.	.	.	2	.	52	.	.	.	.	.	.	.	.
rest64	40	.	.	.	.	.	.	1	.	44	.	.	.	.	.	.	.
rest8	41	.	.	.	.	.	.	.	.	.	79	.	.	.	.	.	.
sbarline	42	.	.	.	.	.	.	.	.	.	.	71	.	.	.	1	.
sharp	43	2	1	.	.	1	1	.	.	.	.	.	75	.	.	.	.
tdf	44	.	.	.	.	.	.	.	1	.	2	.	.	44	.	.	.
tdp	45	.	.	.	.	.	.	.	.	.	.	2	.	.	53	2	.
thinline	46	.	.	.	.	.	.	.	.	.	.	.	.	.	.	143	.
wholenote	47	.	.	.	7	.	.	.	.	.	.	.	.	.	.	8	66

Class	N.	Rejected	Failed	Succeded
accento	0	.	9	61
bigline1	1	2	15	202
bigline2	2	.	52	74
bigline3	3	.	45	145
bigline4	4	.	23	111
chordf2v	5	.	.	105
chordf3v	6	.	.	102
clefalto	7	.	6	82
clefbass	8	.	6	82
cleftreble	9	.	.	82
comma	10	.	29	52
dbarline	11	.	.	72
digit1	12	.	.	32
digit12	13	.	.	34
digit16	14	.	2	38
digit2	15	.	12	55
digit3	16	.	4	68
digit4	17	.	3	70
digit5	18	.	12	41
digit6	19	.	15	57
digit8	20	.	20	49
digit9	21	.	8	24
dot	22	.	9	90
dsharp	23	.	4	38
enotehead	24	.	.	64
flat	25	.	12	49
fnotehead	26	.	7	164
greater	27	.	18	63
half_rest8	28	.	13	35
hook1dwn	29	.	35	87
hook1up	30	.	2	103
hook2dwn	31	.	4	55
hook2up	32	.	.	61
hook3dwn	33	.	16	12
hook3up	34	.	.	42
lower	35	.	2	88
natural	36	.	5	63
rest16	37	.	5	101
rest32	38	.	11	101
rest4	39	.	4	52
rest64	40	.	1	44
rest8	41	.	4	79
sbarline	42	.	1	71
sharp	43	.	8	75
tdf	44	.	4	44
tdp	45	.	10	53
thinline	46	.	1	143
wholenote	47	.	15	66

6.5 Recupero dell'errore di riconoscimento

Dai test effettuati, è stato rilevata l'importanza di conoscere il comportamento generale della rete, soprattutto nella generazione delle confusioni e negli errore di riconoscimento sistematici (ovvero simboli che vengono frequentemente confusi con altri). La conoscenza di ciò ha condotto ad affiancare alla rete un insieme di euristiche, che nelle situazioni di ambiguità e bassa confidenza sul riconoscimento siano in grado di risolvere l'incertezza dell'interpretazione a favore di una migliore classificazione. L'insieme delle euristiche è stato costruito basandosi sulle informazioni topologiche e contestuali provenienti dalla fase di segmentazione e descritte nella sezione 6.1.

6.6 L'uscita del modulo di riconoscimento

L'uscita del modulo di riconoscimento è costituita da un file testuale nel quale sono listati tutti i simboli di base riconosciuti. Le informazioni di contesto che arrivano dal segmentatore e legate ai segmenti di immagine contenenti i simboli di base sono tradotte in riferimenti testuali ed inserite nel file. Tali riferimenti sono principalmente strutturali e ad essi sono associate le seguenti stringhe:

BEGIN e END stabiliscono l'inizio e la fine di un gruppo di note. Questo consente di segnalare che i simboli di base compresi tra le due stringhe appartengono ad un gruppo di note connesse orizzontalmente tra loro:

BEGIN
< *descrizione* 1 >
< *descrizione* 2 >
... ...
< *descrizione* n >
END

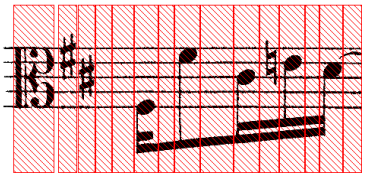
ENDSTAFF segnala la fine di un pentagramma. Esso coincide con il ritorno a capo del pentagramma e consente di mantenere la struttura dello spartito.

STAFF5 è associato agli aggiornamenti delle coordinate delle linee del pentagramma. Questo aggiornamento è eseguito mediante una stringa la cui struttura assume la seguente forma:

$$STAFF5 \;\; X \; Y_5 \; Y_4 \; Y_3 \; Y_2 \; Y_1 \tag{6.3}$$

dove:

- X rappresenta la coordinata lungo il pentagramma relativa al punto di rilevamento dei nuovi valori



USCITA

```
STAFF5 48 57 75 94 113 133
CLEFALTO 49 54 54 84 0.31
STAFF5 107 57 76 95 114 134
SHARP 108 37 25 63 0.99
STAFF5 131 57 76 95 114 134
SHARP 134 65 23 61 0.90
STAFF5 207 57 76 95 114 133
BEGIN
FNOTEHEAD 208 123 32 23 0.99
BIGLINE1 208 165 32 14 0.55
BIGLINE1 208 180 32 15 0.81
BIGLINE1 240 178 21 14 0.97
FNOTEHEAD 261 56 34 23 0.99
BIGLINE1 261 174 34 16 0.50
THINLINE 295 170 44 16 0.36
FNOTEHEAD 339 83 34 25 0.98
BIGLINE1 339 152 34 14 0.80
BIGLINE1 339 168 34 14 0.61
NATURAL 373 46 26 61 0.50
BIGLINE1 373 150 26 14 0.51
BIGLINE1 373 166 26 14 0.66
FNOTEHEAD 399 64 34 25 0.99
BIGLINE1 399 145 34 15 0.68
BIGLINE1 399 163 34 15 0.46
BIGLINE1 433 144 19 13 0.95
BIGLINE1 433 160 19 15 0.76
FNOTEHEAD 452 75 31 23 0.99
THINLINE 452 143 31 14 0.97
THINLINE 452 159 31 14 0.97
END
.....
```

Figura 6.4: Esempio di listato in uscita dal classificatore

- le Y rappresentano le coordinate verticali delle cinque linee del pentagramma e sono riportate in ordine crescente: $Y_5 < Y_1$

In questo modo si hanno sempre i riferimenti precisi rispetto ai quali calcolare le altezze delle note e viene compensata l'eventuale inclinazione del pentagramma.

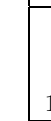
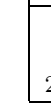
La struttura della stringa descrittiva associata ad ogni simbolo di base riconosciuto ha la seguente forma:

$$< Nome\ simbolo > \ < X >< Y >< dx >< dy >< conf > \tag{6.4}$$

dove:

- $< Nome\ simbolo >$ è il nome della classe di simboli associato al termine della fase di riconoscimento
- X e Y sono le coordinate assolute dell'angolo superiore sinistro relative al bounding box
- dx e dy la larghezza e l'altezza del bounding box
- $conf$ è il valore di confidenza o percentuale di riconoscimento associato dal classificatore

Nella Figura 6.4 è riportato un estratto dell'uscita del classificatore.

N°	Simbolo	Nome della Classe	Descrizione
		(**** Numeri ****)	
1		DIGIT1	1
2		DIGIT2	2
3		DIGIT3	3
4		DIGIT4	4
5		DIGIT5	5
6		DIGIT6	6
7		DIGIT8	8
8		DIGIT9	9
9		DIGIT12	12
10		DIGIT16	16
		(**** Testo Dinamica ****)	
11		TDF	f (forte)
12		TDP	p (piano)
		(**** Simboli Musicali ****)	
13		ACCENTO	Accento
14		BIGLINE1	Beam (Trave) da 1/8
15		BIGLINE2	Beam da 1/16
16		BIGLINE3	Beam da 1/32
17		BIGLINE4	Beam da 1/64
18		CHORDF2V	Accordo 2 note piene vert.
19		CHORDF3V	Accordo 3 note piene vert.
20		CLEFALTO	Chiave di Do
21		CLEFBASS	Chiave di basso
22		CLEFTREBLE	Chiave di violino

Nº	Simbolo	Nome della Classe	Descrizione
23		COMMA	Apostrofo o virgola
24		DBARLINE	Barra verticale doppia
25		DOT	Punto e punto di valore
26		DSHARP	Doppio diesis
27		ENOTEHEAD	Testa di nota vuota
28		FLAT	Bemolle
29		FNOTEHEAD	Testa di nota piena
30		GREATER	Legatura+rigo - maggiore
31		HALF_REST8	Metà superiore pausa 1/8
32		HOOK1DWN	Uncino giù 1/8
33		HOOK1UP	Uncino su 1/8
34		HOOK2DWN	Uncino giù 1/16
35		HOOK2UP	Uncino su 1/16
36		HOOK3DWN	Uncino giù 1/32
37		HOOK3UP	Uncino su 1/32
38		LOWER	Legatura+rigo - minore
39		NATURAL	Bequadro
40		REST16	Pausa da 1/16
41		REST32	Pausa da 1/32
42		REST4	Pausa da 1/4
43		REST64	Pausa da 1/64
44		REST8	Pausa da 1/8
45		SBARLINE	Barra verticale singola
46		SHARP	Diesis
47		THINLINE	Linea sottile
48		WHOLENOTE	Nota semibreve

Tabella 6.1: Database di addestramento della rete neurale

Capitolo 7

Architettura del ricostruttore

Fra le tecniche per l'elaborazione delle immagini, la ricostruzione dell'informazione assume un ruolo importante per la comprensione dei documenti. Il compito svolto dal ricostruttore è quello di ricostruire il documento dalle informazioni estratte nelle fasi di segmentazione e di riconoscimento.

Prima di di procedere con lo studio di un modello di ricostruttore valido e funzionale è utile analizzare il campo di applicazione, nel quale si opera, e quali sono le informazioni in possesso. Un ruolo di primaria importanza è rivestito dallo studio del dominio dal quale sono tratte le immagini da ricostruire. Esiste una notevole differenza, ad esempio, se si devono trattare dei documenti letterari o degli spartiti musicali. Tale differenza sta nel fatto che le primitive estratte e riconosciute dalle immagini possono essere legate tra loro tramite vincoli derivanti da regole (nel caso di spartiti musicali) o possono essere totalmente indipendenti (nel caso di documenti letterari). È quindi opportuno studiare un sistema in grado di utilizzare le informazioni derivanti dal contesto. A questo scopo è importante analizzare come deve essere strutturata tale informazione.

Il sistema di ricostruzione deve essere in grado di correggere gli errori e di eliminare le ambiguità introdotte dalle precedenti fasi di segmentazione e di riconoscimento delle primitive, al fine di fornire un documento finale che sia il più possibile fedele all'originale.

In questo capitolo, sono affrontati questi aspetti applicati all'analisi di spartiti musicali al fine di ottenere un modello efficiente del ricostruttore. I termini simboli di base e primitive sono utilizzati con lo stesso significato usato in precedenza.

7.1 Problematiche relative al ricostruttore

Il ricostruttore, nel sistema OMR, si pone tra il riconoscitore di primitive e l'editor musicale utilizzato per la visualizzazione dell'informazione ricostruita: in questo caso è stato

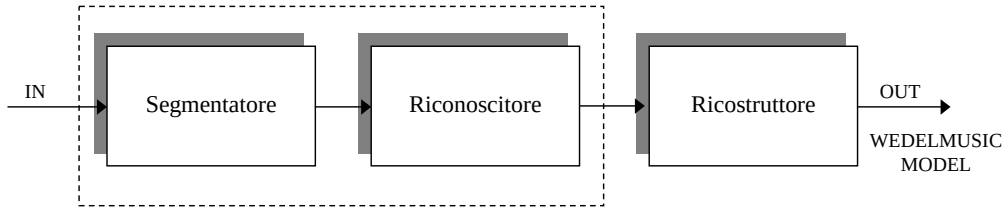


Figura 7.1: Diagramma schematico del sistema OMR.

adottato l'editor musicale del sistema WEDELMUSIC. Tale posizione, relativamente all'intero sistema, è sottoposta agli errori introdotti sia dalla fase di segmentazione che da quella di riconoscimento delle primitive. Questo può condizionare il contenuto informativo rendendolo ambiguo.

Una prima ipotesi è quella di considerare gli errori dovuti alla fase di riconoscimento, ovvero considerare il sistema come l'insieme di due blocchi: il primo costituito dal sottosistema segmentatore più riconoscitore ed il secondo dal ricostruttore (si veda Figura 7.1). Questa ipotesi è la naturale affermazione che gli errori compiuti durante la fase di segmentazione si propagano, in modo più o meno evidente, nella fase di riconoscimento. Si ha infatti, ad esempio, che se il segmentatore "sporca" le primitive grafiche, inserendo del rumore, queste sono classificate dalla rete neurale con un livello di confidenza molto basso. Gli errori dovuti alla fase di riconoscimento possono essere originati da vari fattori quali: errori di stampa del testo (ad esempio macchie bianche dentro delle teste di nota piena, ecc..), variazioni delle dimensioni dei simboli (linee del pentagramma che variano di spessore), etc. La presenza di tali disturbi influenza il processo di riconoscimento abbassando il livello di confidenza associato alla primitiva riconosciuta o addirittura introducendo un errore di classificazione.

Quanto esposto si riflette sugli ingressi del ricostruttore che non possono essere considerati totalmente attendibili e completi per ricostruire lo spartito in ingresso nella sua interezza. Sorgono pertanto i seguenti quesiti: come recuperare l'informazione persa? Come eliminare le ambiguità?

Le risposte a queste domande possono essere molteplici e tutte portano alla definizione di un modello diverso del ricostruttore e a risultati che possono essere più o meno soddisfacenti.

7.2 Modello del ricostruttore

Il modello del ricostruttore messo a punto ed implementato è stato chiamato MOOR (**M**usic **O**bject **O**riented **R**econstructor) ed è rappresentato in Figura 7.2. Nei paragrafi

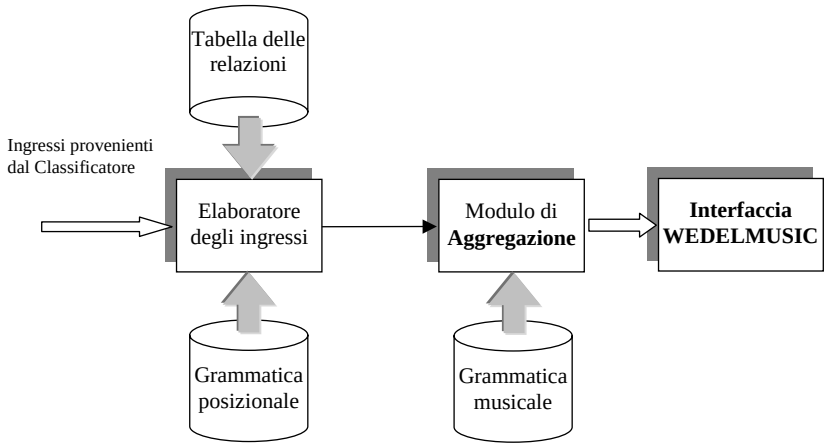


Figura 7.2: Schema del ricostruttore.

seguenti è presentata una prima analisi del modello partendo dallo studio dei dati in ingresso, e successivamente una descrizione dei moduli che costituiscono la struttura.

7.2.1 Struttura degli ingressi al ricostruttore

Gli ingressi del ricostruttore provengono dalla fase di riconoscimento delle primitive grafiche e sono strutturati secondo le regole e la formalizzazione definita nel Capitolo 6. Per comprendere a fondo la struttura dei dati in ingresso di seguito è riportato l'insieme degli ingressi relativo alla Figura 7.3.

```
**INGRESSI AL RICOSTRUTTORE**

//STRIP RELATIVA AI VALORI DELLE COORDINATE DEL PENTAGRAMMA
STAFF5 800 359 378 397 415 435

//STRIP RELATIVA ALLA BARLINE
SBARLINE 881 358 12 84 0.9
THINLINE 881 470 12 11 0.9

//STRIP RELATIVA ALLA PRIMA NOTA DEL VALORE DI 1/4
THINLINE 905 357 27 10 0.9
THINLINE 905 375 27 11 0.6
THINLINE 905 394 27 12 0.6
FNOTEHEAD 905 414 27 28 0.9
THINLINE 905 470 27 11 0.9
THINLINE 905 482 27 9 0.9
THINLINE 905 496 27 11 0.9

//STRIP RELATIVA AL BEQUADRO
THINLINE 974 357 19 8 0.9
THINLINE 974 375 19 9 0.9
THINLINE 974 394 19 8 0.9
NATURAL 974 407 19 58 0.9
```

```
THINLINE 974 468 19 9 0.9
THINLINE 974 482 19 9 0.9
THINLINE 974 492 19 9 0.9

//STRIP RELATIVA AI VALORI DELLE COORDINATE DEL PENTAGRAMMA
STAFF5 990 360 379 398 416 436

//STRIP RELATIVA ALLA NOTA CON UNCINO DEL VALORE DI 1/8
THINLINE 1001 358 26 8 0.9
HOOK1DWN 1001 375 26 28 0.8
THINLINE 1001 413 26 8 0.6
FNOTEHEAD 1001 425 26 22 0.9
THINLINE 1001 465 26 8 0.8
THINLINE 1001 483 26 6 0.9
THINLINE 1001 490 26 6 0.9

//STRIP RELATIVA ALL'ULTIMA NOTA DELLA BATTUTA DEL VALORE DI 1/4
THINLINE 1061 355 26 8 0.9
THINLINE 1061 374 26 8 0.9
THINLINE 1061 392 26 10 0.7
THINLINE 1061 411 26 10 0.7
FNOTEHEAD 1061 429 26 25 0.9
THINLINE 1061 465 26 8 0.4
THINLINE 1061 482 26 11 0.6

//STRIP RELATIVA ALLA PUNTO
THINLINE 1095 354 14 10 0.9
THINLINE 1095 372 14 10 0.9
THINLINE 1095 390 14 11 0.9
THINLINE 1095 410 14 11 0.9
DOT 1095 428 14 22 0.9
THINLINE 1095 481 14 11 0.9

//STRIP RELATIVA AI VALORI DELLE COORDINATE DEL PENTAGRAMMA
STAFF5 1110 361 380 399 417 437

//STRIP RELATIVA ALLA BARLINE DI FINE BATTUTA
SBARLINE 1185 352 14 86 0.9
```

Ciascun elemento informativo è costituito da una stringa nella quale sono presenti il nome (*token*), che rappresenta la classe associata dalla fase di riconoscimento, i quattro parametri rappresentativi della topologia relativi all'ascissa, all'ordinata, alla larghezza e all'altezza del bounding box della primitiva grafica, ed infine un valore, appartenente all'intervallo [0,1], che rappresenta il livello di confidenza con il quale la rete neurale ha classificato la primitiva. In particolare, si ha che valori elevati del livello di confidenza sono sintomo di un buon riconoscimento da parte della rete neurale, mentre valori minori di 0.6-0.5 rappresentano una scarsa affidabilità del riconoscimento. L'insieme costituito dagli ingressi che presentano lo stesso riferimento dell'ascissa è chiamato *strip* .

Nella riga rappresentata da:

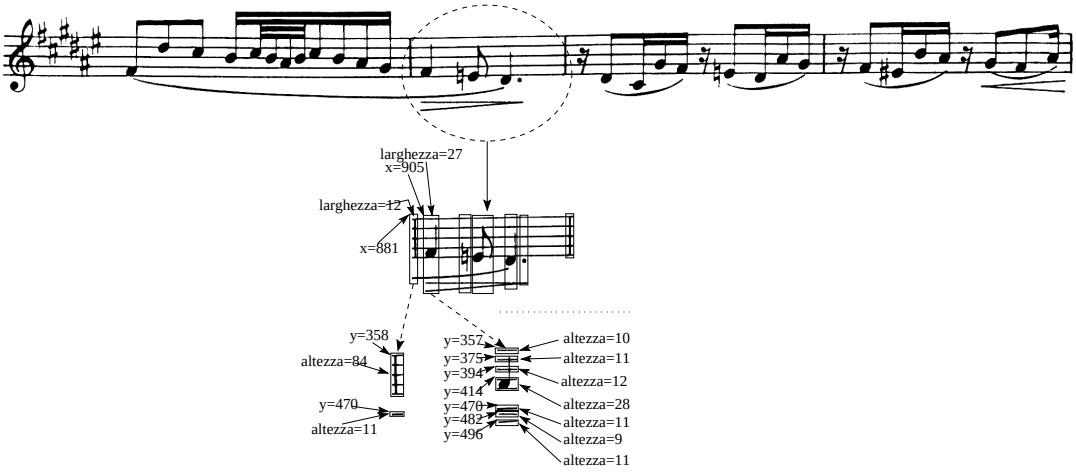


Figura 7.3: Esempio di estrazione delle primitive da una battuta a partire da uno spartito.

STAFF5 800 359 378 397 415 435

l'indicazione, individuata da STAFF5, non rappresenta un token associato dalla rete neurale ad una primitiva, ma definisce un'informazione sull'andamento del pentagramma. Il primo parametro numerico, dopo il nome, rappresenta l'ascissa del pentagramma mentre i cinque successivi rappresentano l'ordinata delle cinque linee del pentagramma, in particolare si ha che il primo rappresenta l'ordinata della prima riga in alto del pentagramma, il secondo la seconda riga e così via. Le coordinate del pentagramma, che rappresentano un'informazione aggiuntiva rispetto alle uscite della rete neurale, sono state introdotte per determinare la posizione, relativamente al pentagramma, delle primitive grafiche. Esaminando attentamente i dati in ingresso riferiti alla Figura 7.3 si nota che la riga relativa alle coordinate del pentagramma è presente svariate volte. Questa ridondanza consente di superare il problema della variazione dello spessore e dell'inclinazione delle linee del pentagramma.

Riferimenti strutturali: gruppi di note

In Figura 7.4 è riportata una battuta contenente gruppi di note. L'insieme degli ingressi relativo alle prime cinque note dell'esempio è definito nel modo seguente:

```
//STRIP RELATIVA ALLA BARLINE
SBARLINE 770 82 10 84 0.98

//STRIP RELATIVA AI VALORI DELLE COORDINATE DEL PENTAGRAMMA
STAFF5 799 85 104 123 142 162

//STRIP RELATIVA ALLA NOTA CON UNCINO DEL VALORE DI 1/8
FNOTEHEAD 800 73 30 25 1.00
THINLINE 800 100 30 8 0.87
```

```
THINLINE 800 120 30 8 0.87
HOOK1UP 800 131 30 15 0.03
THINLINE 800 158 30 8 0.77

//STRIP RELATIVA AL LEGATURA
THINLINE 830 54 69 18 0.93
THINLINE 830 82 69 8 0.83
THINLINE 830 101 69 8 0.91
THINLINE 830 122 69 9 0.73
THINLINE 830 139 69 8 0.83
THINLINE 830 159 69 8 0.88

//STRIP RELATIVA AI VALORI DELLE COORDINATE DEL PENTAGRAMMA
STAFF5 896 84 103 122 141 161

//INDICAZIONE STRUTTURALE DI INIZIO GRUPPO DI NOTE
BEGIN

//STRIP RELATIVA ALLA NOTA DI INIZIO GRUPPO DEL VALORE DI 1/32
FNOTEHEAD 901 72 32 24 0.99
THINLINE 901 100 32 8 0.80
THINLINE 901 120 32 8 0.87
BIGLINE3 901 131 32 47 0.55

//STRIP RELATIVA ALLE TRAVI
THINLINE 933 82 18 8 0.80
THINLINE 933 100 18 8 0.90
THINLINE 933 120 18 9 0.87
BIGLINE3 933 136 18 53 0.94

//STRIP RELATIVA ALLA NOTA DEL VALORE DI 1/32
THINLINE 951 81 32 8 0.81
FNOTEHEAD 951 91 32 23 0.99
THINLINE 951 118 32 8 0.86
THINLINE 951 139 32 8 0.78
BIGLINE3 951 149 32 48 1.00

//STRIP RELATIVA ALLE TRAVI
THINLINE 983 82 24 9 0.93
THINLINE 983 99 24 8 0.81
THINLINE 983 120 24 9 0.75
THINLINE 983 139 24 8 0.83
BIGLINE3 983 156 24 55 0.53

//STRIP RELATIVA ALLA NOTA DEL VALORE DI 1/32
THINLINE 1007 81 32 9 0.83
THINLINE 1007 100 32 8 0.61
FNOTEHEAD 1007 116 32 27 0.99
THINLINE 1007 158 32 9 0.78
BIGLINE3 1007 171 32 48 1.00

//STRIP RELATIVA ALLE TRAVI
THINLINE 1039 81 17 9 0.83
THINLINE 1039 100 17 8 0.61
```

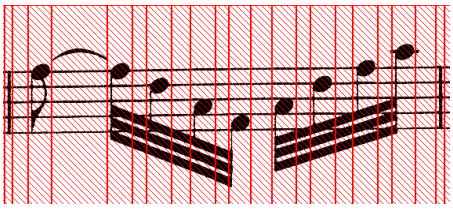


Figura 7.4: Segmentazione di una battuta contente un gruppo di note

```
THINLINE 1039 123 17 9 0.75
THINLINE 1039 138 17 9 0.93
THINLINE 1039 158 17 9 0.93
BIGLINE3 1039 178 17 53 0.89

//STRIP RELATIVA ALLA NOTA DI FINE GRUPPO
THINLINE 983 82 28 9 0.93
THINLINE 983 98 28 8 0.81
THINLINE 983 119 28 9 0.65
FNOTEHEAD 1056 137 28 26 0.94

//INDICAZIONE STRUTTURALE DI FINE GRUPPO DI NOTE
END

//STRIP RELATIVA AI VALORI DELLE COORDINATE DEL PENTAGRAMMA
STAFF5 1109 81 100 119 138 158
```

Tra gli ingressi sono presenti delle indicazioni associate alle righe BEGIN e END. Tali riferimenti, sono di carattere strutturale e definiscono un sottoinsieme di ingressi associati a strip appartenenti ad un gruppo di note. Questo meccanismo consente di discriminare le strip all'interno di un gruppo di note, tenere traccia della struttura e delle relazioni tra i simboli che appartengono al gruppo e di applicare regole di ricostruzione specifiche.

Riferimenti strutturali: fine del pentagramma

La fine fisica del pentagramma è segnalata dall'indicazione ENDSTAFF, posta al termine all'elenco degli ingressi. Questa indicazione permette di mantenere traccia della struttura della pagina e del numero di pentagrammi presenti.

7.2.2 L'elaboratore degli ingressi e la grammatica posizionale

Questo blocco ha il compito di effettuare una prima elaborazione degli ingressi che provengono dal riconoscimento. Tale elaborazione consiste nel determinare l'insieme dei simboli correlati con i token in ingresso definiti nella **Tabella delle Relazioni**, e individuare il simbolo che rappresenta la migliore interpretazione del token. La tabella delle relazioni, descritta nel Capitolo 8, contiene l'insieme dei simboli collegati ai token di uscita della

rete neurale. Per quanto riguarda la determinazione del simbolo da associare al token, questa è effettuata scegliendo nell'insieme quello che ha la maggiore probabilità.

Per determinare la probabilità, chiamata **probabilità di verosimiglianza**, è utilizzato sia il modello probabilistico contenuto nell'archivio **Grammatica posizionale**, sia il livello di confidenza associato dalla rete neurale. Il modello probabilistico è basato sulla probabiltà che un simbolo di base ha di trovarsi in una determinata posizione rispetto al pentagramma. Il principio che ha portato alla formalizzazione di questo modello si basa sullo studio del contesto musicale. Infatti, da un'analisi del contesto si può notare che esistono dei vincoli che riguardano la posizione dei simboli musicali all'interno del pentagramma. Ad esempio, non è possibile trovare un accento o una corona dentro il pentagramma, oppure chiavi di violino sopra o sotto il pentagramma e così via. È importante osservare che questi vincoli non sono solo di tipo esclusivo, nel senso che determinano la posiziona sicura di un simbolo, ma sono di tipo probabilistico ovvero possono essere tradotti in probabilità. Se si considera ad esempio il simbolo musicale corrispondente alla pausa di semicroma si ha che la probabilità che sia al di fuori del pentagramma è molto bassa.

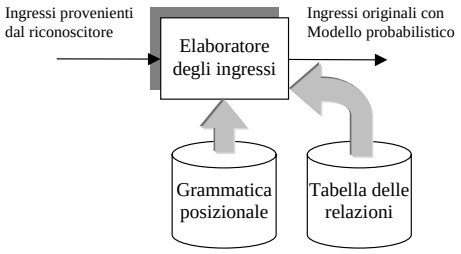


Figura 7.5: Particolare del modello del ricostruttore.

7.2.3 Il modulo di aggregazione e la grammatica musicale

Il modulo di aggregazione rappresenta il motore dell'intera fase di ricostruzione. Gli ingressi al corrispondono alle uscite del modulo di elaborazione degli ingressi (si veda Figura 7.6). Questo blocco ha il compito di aggregare insieme i simboli collegati ai token di una strip in modo da ricostruire il simbolo musicale completo e ricostruire le relazioni tra i simboli musicali composti. Il risultato finale è la ricostruzione dell'intero spartito. La ricostruzione è stata definita sulla base del modello ad oggetti fornito dal formato WEDELMUSIC.

Il processo di aggregazione è svolto, come mostrato in Figura 7.6, tramite l'utilizzo di una **Grammatica musicale**, costituita dall'insieme delle regole grafiche che descrivono la scrittura musicale. Un esempio di regola è quella che concerne la scrittura di una nota con il gambo: se la testa della nota è posizionata nella metà inferiore del pentagramma, se non si è in polifonia e se la nota non fa parte di un gruppo, allora il gambo deve essere direzionato verso l'alto e la sua lunghezza deve essere pari a 3,5 volte la distanza tra le righe del pentagramma. Questa grammatica, quindi, è la rappresentazione dei vincoli locali che il contesto musicale fornisce. I vincoli, come esposto nel Capitolo 3, possono essere sia di tipo locale che globale; quelli locali operano secondo la distanza che intercorre tra i simboli musicali, quindi se due simboli sono relativamente vicini possono interagire fra loro, mentre

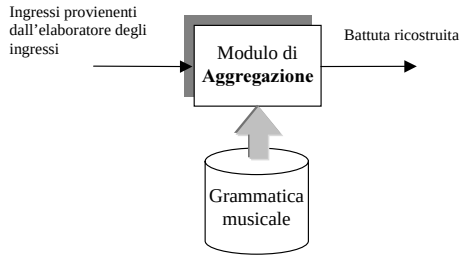


Figura 7.6: Particolare del modello del ricostruttore.

non possono essere relazionati se sono lontani. Quelli globali valutano la correttezza di un insieme di simboli, un esempio è il vincolo che considera la consistenza di una battuta con il tempo in chiave. In particolare, analizzando attentamente i vincoli locali, si ha che questi possono essere separati a seconda che considerino la distanza lungo l'asse delle ordinate o quella lungo l'asse delle ascisse. Ciò condiziona la struttura della grammatica, la quale deve distinguere i vincoli locali, che interessano la struttura dei simboli musicali composti da primitive appartenenti alla stessa strip, dai vincoli che interessano simboli musicali appartenenti a strip diverse. Come esposto nel Capitolo 9 tale considerazione porta alla definizione di regole verticali, che interessano i simboli di una strip, e di regole orizzontali, che interessano simboli di strip diverse. Allo stesso modo il modulo di aggregazione evolve prima aggregando fra loro i simboli appartenenti ad una singola strip e poi collegando i simboli musicali appartenenti a strip diverse.

Capitolo 8

Elaboratore degli ingressi

In questo capitolo è analizzato il primo modulo del sistema di ricostruzione: l'elaboratore degli ingressi. Sono descritti dapprima le informazioni utilizzate e successivamente il modulo stesso.

8.1 Archivi relativi all'elaboratore degli ingressi

Con riferimento alla Figura 8.1, l'elaboratore degli ingressi si avvale della Tabella delle Relazioni, contenente l'insieme dei simboli collegati con le classi di uscita della rete neurale, e dell'archivio relativo alla grammatica posizionale.

8.1.1 La Tabella delle Relazioni

Le classi che costituiscono l'uscita della rete neurale sono state definite in funzione delle immagini, primitive grafiche o simboli di base, in ingresso al riconoscitore e del comportamento della fase di segmentazione. Ogni primitiva grafica, a livello di immagine, può corrispondere ad un simbolo fondamentale (completo) della notazione musicale oppure ad una particolare grafico. La primitiva può essere associata al simbolo di base corrispondente, se è stato definito, oppure ad una classe considerata rappresentativa dalla rete neurale. Questo comportamento realizza una confusione nel riconoscimento. Esaminando l'esempio

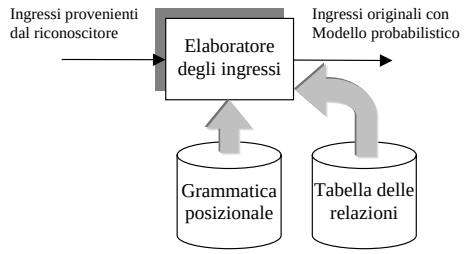


Figura 8.1: Elaboratore degli ingressi

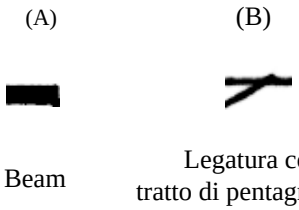


Figura 8.2: Primitive grafiche estratte dal segmentatore

di Figura 8.2: la primitiva grafica (A) corrisponde, nella notazione musicale, al beam di una nota, ma analizzandola solo dal punto di vista grafico può essere confusa con molti altri simboli musicali, quali ad esempio una pausa di semibreve, una pausa di minima, etc. La primitiva (B) corrisponde alla composizione di un segmento di linea del pentagramma con la parte finale di una legatura e può essere confusa con la parte terminale di un crescendo o con un accento. Per determinare l'insieme dei simboli collegati o confusi con una classe si utilizza la **Matrice di Confusione**. Di seguito è riportata una possibile matrice:

*** Confusion table ***

Class	N.	0	1	2	3	4	5	6	7	8	9	10
accento	0	12	.	.	.	.	.	.	.	.	.	.
arc	1	.	5	.	2	.	.	.	.	.	.	.
barline_final	2	.	.	13	.	.	.	.	.	.	.	.
beam-1	3	7	33	.	15	26	2	11	.	1	.	.
beam-2	4	7	1	.	.	22	.	6	.	.	.	2
beam-3	5	2	.	2	.	2	93	.	.	.	3	1
chord-2_noteheads_filled	6	.	.	.	.	.	.	36	.	.	.	.
clef_treble	7	.	.	.	.	.	.	.	11	.	.	.
flat	8	.	.	.	.	.	.	.	.	7	.	.
letter-a	9	.	.	.	.	.	.	.	.	.	13	.
letter-e	10	.	.	.	.	.	.	.	.	.	3	6
letter-m	11	.	.	.	.	.	.	.	.	.	.	.
natural	12	.	.	.	.	.	.	.	.	.	.	.
notehead_empty	13	.	.	.	.	.	.	.	.	.	.	.
notehead_filled	14	.	.	.	18	1	.	.	.	.	.	1
number-3	15	1	.	.	.	1	.	.	.	.	.	.
rest_head	16	.	.	.	2	.	1	.	.	.	.	.
rest_quarter	17	.	.	.	.	.	.	.	.	.	.	.
sharp	18	.	.	.	.	.	.	.	.	.	.	.
staff_line_with_stem_and_hook_cut	19	.	.	.	.	.	.	.	.	.	.	.

Class	N.	11	12	13	14	15	16	17	18	19
accento	0	.	.	.	.	.	.	.	.	.
arc	1	.	.	.	2	.	.	.	.	2
barline_final	2	.	.	.	.	1	.	.	.	1
beam-1	3	3	.	2	8	.	4	.	.	15
beam-2	4	.	.	.	.	1	.	2	2	.
beam-3	5	1	.	1	.	3	1	22	.	.
chord-2_noteheads_filled	6	.	.	.	.	.	.	.	.	.
clef_treble	7	.	.	.	.	.	.	.	.	.
flat	8	.	.	1	.	1	1	.	.	.

letter-a	9	.	.	.	.	.	.	.	.	1
letter-e	10	.	.	.	1	.	.	.	.	.
letter-m	11	31	.	1	.	2	.	.	.	.
natural	12	.	10	.	.	.	.	1	.	.
notehead_empty	13	.	.	12	.	.	.	.	.	.
notehead_filled	14	.	.	.	417	2	1	.	.	1
number-3	15	.	.	.	.	11	.	.	.	.
rest_head	16	2	.	.	.	2	62	.	.	.
rest_quarter	17	.	.	.	.	.	.	15	.	.
sharp	18	.	.	.	.	.	.	.	10	.
staff_line_with_stem_and_hook_cut	19	.	.	.	.	.	.	.	.	8

Per ogni classe si ha che l'insieme dei simboli che può rappresentare è dato dall'unione delle classi con le quali è stata confusa, determinate in base alla matrice di confusione e dal comportamento generale del riconoscitore. Tali classi sono identificabili leggendo la riga delle matrice di confusione in corrispondenza della classe considerata. Con riferimento alla matrice di esempio, se si considera la classe relativa alla testa di nota piena (notehead_filled) si ha che questa può essere confusa, osservando la matrice di confusione, con il beam1, la testa di una pausa, etc. Quindi, si ha che la classe notehead_filled è associata ai simboli: testa di nota piena, beam1, la testa di una pausa, etc. L'insieme delle possibili confusioni associate ad una classe è esteso aggiungendo i simboli rappresentativi di primitive grafiche che pur non rientrando tra le classi di uscita della rete neurale, sono state ad essa associate. In questo modo, i simboli musicali decomposti in primitive grafiche che non fanno parte dell'insieme dei simboli di base possono essere ricostruiti considerando la confusione generata dalla rete e associando un'interpretazione alla classe che è stata selezionata.

Il risultato finale di tale processo di associazioni porta alla definizione di un archivio, chiamato **Tabella delle Relazioni**, nel quale sono memorizzati i simboli associati alle classi rappresentative dei simboli di base.

Struttura della Tabella delle Relazioni – Ogni elemento (riga) della tabella è stato definito nel seguente modo:

$$< token > < S\_confusione\ 1 > \ | \ < S\_confusione\ 2 > \ | \ ... \ | < S\_confusione\ n >$$

dove:

- *token* rappresenta il nome della classe di uscita della rete neurale,
- *S_confusione* rappresenta il nome della classe di uscita della rete neurale o di un simbolo non di base con il quale il token può essere confuso. "S_" è stato introdotto per distinguere il token dal simbolo confuso,

- "|" rappresenta un elemento separatore nella lista delle confusioni possibili. Può essere visto come espressione dell'opzionalità.

L'insieme delle confusioni è chiamato **insieme dei simboli collegati**. Nell'esempio seguente:

```
FNOTEHEAD   S_FNOTEHEAD | S_PUNTVAL | S_BEAM1
```

indica che le primitive grafiche associate alla classe FNOTEHEAD, in uscita della rete neurale possono corrispondere al simbolo S_FNOTEHEAD (testa di nota piena), oppure a S_PUNTVAL (il punto di valore), oppure a S_BEAM1 (tratto di beam).

La tabella che segue è stata ottenuta a partire dalla matrice di confusione generata dalla rete neurale descritta nel Capitolo 6.

```
//CHIAVI
CLEFBASS    S_BASS
CLEFTREBLE  S_TREBLE
CLEFALTO    S_CC | S_CCLOW | S_CCLLOW | S_TENOR

//NUMERI
DIGIT0  S_DIGIT0 | S_CUPO
DIGIT1  S_DIGIT1 | S_C_SX | S_C_DX
DIGIT2  S_DIGIT2 | S_REST8
DIGIT3  S_DIGIT3
DIGIT4  S_DIGIT4 | S_STAFLINESLUR
DIGIT5  S_DIGIT5
DIGIT6  S_DIGIT6
DIGIT7  S_DIGIT7
DIGIT8  S_DIGIT8
DIGIT9  S_DIGIT9
DIGIT12 S_DIGIT12 | S_HOOK1UP
DIGIT16 S_DIGIT16 | S_REST8

//INDICAZIONI DINAMICHE
TDP S_TDP
TDF S_TDF | S_HOOK1UP

//BARRE DI SUDDIVISIONE
SBARLINE    S_SBARLINE
DBARLINE    S_DBARLINE

//TESTE DELLE NOTE
FNOTEHEAD   S_FNOTEHEAD | S_PUNTVAL | S_BEAM1
ENOTEHEAD   S_ENOTEHEAD
WHOLENOTE   S_WHOLENOTE | S_REST8 | S_HOOK1UP

//LINEE SOTTILI O CON SPESSORE
BIGLINE1    S_BEAM1 | S_BEAM2 | S_SLUR
BIGLINE2    S_BEAM2 | S_BEAM3
BIGLINE3    S_BEAM3
BIGLINE4    S_BEAM4 | S_C_SX
```

```
BIGLINE5    S_BEAM5
THINLINE    S_STAFLINE | S_STAFLINESTEM | S_TENUTO | S_DECRESCLINE | S_CRESCLINE | S_SLUR
        | S_CRESCBEGIN | S_DECREEND | S_SLURSTAFLINE | S_STAFLINESLUR

//UNCINI
HOOK1UP S_HOOK1UP | S_BEAM1 | S_REST8 | S_SLUR
HOOK2UP S_HOOK2UP | S_BEAM2
HOOK3UP S_HOOK3UP | S_BEAM3 | S_CHORDE2V
HOOK4UP S_HOOK4UP | S_BEAM4
HOOK5UP S_HOOK5UP | S_BEAM5
HOOK1DWN    S_HOOK1DWN | S_BEAM1 | S_STAFLINESTEM | S_C_DX
HOOK2DWN    S_HOOK2DWN | S_BEAM2
HOOK3DWN    S_HOOK3DWN | S_BEAM3
HOOK4DWN    S_HOOK4DWN | S_BEAM4
HOOK5DWN    S_HOOK5DWN | S_BEAM5

//ALTERAZIONI
SHARP   S_SHARP
FLAT    S_FLAT
NATURAL S_NATURAL
DSHARP  S_DSHARP
DFLAT   S_DFLAT

//STRUTTURE DI ACCORDI DI NOTE
CHORDF2V    S_CHORD2V | S_BEAM2
CHORD2H     S_FNOTEHEAD
CHORDF3V    S_CHORD3V | S_BEAM3
CHORDE2V    S_CHORDE2V

//PAUSE
REST4   S_REST4
REST8   S_REST8
REST16  S_REST16 | S_DIGIT3
REST32  S_REST32
REST64  S_REST64
HALF_REST8 S_HALFREST8 | S_HOOK1UP
WREST S_WREST | S_BEAM1

//SIMBOLI VARI
COMMA   S_COMMA
DOT S_PUNTVAL
GREATER S_SLURSTAFLINE | S_STAFLINESTEM | S_STAFLINE | S_STAFLINESLUR | S_SLUR | S_DECREEND | S_BEAM1
LOWER   S_STAFLINESLUR | S_SLURSTAFLINE | S_CRESCBEGIN | S_SLUR | S_BEAM1
ACCENTO S_ACCENTO | S_STAFLINESLUR | S_SLURSTAFLINE
```

All'interno della tabella sono individuabili gruppi di token espressione di primitive grafiche con cararatteristiche musicali comuni:

- le chiavi (CLEFBASS, CLEFTREBLE e CLEFALTO);
- i numeri (DIGIT0, DIGIT1, ...);
- indicazioni di dinamica (TDF e TDP);

- la barra di suddivisione delle battute (SBARLINE, DBARLINE);
- le teste delle note (FNOTEHEAD, ENOTEHEAD);
- le linee fine (THINLINE) e le linee grosse (BIGLINE1, BIGLINE2, ...);
- gli uncini sù (HOOK1UP, HOOK2UP, ...) e giù (HOOK1DWN, HOOK2DWN, ...);
- le alterazioni (SHARP, FLAT, ...);
- gli accordi (CHORDF2V, CHORD2H, ...);
- le pause (REST4, REST8, ...);
- simboli vari (ACCENTO, COMMA,...).

Per come è stata definita la tabella, ad una classe può corrispondere un simbolo collegato espressione diretta della classe stessa: FNOTEHEAD e S_FNOTEHEAD. Questa corrispondenza è stata introdotta per le classi relative ai simboli di base che sono riconosciuti dalla rete con un alto grado di confidenza. Questo consente di definire delle certezze di tipo probabilistico come è meglio descritto nel Capitolo 9.

Poiché la Tabella delle Relazioni è costruita a partire dalla rete neurale addestrata su un determinato insieme di simboli di base, nel momento in cui è utilizza una rete con diverso insieme di simboli, occorre adottare la tabella delle relazioni definita per la nuova rete. Questo consente di estendere l'insieme dei simboli gestiti nel riconoscimento e di rendere scalabile il sistema.

8.1.2 La Grammatica posizionale

È l'archivio dove è memorizzato il modello probabilistico derivato dallo studio del contesto musicale. L'analisi condotta sugli spartiti musicali ha evidenziato che non tutte le posizioni, relativamente al pentagramma, sono consentite ai vari simboli musicali, e che in generale si possono definire le seguenti sei posizioni:

- pos_1 all'interno del pentagramma;
- pos_2 a cavallo del pentagramma;
- pos_3 al di sopra del pentagramma;
- pos_4 al di sotto del pentagramma;
- pos_5 all'interno e uscendo in parte al di sopra del pentagramma;
- pos_6 all'interno e uscendo in parte al di sotto del pentagramma.

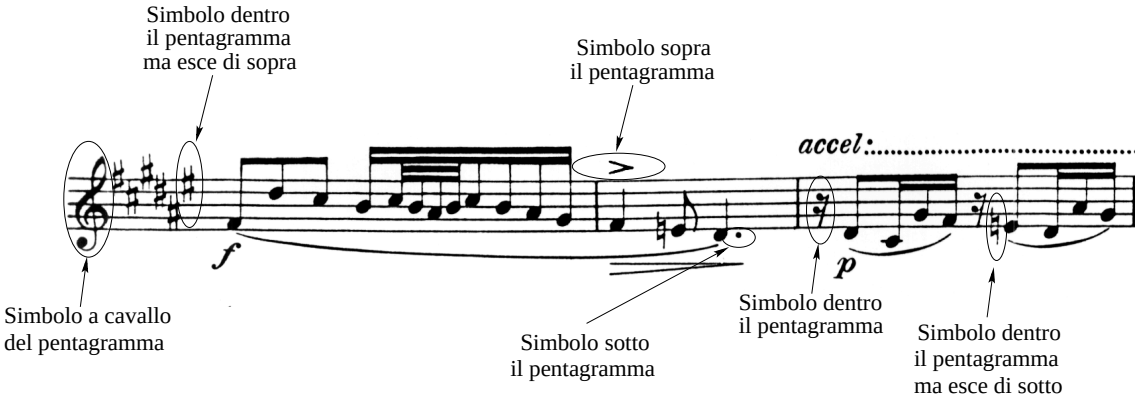


Figura 8.3: Esempio grafico del significato delle posizioni rispetto al pentagramma.

Sulla base delle posizioni individuate e la probabilità che il simbolo collegato possa occuparle, sono state definite delle regole che consentono di descrivere le proprietà posizionali. La struttura di una generica regola è definita nel modo seguente:

$$< nome > < P(pos_1) > \ ... \ < P(pos_6) > \tag{8.1}$$

dove:

- *nome* rappresenta il nome del simbolo collegato (es: S_REST16);
- $P(pos_i)$ esprime la probabilità che il simbolo ha di trovarsi nelle posizione definite in precedenza (**probabilità di posizione**).

L'insieme delle regole costituisce la **Grammatica posizionale**.

Nel caso di un simbolo collegato che rappresenti una pausa di semicroma, si può definire:

```
114 S_REST16 0.9 0 0 0 0.6 0.5
```

Infatti, la pausa nella maggioranza dei casi è posizionata all'interno del pentagramma, quindi il valore della probabilità per pos_1 è alto. Tuttavia esistono casi in cui la pausa è collocata in pos_5 o in pos_6, pertanto i relativi valori di probabilità hanno un valore più basso. Le altre posizioni non sono consentite e quindi si ha un valore nullo.

Se un simbolo presenta un valore uguale per ognuna delle sei probabilità di posizione, significa che non esiste una posizione fissa nel pentagramma, ma che tutte e sei sono possibili in uguale misura. Il riferimento numero ad inizio regola rappresenta un identificativo utilizzato nella fase implementativa.

Le regole della Grammatica posizionale consentono di selezionare tra i simboli collegati quello che la probabilità maggiore di rappresentare la primitiva grafica classificata nella classe relativa al token in ingresso. Di seguito è riporta l'insieme completo delle

regole della grammatica. Sono presenti regole per simboli direttamente collegati alle classi della rete neurale e simboli associati a primitive grafiche che non fanno parte dell'insieme dei simboli di base, ma che possono essere prodotte dalla segmentazione e successivamente associate ad una classe.

```
1 S_ACCENTO 0.5 0 0.9 0.9 0 0
2 S_BASS 0.9 0 0 0 0 0
3 S_BASSLOW 0.9 0 0 0 0 0
4 S_BEAM1 0.9 0 0.9 0.9 0.9 0.9
5 S_BEAM2 0.9 0 0.9 0.9 0.9 0.9
6 S_BEAM3 0.9 0 0.9 0.9 0.9 0.9
7 S_BEAM4 0.9 0 0.9 0.9 0.9 0.9
8 S_BEAM5 0.9 0 0.9 0.9 0.9 0.9
9 S_BOWDWN 0 0 0.9 0.9 0 0
10 S_BOWUP 0 0 0.9 0.9 0 0
11 S_BREATH 0 0 0.9 0 0.9 0
12 S_C 0.99 0 0 0 0 0
13 S_CC 0.99 0 0 0 0 0
14 S_CCLLOW 0 0 0 0 0 0.9
15 S_CCLOW 0 0 0 0 0 0.9
16 S_CHORD2V 0.9 0 0.9 0.9 0.9 0.9
17 S_CHORD3V 0.9 0 0.9 0.9 0.9 0.9
18 S_CLWA 0 0 0.9 0.9 0 0
19 S_CLWB 0 0 0.9 0.9 0 0
20 S_CLWC 0 0 0.9 0.9 0 0
21 S_CLWD 0 0 0.9 0.9 0 0
22 S_CLWE 0 0 0.9 0.9 0 0
23 S_CLWF 0 0 0.9 0.9 0 0
24 S_CLWG 0 0 0.9 0.9 0 0
25 S_CLWH 0 0 0.9 0.9 0 0
26 S_CLWI 0 0 0.9 0.9 0 0
27 S_CLWJ 0 0 0.9 0.9 0 0
28 S_CLWK 0 0 0.9 0.9 0 0
29 S_CLWL 0 0 0.9 0.9 0 0
30 S_CLWM 0 0 0.9 0.9 0 0
31 S_CLWN 0 0 0.9 0.9 0 0
32 S_CLWO 0 0 0.9 0.9 0 0
33 S_CLWP 0 0 0.9 0.9 0 0
34 S_CLWQ 0 0 0.9 0.9 0 0
35 S_CLWR 0 0 0.9 0.9 0 0
36 S_CLWS 0 0 0.9 0.9 0 0
37 S_CLWT 0 0 0.9 0.9 0 0
38 S_CLWU 0 0 0.9 0.9 0 0
39 S_CLWV 0 0 0.9 0.9 0 0
40 S_CLWW 0 0 0.9 0.9 0 0
41 S_CLWX 0 0 0.9 0.9 0 0
42 S_CLWY 0 0 0.9 0.9 0 0
43 S_CLWZ 0 0 0.9 0.9 0 0
44 S_COMMA 0 0 0.9 0 0.9 0
45 S_CRESCBEGIN 0 0 0 0.9 0 0
46 S_CRESCLINE 0 0 0 0.9 0 0
47 S_CSLASH 0.99 0 0 0 0 0
48 S_CUPA 0 0 0.9 0.9 0 0
49 S_CUPB 0 0 0.9 0.9 0 0
```

```
50 S_CUPC 0 0 0.9 0.9 0 0
51 S_CUPD 0 0 0.9 0.9 0 0
52 S_CUPE 0 0 0.9 0.9 0 0
53 S_CUPF 0 0 0.9 0.9 0 0
54 S_CUPG 0 0 0.9 0.9 0 0
55 S_CUPH 0 0 0.9 0.9 0 0
56 S_CUPI 0 0 0.9 0.9 0 0
57 S_CUPJ 0 0 0.9 0.9 0 0
58 S_CUPK 0 0 0.9 0.9 0 0
59 S_CUPL 0 0 0.9 0.9 0 0
60 S_CUPM 0 0 0.9 0.9 0 0
61 S_CUPN 0 0 0.9 0.9 0 0
62 S_CUPO 0 0 0.9 0.9 0 0
63 S_CUPP 0 0 0.9 0.9 0 0
64 S_CUPQ 0 0 0.9 0.9 0 0
65 S_CUPR 0 0 0.9 0.9 0 0
66 S_CUPS 0 0 0.9 0.9 0 0
67 S_CUPT 0 0 0.9 0.9 0 0
68 S_CUPU 0 0 0.9 0.9 0 0
69 S_CUPV 0 0 0.9 0.9 0 0
70 S_CUPW 0 0 0.9 0.9 0 0
71 S_CUPX 0 0 0.9 0.9 0 0
72 S_CUPY 0 0 0.9 0.9 0 0
73 S_CUPZ 0 0 0.9 0.9 0 0
74 S_DBARLINE 0.9 0 0 0 0 0
75 S_DECREEND 0 0 0 0.9 0 0
76 S_DECRESCLINE 0 0 0 0.9 0 0
77 S_DFLAT 0.9 0 0.9 0.9 0.9 0.9
78 S_DIGIT0 0.9 0 0.9 0.9 0 0
79 S_DIGIT1 0.9 0 0.9 0.9 0 0
80 S_DIGIT2 0.9 0 0.9 0.9 0 0
81 S_DIGIT3 0.9 0 0.9 0.9 0 0
82 S_DIGIT4 0.9 0 0.9 0.9 0 0
83 S_DIGIT5 0.9 0 0.9 0.9 0 0
84 S_DIGIT6 0.9 0 0.9 0.9 0 0
85 S_DIGIT7 0.9 0 0.9 0.9 0 0
86 S_DIGIT8 0.9 0 0.9 0.9 0 0
87 S_DIGIT9 0.9 0 0.9 0.9 0 0
88 S_DOTINCOMMA 0 0 0.9 0 0.9 0
89 S_DSHARP 0.9 0 0.9 0.9 0.9 0.9
90 S_ENOTEHEAD 0.9 0 0.9 0.9 0.9 0.9
91 S_FERMATA
92 S_FLAT 0.9 0 0.9 0.9 0.9 0.9
93 S_FLAT1Q 0.9 0 0.9 0.9 0.9 0.9
94 S_FLAT3Q 0.9 0 0.9 0.9 0.9 0.9
95 S_FNOTEHEAD 0.9 0 0.9 0.9 0.9 0.9
96 S_GREST 0.9 0 0 0 0 0
97 S_HOOK1DWN 0.9 0 0.9 0.9 0.9 0.9
98 S_HOOK1UP 0.9 0 0.9 0.9 0.9 0.9
99 S_HOOK2DWN 0.9 0 0.9 0.9 0.9 0.9
100 S_HOOK2UP 0.9 0 0.9 0.9 0.9 0.9
101 S_HOOK3DWN 0.9 0 0.9 0.9 0.9 0.9
102 S_HOOK3UP 0.9 0 0.9 0.9 0.9 0.9
103 S_HOOK4DWN 0.9 0 0.9 0.9 0.9 0.9
```

```
104 S_HOOK4UP 0.9 0 0.9 0.9 0.9 0.9
105 S_HOOK5DWN 0.9 0 0.9 0.9 0.9 0.9
106 S_HOOK5UP 0.9 0 0.9 0.9 0.9 0.9
107 S_MORDENT 0 0 0.9 0.9 0 0
108 S_NATURAL 0.9 0 0.9 0.9 0.9 0.9
109 S_NATURALDOWN 0.9 0 0.9 0.9 0.9 0.9
110 S_NATURALUP 0.9 0 0.9 0.9 0.9 0.9
111 S_PUNTVAL 0.9 0 0.9 0.9 0.9 0.9
112 S_QUARTERFLAT 0.9 0 0.9 0.9 0.9 0.9
113 S_QUARTERSHARP 0.9 0 0.9 0.9 0.9 0.9
114 S_REST16 0.9 0 0 0 0.6 0.5
115 S_REST2 0.9 0 0 0 0 0
116 S_REST32 0.9 0 0 0 0.9 0.9
117 S_REST4 0.9 0 0 0 0 0
118 S_REST8 0.9 0 0 0 0.5 0.5
119 S_SBARLINE 0.9 0 0 0 0 0
120 S_SHARP 0.9 0 0.9 0.9 0.9 0.9
121 S_SHARP1Q 0.9 0 0.9 0.9 0.9 0.9
122 S_SHARP3Q 0.9 0 0.9 0.9 0.9 0.9
123 S_SLUR 0 0 0.9 0.9 0 0
124 S_SLURSTAFLINE 0.9 0 0.9 0.9 0.9 0.9
125 S_STACCATO 0.9 0 0.9 0.9 0 0
126 S_STAFLINE 0.9 0 0.9 0.9 0.9 0.9
127 S_STAFLINESTEM 0.9 0 0.9 0.9 0.9 0.9
128 S_STAFLINESLUR 0.9 0 0.9 0.9 0.9 0.9
129 S_TDF 0.9 0 0.9 0.9 0.9 0.9
130 S_TDFF 0 0 0 0.9 0 0
131 S_TDFFF 0 0 0 0.9 0 0
132 S_TDFFFF 0 0 0 0.9 0 0
133 S_TDFFFFF 0 0 0 0.9 0 0
134 S_TDFFFFFF 0 0 0 0.9 0 0
135 S_TDFFP 0 0 0 0.9 0 0
136 S_TDFFPP 0 0 0 0.9 0 0
137 S_TDFP 0 0 0 0.9 0 0
138 S_TDFPP 0 0 0 0.9 0 0
139 S_TDFZ 0 0 0 0.9 0 0
140 S_TDMF 0 0 0 0.9 0 0
141 S_TDMP 0 0 0 0.9 0 0
142 S_TDP 0.9 0 0.9 0.9 0.9 0.9
143 S_TDPP 0 0 0 0.9 0 0
144 S_TDPPP 0 0 0 0.9 0 0
145 S_TDPPPP 0 0 0 0.9 0 0
146 S_TDPPPPP 0 0 0 0.9 0 0
147 S_TDPPPPPP 0 0 0 0.9 0 0
148 S_TDRF 0 0 0 0.9 0 0
149 S_TDRFZ 0 0 0 0.9 0 0
150 S_TDSF 0 0 0 0.9 0 0
151 S_TDSFF 0 0 0 0.9 0 0
152 S_TDSFFF  0 0 0.9 0 0
153 S_TDSFFP 0 0 0 0.9 0 0
154 S_TDSFFZ 0 0 0 0.9 0 0
155 S_TDSFP 0 0 0 0.9 0 0
156 S_TDSFPP 0 0 0 0.9 0 0
157 S_TDSFZ 0 0 0 0.9 0 0
```

```
158 S_TDSP 0 0 0 0.9 0 0
159 S_TENOR 0 0 0 0 0.9 0
160 S_TENUTO 0 0 0 0 0 0
161 S_TREBLE 0 0.9 0 0 0 0
162 S_TRILL 0 0 0.9 0.5 0 0
163 S_TURN  0.9 0 0.9 0.9 0.9 0.9
164 S_W2NOTE 0.9 0 0.9 0.9 0.9 0.9
165 S_W2REST 0.9 0 0 0 0 0
166 S_WREST 0.9 0 0 0 0 0
167 S_TRILLWAVE 0 0 0.9 0.9 0 0
168 S_CHORD2H 0.9 0 0.9 0.9 0.9 0.9
169 S_CORONA 0 0 0.9 0 0 0
170 S_WHOLENOTE 0.9 0 0.9 0.9 0.9 0.9
171 S_REST64 0.9 0 0 0 0.4 0.4
172 S_HALFREST8 0.9 0 0.5 0 0.5 0.5
173 S_DIGIT12 0.9 0 0.9 0.9 0 0
174 S_DIGIT16 0.9 0 0.9 0.9 0 0
175 S_CHORDE2V 0.9 0 0.9 0.9 0.9 0.9
176 S_CHORDE3V 0.9 0 0.9 0.9 0.9 0.9
177 S_C_SX 0.9 0 0 0 0 0
178 S_C_DX 0.9 0 0 0 0 0
```

8.2 Determinazione dei simboli collegati e delle probabilità di posizione

In questa sezione è riportato un esempio sull'utilizzo della Tabella delle Relazioni e della Grammatica posizionale. Gli ingressi al ricostruttore relativi ad una battuta sono scanditi sequenzialmente partendo dal primo fino all'ultimo, escludendo dal procedimento tutte le righe in ingresso che definiscono l'aggiornamento delle coordinate del pentagramma, le indicazioni strutturali e di fine pentagramma. Per ogni ingresso, si determina l'insieme dei simboli collegati alla classe, cui è stata associata la primitiva grafica, definito dalla Tabella delle Relazioni. Per ciascuno dei simboli collegati si determinano le relative probabilità di posizione utilizzando la grammatica posizionale. A ciascun simbolo collegato è associato il valore di probabilità della posizione (pos_i) determinata dalle coordinate del bounding box relativamente a quelle del pentagramma. Si consideri l'ingresso nella forma seguente:

```
STAFF5 0 10 20 30 40 50
    .
    .
    .
CLEFALTO 20 10 50 40 0,765
    .
    .
    .
```

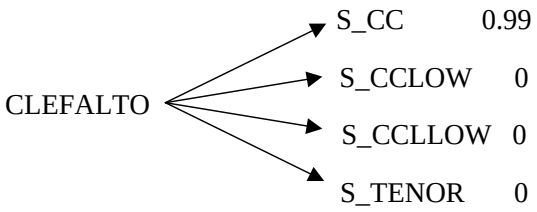


Figura 8.4: Risultato finale del processo di determinazione dei simboli collegati e delle relative probabilità di posizione.

Memorizzate le coordinate del pentagramma, che derivano dalla prima riga in ingresso, in corrispondenza del token CLEFALTO, la Tabella delle Relazioni genera il seguente insieme di simboli collegati che rappresentano rispettivamente:

- S_CC la chiave di Contralto;
- S_CCLOW la chiave di Mezzo-Soprano;
- S_CCLLOW la chiave di Soprano;
- S_TENOR la chiave di Tenore.

Per ciascuno di essi, si hanno i seguenti valori di probabilità di posizione:

```
13 S_CC 0.99 0 0 0 0 0
14 S_CCLLOW 0 0 0 0 0 0.9
15 S_CCLOW 0 0 0 0 0 0.9
159 S_TENOR 0 0 0 0 0.9 0
```

Per il simbolo S_CC, ad esempio, le probabilità di posizione sono rispettivamente:

0.99 che sia interno al pentagramma (pos_1);

0 che sia a cavallo del pentagramma (pos_2);

0 che sia sopra il pentagramma (pos_3);

0 che sia sotto il pentagramma (pos_4);

0 che sia all'interno e uscendo in parte al di sopra del pentagramma (pos_5);

0 che sia all'interno e uscendo in parte al di sotto del pentagramma (pos_6);

Considerando il valore dell'ordinata (Y=10) e dell'altezza (ALTEZZA=40) del bounding box dell'elemento, si ha che il token considerato è posizionato all'interno del pentagramma. Si ottiene che la probabilità di posizione associato al simbolo S_CC è pari a 0.99.

Procedendo allo stesso modo per i simboli rimanenti si hanno i valori mostrati nella Figura 8.4.

Valutando i valori associati ai simboli collegati, il simbolo S_CC è quello che ha la probabilità maggiore di rappresentare il token in ingresso.

Capitolo 9

Aggregazione

In questo capitolo sono descritti la grammatica musicale ed il modulo di aggregazione. Quest'ultimo realizza la fase di ricostruzione finale nella quale i simboli di base sono assemblati per generare figure musicali semplici e complesse, insieme alle relazioni strutturali individuate dalla fase di segmentazione. Il risultato finale è la ricostruzione dello spartito iniziale.

9.1 La Grammatica Musicale

Il linguaggio musicale, come gli altri linguaggi, è soggetto a regole, sia di tipo sintattico che semantico. In particolare, le regole di tipo sintattico hanno lo scopo di individuare le caratteristiche, come la forma e la posizione che un determinato simbolo deve possedere per poter essere ritenuto corretto, mentre le regole di tipo semantico determinano il significato di una nota (altezza e durata), della battuta (consistenza temporale), e più in generale di uno spartito.

Una parte delle regole relative al linguaggio musicale definiscono il modo di scrivere la musica e sono individuate in questo contesto con il termine **regole grafiche**. Queste regole, che sono esclusivamente di tipo sintattico, riguardano sia la struttura dei simboli appartenenti alla notazione musicale, come la struttura di una nota, sia il modo in cui tali simboli sono collegati fra loro, ad esempio dove deve essere posizionata l'alterazione di una nota.

La grammatica musicale utilizzata dal modulo di aggregazione è costituita esclusivamente da regole grafiche. Tali regole sono state separate in due principali gruppi quali:

- regole **verticali**;

- regole **orizzontali**.

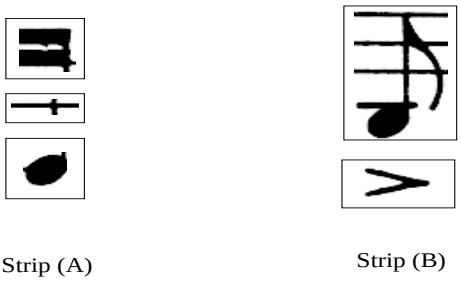


Figura 9.1: Esempi di strip che devono usare regole grafiche verticali.

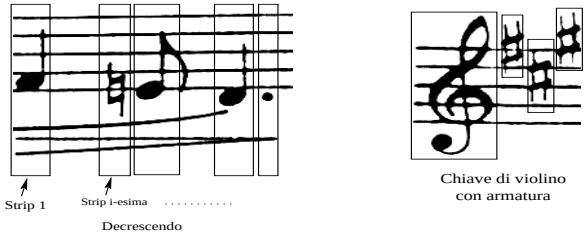


Figura 9.2: Esempio di simboli con collegamenti orizzontali.

L'insieme delle **regole verticali** è costituito da tutte le regole grafiche che interessano i simboli musicali, elementari o composti, *appartenenti alla stessa strip*. Nella Figura 9.1 sono riportati due esempi nei quali è necessario l'utilizzo delle regole verticali. In particolare, le regole che interessano la strip (A), relativamente alle primitive grafiche ottenute con la segmentazione di una semicroma, sono quelle che definiscono la struttura di una nota. Le regole che interessano la strip (B), relativamente ad una strip contenente una croma e un accento, sono quelle che consentono l'aggregazione fra simboli musicali. Per quanto riguarda l'insieme delle **regole orizzontali**, esso è costituito da tutte le regole grafiche che interessano i simboli musicali, elementari o composti, *appartenenti a strip diverse*. Nella Figura 9.2 sono riportati due esempi nei quali è necessario l'utilizzo delle regole orizzontali. In dettaglio, si osserva che le regole orizzontali sono usate per i simboli musicali che si estendono orizzontalmente e che coinvolgono più strip, come il decrescendo del primo disegno, e per i simboli che hanno un'influenza reciproca, come le alterazioni appartenenti all'armatura di una chiave e la chiave stessa oppure l'alterazione associata ad una nota.

I motivi che hanno condotto alla definizione di due insiemi disgiunti per le regole grafiche sono legati sia al modo di operare del segmentatore sia alla struttura grafica del linguaggio musicale. Per quanto concerne la segmentazione (si veda il Capitolo 4), si ricorda che l'estrazione delle primitive grafiche avviene nel livello 2 ed è ottenuta a partire dalle strip determinate nel livello 1. Questo modo di operare richiede delle regole di ricostruzione che operino sugli elementi appartenenti ad una sola strip, in modo da

consentire il ricongiungimento e ottenere il simbolo musicale di partenza. Un esempio è dato dalla strip (A) di Figura 9.1 nella quale le regole verticali devono essere in grado di ricongiungere le tre primitive grafiche presenti, in modo da ottenere la semicroma di partenza. Per i simboli musicali, come la legatura, il decrescendo ed il crescendo, per i quali la segmentazione genera delle primitive grafiche che appartengono a strip diverse, il ricongiungimento può avvenire con delle regole che operino attraverso strip diverse. Infine, per quanto concerne le regole di scrittura della musica, i simboli musicali possono essere collegati fra loro sia se appartengono alla stessa strip, come nel caso della strip (B) di Figura 9.1, che a strip adiacenti, come nel secondo esempio della Figura 9.2.

9.1.1 Struttura delle regole

In questo paragrafo è analizzata la struttura delle regole e sono riportate le motivazioni che hanno condotto alla loro definizione. La struttura delle regole deve soddisfare alcuni requisiti fondamentali per una grammatica. La grammatica deve essere insensibile a modifiche e/o ampliamenti delle regole che la compongono, ovvero possedere le caratteristiche di *modularità* ed *flessibilità*, e godere della proprietà di indipendenza dall'algoritmo risolutivo del problema.

Un primo ostacolo da superare è quello di rendere la struttura delle regole indipendente dal fatto che appartengano all'insieme delle regole verticali o orizzontali. In questo modo la formalizzazione è unica per i due insiemi. Questo problema è stato superato anteponendo, alla regola stessa, la tipologia dell'insieme di appartenenza e che è definita **Intestazione della regola**. Quindi prima della definizione di una regola è presente il termine **VRULE** per indicare l'insieme delle regole verticali, oppure il termine **HRULE** per quelle orizzontali. Si può quindi definire la struttura di una generica regola nel seguente modo:

Intestazione della Regola
Lista dei Simboli della regola(condizioni)
==>Simbolo Musicale Risultante(assegnazioni);

Lista dei Simboli della regola: in questa sezione, racchiusa fra l'intestazione e il simbolo grafico ==> (risultante), sono posizionati i simboli necessari per l'applicazione della regola. Per meglio comprendere la forma assunta dalle regole sono riportati alcuni esempi, estratti dall'archivio della grammatica musicale, con accanto la struttura:

```
VRULE                                          INTESTAZIONE
  ::S_SHARP(altezzaGreat>1.7)          ------>     SIMBOLO
  ==> ALTERAZIONE(isDiesis);                     ==> RISULTANTE(Assegnazioni);

VRULE                                          INTESTAZIONE
  S_STAFLINE                                     SIMBOLO
  S_STAFLINE                           ------>   SIMBOLO
  ::S_ENOTEHEAD(altezzaGreat>1,notBeamed)        SIMBOLO CHIAVE
  ==> NOTE(stem:=Up,head:=Empty,durate:=2);      ==> RISULTANTE(Assegnazioni);

VRULE                                          INTESTAZIONE
  ::S_FNOTEHEAD(altezzaGreat>1)                  SIMBOLO CHIAVE
  [S_STAFLINE]                         ------>   SIMBOLO
  S_BEAM1(altezzaGreat>0.52,altezzaLess<1.3)     SIMBOLO OPZIONALE
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=8,inBeam); ==> RISULTANTE(Assegnazioni);

HRULE                                          INTESTAZIONE
  ::S_SLUR(dyMax=0.4)                  ------>   SIMBOLO CHIAVE(Condizioni)
  S_STAFLINESLUR                                 SIMBOLO
  ==> LINE;                                      ==> RISULTANTE;
```

Dagli esempi si osserva che la sezione in esame è formata da uno o più simboli che rappresentano, nell'ordine in cui sono scritti, la sequenza necessaria per la costruzione del simbolo rappresentato dalla risultante. Alcuni simboli appartenenti alla sezione in esame hanno dei dati racchiusi fra parentesi tonde. Tali dati, individuati con il termine **Condizioni** e separati fra loro con una virgola, rappresentano le condizioni aggiuntive che il simbolo deve soddisfare congiuntamente (AND) per essere ritenuto corretto. La struttura delle condizioni è del tipo:

attributo = *valore*.

Nel Paragrafo 9.13.5 è riportato il significato degli attributi e dei valori utilizzati all'interno della grammatica musicale.

Tornando ad esaminare la sezione della lista dei simboli della regola si osserva, sempre con riferimento agli esempi, che alcuni simboli sono racchiusi nel grafema "[ ]". Questa indicazione definisce il numero di volte che un simbolo, racchiuso al suo interno, deve essere presente per poter applicare la regola, imponendo il numero minimo uguale a 0 ed il massimo uguale ad infinito. La sua funzione è quella di fornire un meccanismo di opzionalità per il simbolo coinvolto e di risolvere il problema legato al fatto che un simbolo musicale può essere composto da un numero variabile di simboli elementari dello stesso tipo. Un esempio è dato da una nota che può avere un gambo di lunghezza variabile a seconda del contesto in cui si trova e come riportato nella Figura 9.3. Si osserva che il numero di linee di pentagramma che intersecano il gambo può essere variabile, pertanto, il meccanismo di opzionalità consente di ridurre il numero di regole in grado per gestire queste situazioni.

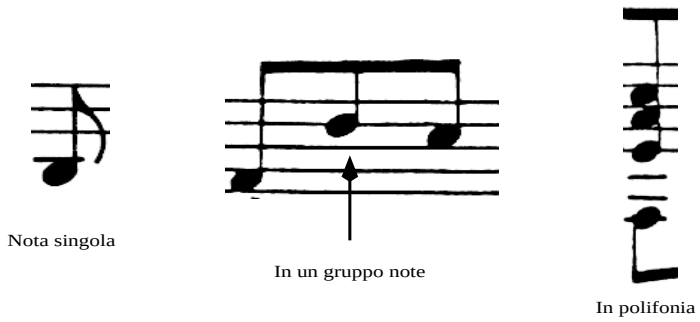


Figura 9.3: Esempio di croma in tre diversi contesti.

L'ultimo grafema da analizzare, per quanto riguarda la sezione in esame, è dato da "::". Questo simbolo grafico è utilizzato per individuare, fra tutti i simboli di una regola, quello identificativo della regola stessa. Tale simbolo prende il nome di **chiave della regola** ed è utilizzato per indicizzare le regole all'interno dell'archivio. Si ha quindi che:

Definizione 1 *Il simbolo identificativo di una regola viene chiamato* ***"chiave della regola"****.*

L'introduzione di una chiave per ogni regola appartenente alla grammatica musicale è necessaria per velocizzare il modulo di aggregazione. Infatti, una volta caricati i dati in ingresso e stabilite le relazioni fra gli elementi acquisiti e i simboli elementari, l'applicazione di una regola richiede la ricerca di tutti quei vincoli che contengono un determinato simbolo. Da qui si evince che la struttura della grammatica deve essere creata in modo da permettere l'individuazione dell'insieme di regole che hanno al loro interno uno specifico simbolo.

Simbolo Musicale Risultante: questa sezione è rappresentata dall'indicazione che segue il grafema "==>" e che rappresenta il risultato dell'aggregazione col quale possono essere sostituiti i simboli appartenenti alla regola. Il simbolo musicale risultante può avere dei dati racchiusi fra parentesi tonde, così come visto per i simboli della regola. Tali dati, individuati con il termine **Assegnazioni** e separati fra loro con una virgola, rappresentano le caratteristiche del simbolo musicale risultante. La forma di tali assegnazioni è del tipo:

attributo **:=** *valore.*

Per il significato degli attributi e dei valori utilizzati per le assegnazioni si rimanda al Paragrafo 9.13.5.

Per quanto riguarda le caratteristiche del simbolo risultante queste sono definite, oltre che dalle assegnazioni della risultante stessa, anche dalle caratteristiche associate ad ogni singolo simbolo utilizzato per l'applicazione della regola. Si ha quindi che la

risultante della regola **eredita tutte le caratteristiche dei simboli utilizzati per l'applicazione della regola, con il relativo valore**.

Si consideri, ad esempio, la regola:

```
VRULE
   SIMBOLO
   ::NOTE(stem=Up)
   [S_STAFLINE]
   ==> NOTE;
```

il cui significato, per quanto riguarda la *Lista dei simboli della regola*, è dato da:

1. una generica NOTA con l'unico vincolo dato dalla presenza del gambo la cui direzione deve essere verso l'alto (*stem=Up*);
2. un generico simbolo, che può essere rappresentato da una corona, da un accento o da un punto;
3. dei segmenti di pentagramma, la cui presenza è resa opzionale dal simbolo grafico "[ ]".

Per quanto riguarda la *risultante*, è generata:

1. una NOTA che non inserisce nessuna assegnazione particolare, infatti la risultante non presenta nessuna assegnazione aggiuntiva.

Nel caso specifico di una strip formata dai seguenti simboli musicali:

1. SIMBOLO(isCorona);
2. NOTE(stem=Up,head=Fill,durate=4).

che corrispondono rispettivamente ad:

1. una corona (*isCorona*);
2. una nota con il gambo rivolto verso l'alto (*stem=Up*), la testa piena (*head=Fill*) e la durata pari ad un 1/4 (*durate=4*).

si ha che il simbolo risultante dall'applicazione della regola, è dato da:

NOTE(stem:=Up,head:=Fill,durate:=4,isCorona)

ovvero una nota le cui caratteristiche sono date dalla somma delle caratteristiche dei simboli che costituiscono la regola.

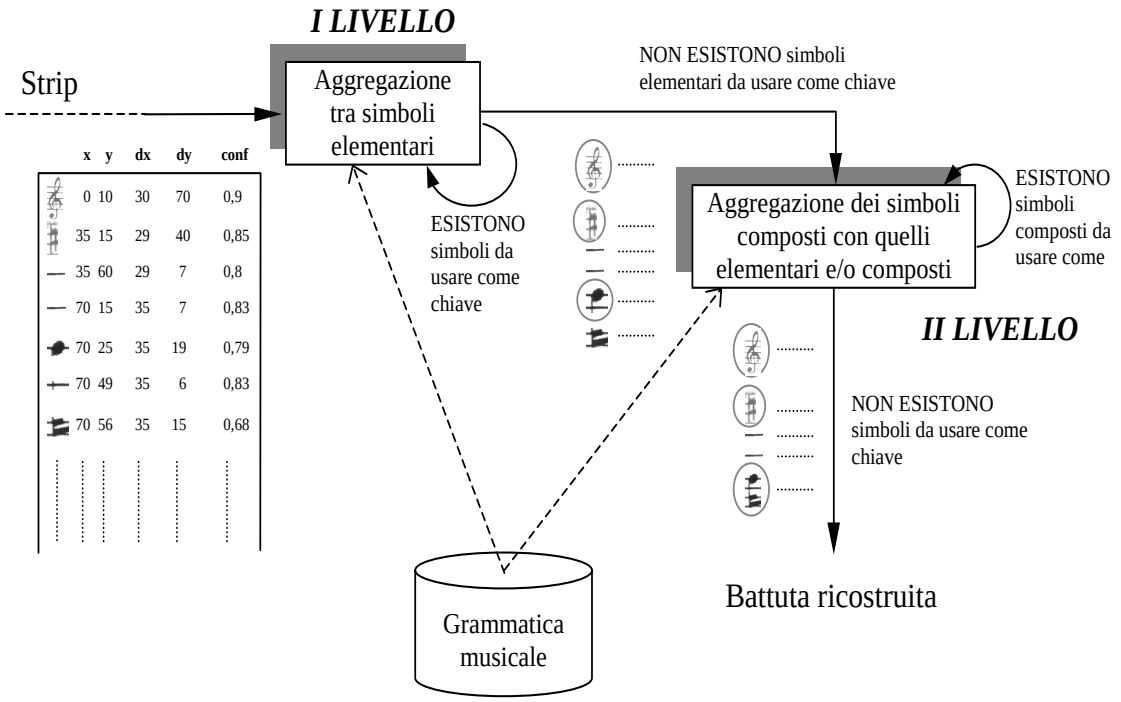


Figura 9.4: Schema della fase di aggregazione.

9.2 Il modulo di aggregazione

Il modulo di aggregazione è incaricato di assemblare i simboli all'interno di una strip per ricostruire i simboli musicali. Il processo di aggregazione è suddiviso in due livelli come schematizzato nella Figura 9.4 ed implementa un modello gerarchico basato sull'applicazione di due livelli di aggregazione agli elementi in ingresso.

Aggregazione di Livello I – Le aggregazioni sono effettuate in modo iterativo applicando le regole verticali. Nell'aggregazione di Livello I, sono possibili tutte le aggregazioni che generano simboli composti a partire da una strip, i simboli non aggregati residui sono riconsiderati nell'aggregazione del Livello II. L'aggregazione di Livello I termina quando nella strip non vi sono più simboli da aggregare o simboli non ancora aggregati che siano la chiave di una regola.

Aggregazione di Livello II – Il secondo livello applica le regole, sia verticali che orizzontali. L'applicazione delle regole verticali può avvenire tra simboli composti ed elementi della strip non ancora usati in aggregazioni precedenti oppure tra simboli composti inclusi nella stessa strip. L'applicazione delle regole orizzontali consente l'aggregazione tra simboli composti e/o elementi non ancora aggregati di una strip con simboli composti

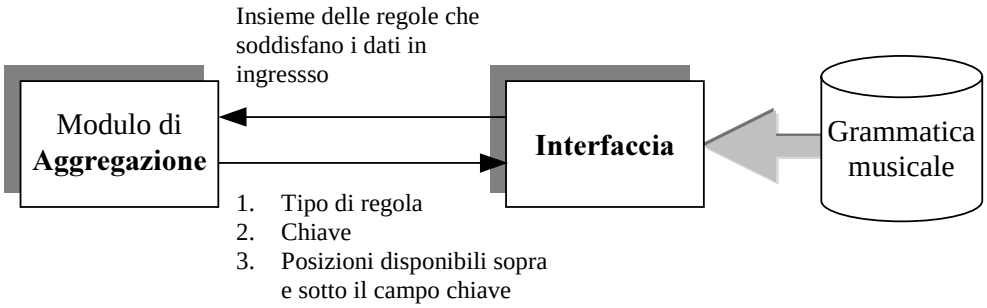


Figura 9.5: Schema di utilizzo della grammatica musicale.

e/o elementi non ancora aggregati di una strip adiacente. L'uscita è costituita da simboli composti o simboli musicali completi.

Il ricostruttore consente la generazione di simboli via via più complessi. Questo deriva dal fatto che il secondo livello di regole ha la possibilità di aggregare fra loro i simboli composti, ottenuti al primo livello, ed i simboli elementari, consentendo di produrre un numero maggiore di combinazioni fra i simboli musicali e di determinare molte alternative nel caso in cui si verifichi che la battuta ricostruita non sia corretta. L'aggregazione progressiva dei simboli è messa in evidenza anche nella Figura 9.4 dove i risultati prodotti dai due livelli sono stati cerchiati. In particolare, osservando gli ingressi e le uscite dei livelli di aggregazione, si nota che il I livello, con riferimento alle primitive relative alla semicroma dell'esempio, può operare un'aggregazione fra la testa della nota e il taglio addizionale (equivalente dal punto di vista grafico ad una linea del pentagramma), creando quindi una nota di semiminima, mentre il secondo livello di aggregazione può operare l'unione fra la semiminima e le travi (beam), creando quindi la semicroma in ingresso. Il processo di aggregazione si arresta quando non esistono simboli da usare come chiave di una regola. Il modulo di aggregazione applica le regole contenute nella grammatica musicale. A livello di architettura, è stato introdotto il modulo **Interfaccia** (si veda Figura 9.5). Il compito dell'Interfaccia è quello di restituire l'insieme delle regole richieste per effettuare le aggregazioni.

9.3 La strip e l'aggregazione verticale

La struttura di una strip nella fase di iniziale dell'aggregazione è descritta nel modo seguente:

$$E_s = \begin{cases} X_1(e_1) = \{(x_1, p_1), ..., (x_n, p_n)\} \\ ... \\ X_p(e_p) = \{(x_1, p_1), ..., (x_m, p_m)\} \end{cases} \tag{9.1}$$

dove:

- E_s indica la strip come insieme di elementi;
- e_i indica l'i-esimo elemento di E_s;
- $X_i(e_i)$ rappresenta l'insieme dei simboli collegati all'i-esimo elemento;
- (x, p) indica la coppia simbolo collegato e probabilità di posizione.

Ciascun elemento della strip ha associato il proprio insieme dei simboli collegati definito nella Tabella delle Relazioni. Ad ogni simbolo collegato è associata la probabilità associata alla posizione dell'elemento nel pentagramma e stabilita dalla Grammatica posizionale. Analizzando la struttura della strip, si osserva che ogni elemento è interpretabile in modi differenti quanti sono i simboli collegati. Se si considerano le possibili combinazioni, si ottiene un insieme di configurazioni. Tra queste, occorre individuare quella che a livello probabilistico descrive l'informazione che la strip contiene.

La configurazione di partenza ritenuta migliore è quella che per ogni elemento considera il simbolo collegato più probabile, definito come **simbolo a massima probabilità di verosimiglianza** e indicato con X_{Max}. La configurazione di partenza è rappresentata nel modo seguente:

$$E_s^0 = \begin{cases} X_{Max}^{(1)} \\ ... \\ X_{Max}^{(p)} \end{cases} \tag{9.2}$$

Individuata la configurazione E_s^0, per conoscere quale regola applicare, quali e quanti elementi della strip sono coinvolti è necessario determinare il simbolo nella strip da utilizzare come chiave della regola. La condizione migliore di aggregazione è coinvolgere tutti i simboli della configurazione in una volta sola. Questo non è sempre possibile, poiché non è garantito che la configurazione scelta individui una regola che sia in grado di aggregare tutti gli elementi, piuttosto è possibile che solo un sottoinsieme di simboli della strip sia aggregabile dalla regola, che può generare o un simbolo musicale completo o un simbolo composto. Quindi, alcuni simboli possono essere coinvolti in un'aggregazione mentre gli altri rimangono inalterati. Questo consente di generare un processo iterativo che aggrega per livelli successivi i simboli composti e i simboli non ancora aggregati in modo da utilizzare tutta l'informazione presente nella strip. Ad ogni aggregazione segue una nuova configurazione della strip nella quale si possono avere i simboli composti (come insieme di simboli aggregati, ma visto a tutti gli effetti come un simbolo) e simboli non aggregati. I primi sono identificati dal nome presente nella risultante della regola che li ha aggregati mentre gli ultimi coincidono ancora con i simboli più probabili della configurazione E_s^0. Nelle Figure 9.6, 9.7 e 9.8 sono riportati tre esempi di aggregazione verticale, mentre una

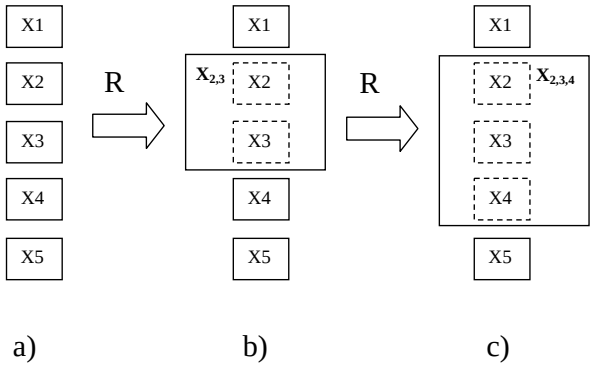


Figura 9.6: Aggregazioni successive a partire dalla strip di partenza (a). La prima regola genera un simbolo composto (b), mentre la seconda aggrega un elemento al simbolo composto (c).

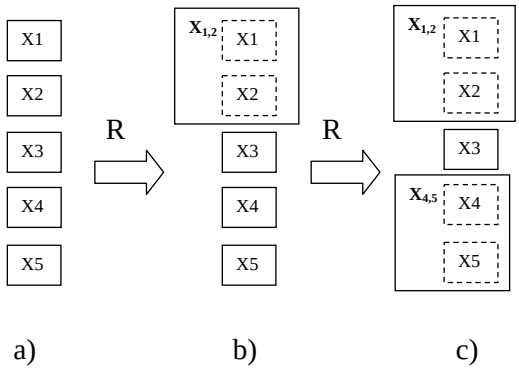


Figura 9.7: Aggregazioni successive a partire dalla strip di partenza (a). La prima regola genera un simbolo composto (b), mentre la seconda genera un nuovo simbolo composto (c).

possibile configurazione della strip della Figura 9.6.b è rappresentata nel modo seguente:

$$E_s^i = \begin{cases} X_{Max}^{(1)} \\ \hat{X} = X_{2,3} = \{X_{Max}^{(2)}, X_{Max}^{(3)}\} \\ X_{Max}^{(4)} \\ ... \\ X_{Max}^{(p)} \end{cases} \tag{9.3}$$

Dove il simbolo $\hat{X}$ rappresenta il simbolo composto ottenuto dall'aggregazione e assume il nome della risultante della regola utilizzata.

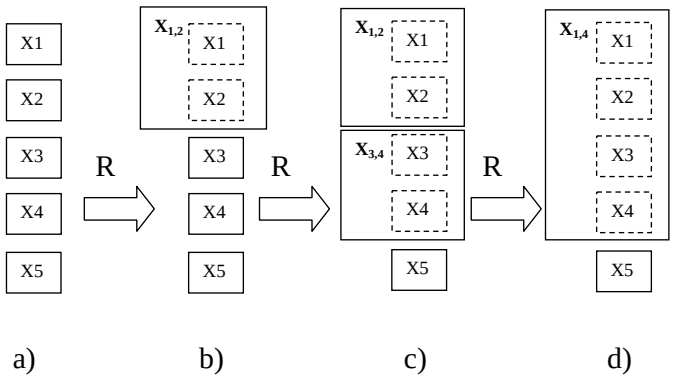


Figura 9.8: Aggregazioni successive a partire dalla strip di partenza (a). La prima regola genera un simbolo composto (b), la seconda genera un nuovo simbolo composto (c) infine la terza li assembla (d).

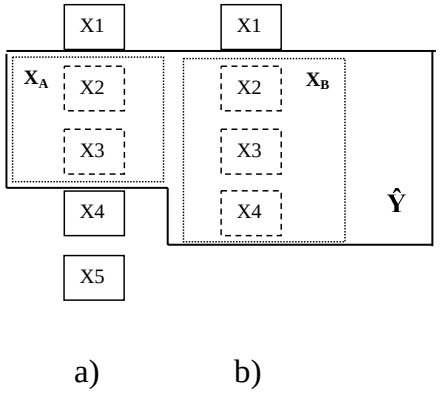


Figura 9.9: Aggregazione orizzontale tra i simboli composti della strip (a) e la strip (b).

9.4 La strip e l'aggregazione orizzontale

Le considerazioni svolte per l'aggregazione verticale valgono anche per l'aggregazione orizzontale definita per due strip adiacenti. Per come sono state definite le regole è possibile vincolare l'aggregazione definendo delle condizioni sulla distanza tra i simboli. Le strip coinvolte, possono presentare una configurazione costituita da simboli composti (generati dalla regoli verticali) e simboli non aggregati. L'aggregazione è ottenuta operando trasversalmente sui simboli composti o non ancora aggregati. Nell'ordine delle aggregazioni, la ricostruzione delle relazioni orizzontali è l'ultima fase del processo di ricostruzione. Nelle Figure 9.9, 9.10 e 9.11 sono riportati alcune situazioni di aggregabilità tra simboli. Come per le aggregazioni verticali il simbolo ottenuto prende il nome della risultante della regola.

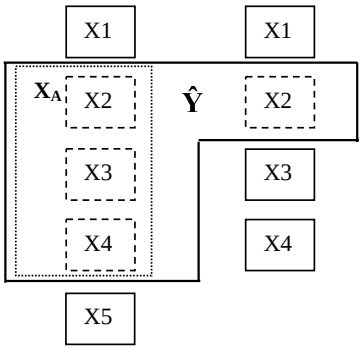


Figura 9.10: Aggregazione orizzontale tra il simbolo composto della strip (a) e il simbolo non aggregato della strip (b).

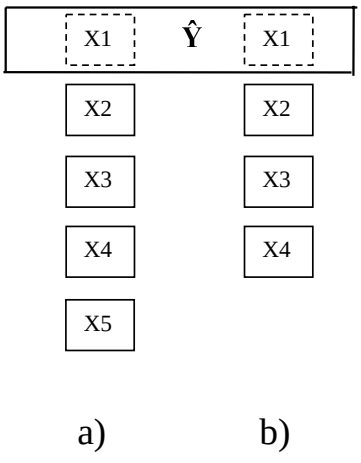


Figura 9.11: Aggregazione orizzontale tra il simbolo non aggregato della strip (a) e il simbolo non aggregato della strip (b).

9.5 Tipi di aggregazione

Sono state definite due modalità di aggegazione. Entrambe sono utilizzate nel procedimento iterativo di assemblaggio dei simboli e hanno l'obiettivo di ricostruire quanta più informazione musicale possibile nascosta all'interno della strip.

Aggregazione senza cambio di simboli – Individuato il simbolo chiave e la regola determinata dalla configurazione della strip, se la lista dei simboli della regola è soddisfatta allora l'aggregazione coinvolge i simboli della configurazione che verificano la lista e lascia inalterati i simboli non coinvolti. L'aggregazione descritta prende il nome di *Aggregazione senza cambio di simboli* o *Aggregazione senza forzatura*.

Aggregazione con sostituzione di simboli – Se non è possibile individuare una regola che consenta un'aggregazione senza cambio di simboli e nella strip sono presenti ancora tutti i simboli o un loro sottoinsieme, occorre un metodo alternativo che provi a estrarre l'informazione contenuta nei simboli e quindi tenti di aggregare, piuttosto che dichiarare conclusa l'aggregazione. Si supponga che esista una regola la cui lista dei simboli differisca per uno o più simboli rispetto alla configurazione iniziale o ad un sottoinsieme della strip. Si ipotizzi che in corrispondenza del simbolo non valido, l'insieme dei simboli collegati contenga il simbolo richiesto dalla regola. Questo condizione suggerisce di effettuare una sostituzione del simbolo o dei simboli in questione. In questa modo si realizza un cambio di configurazione che consente l'applicazione della regola e il recupero dell'informazione. Anche in questo caso l'aggregazione coinvolge i simboli della configurazione che verificano la lista e lascia inalterati i simboli non coinvolti. Questa procedura è definita *Aggregazione con sostituzione di simboli* o *Aggregazione con forzatura*.

9.6 Definizione dello stato S

Dalle considerazioni effettuate sul processo di aggregazione, si osserva che i simboli di partenza se sono usati da una regola concorrono alla generazione di un nuovo simbolo di livello più alto. Il passaggio del simbolo più probabile all'insieme dei simboli che costituiscono il simbolo ottenuto con l'applicazione della regola, deve essere tracciato in modo che questo non sia utilizzato successivamente per generare nuovi simboli composti con i simboli probabili non ancora aggregati. Il tracciamento delle operazioni che ogni simbolo della configurazione di partenza subisce nella fase di aggregazione è effettuato attraverso la definizione di uno stato S per ciascuno dei simboli di partenza. L'introduzione dello stato consente di:

- tenere in considerazione la condizione di partenza individuata dalla configurazione ritenuta migliore (E_s^0) e dalla quale parte la procedura di aggregazione;

- definire l'importanza a livello probabilistico per ciascun simbolo collegato più probabile e costruire un ordine per la scelta del simbolo chiave per la regola di aggregazione;
- mantenere traccia della tipologia di aggregazione che coinvolge il simbolo e quindi l'evoluzione dell'aggregazione.

9.6.1 Lo stato iniziale S_0 e il simbolo a massima probabilità di verosimiglianza

Lo stato S_0 è caratteristico di ciascun simbolo collegato appartenente alla strip da aggregare. Sia:

- e_i^* il simbolo collegato ottenuto da e_i con l'aggiunta di "S_"
- x^* il simbolo collegato con la probabilità di posizione più alta (in caso di più simboli si considera il primo nell'ordine dell'insieme dei simboli collegati)
- $|A|$ l'operatore che restituisce la cardinalità dell'insieme A (numero di elementi contenuti)

Le condizioni che consentono di definire lo stato di un simbolo collegato della strip da aggregare e la determinazione del simbolo a massima probabilità di verosimiglianza sono le seguenti (si veda Figura 9.12):

Condizione di certezza (C_1):

$$|X_i(e_i)| = 1 \Rightarrow S = 1 \wedge X_{Max}^{(i)} = x_1 \tag{9.4}$$

Indica che la classe di uscita della rete neurale ha un unico simbolo collegato. Un esempio è dato dal token BIGLINE5 che, come si vede dalla Tabella delle Relazioni, può essere relazionata solo con il simbolo S_BEAM5. La scelta del simbolo a massima probabilità è obbligata all'unico simbolo collegato al token in esame.

Condizione di certezza probabile (C_0):

$$|X_i(e_i)| > 1 \wedge (conf(e_i) > soglia) \wedge (\exists\, x \in X_i(e_i) \mid x = e_i^*) \Rightarrow S = 0 \wedge X_{Max}^{(i)} = e_i^* \tag{9.5}$$

La condizione esprime che l'elemento in esame ha un proprio simbolo collegato e, allo stesso tempo, che il livello di confidenza col quale è stato classificato dalla rete neurale è maggiore di un prefissato valore di *soglia*. Se si considera, ad esempio, che il token in

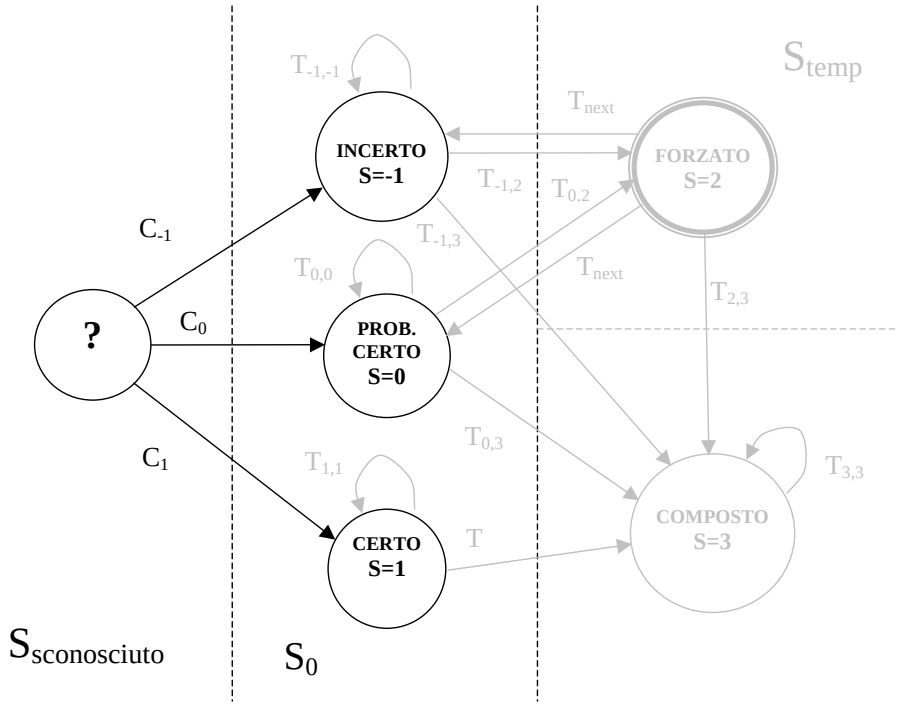


Figura 9.12: Diagramma degli stati: lo stato iniziale S_0

esame sia FNOTEHEAD e che il suo livello di confidenza sia maggiore di 0,7 (0,7 individua il valore della soglia) allora, esistendo il simbolo S_FNOTEHEAD, tale token ha uno stato iniziale pari a 0. La scelta del simbolo a massima probabilità è obbligata all'unico simbolo collegato al token in esame.

Condizione di incertezza (C_{-1}):

$$|X_i(e_i)| > 1 \wedge (conf(e_i) \leq soglia) \vee (\forall\, x \in X_i(e_i)\; x \neq e_i^*) \Rightarrow S = -1 \wedge X_{Max}^{(i)} = x^* \tag{9.6}$$

Indica che il token in esame non ha un proprio simbolo collegato, ovvero non esiste il simbolo corrispondente al nome del token con aggiunto il prefisso "S_", e nello stesso tempo, ha più simboli collegati. Se ad esempio si considera il token CLEFBASS, dall'analisi della tabella delle relazioni, si ha che questo può essere confuso con i simboli S_BASS e S_BASSLOW e pertanto non esiste il simbolo S_CLEFBASS. Sono inclusi i casi in cui non valgono le condizioni per gli stati precedenti.

Una volta stabilito il valore dello stato iniziale, il token in ingresso è completamente caratterizzato sia dal punto di vista simbolico che probabilistico.

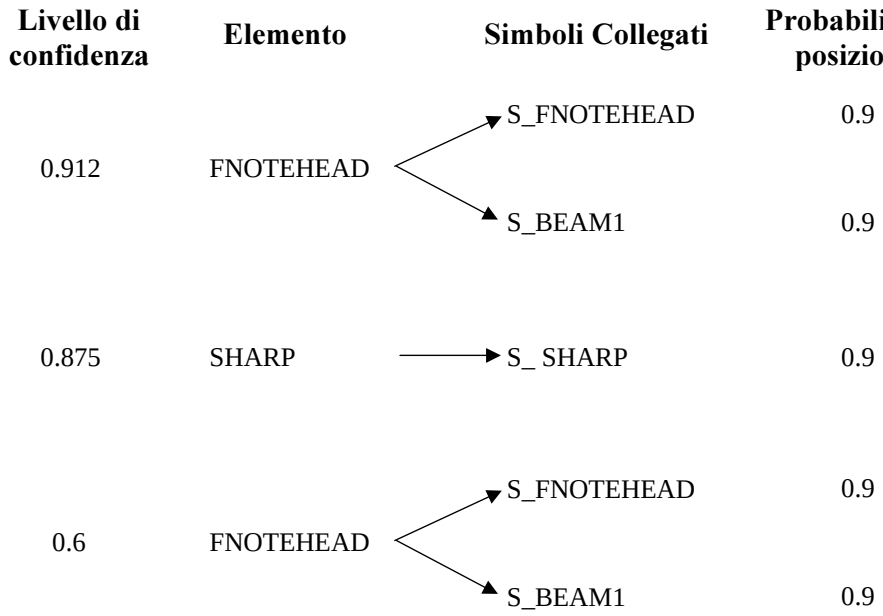


Figura 9.13: Risultato del processo di determinazione dei simboli e delle relative probabilità di posizione.

Esempio del calcolo di S_0 Si considerino i seguenti ingressi:

```
STAFF5 800 359 378 397 415 435
...
FNOTEHEAD 905 414 28 27 0.912
...
SHARP 974 360 58 19 0.875
...
FNOTEHEAD 1001 425 22 26 0.6
```

e si assuma che il valore della soglia, per quanto riguarda il livello di confidenza della rete neurale, sia pari a 0,8.

Per prima cosa è opportuno analizzare la posizione, rispetto al pentagramma, dei tre simboli. In particolare si ha che il primo token, FNOTEHEAD, e posizionato in parte dentro il pentagramma e in parte sotto (pos_6); il secondo è interno (pos_1) al pentagramma mentre il terzo è sotto (pos_4). Determinate le posizione dei token il risultato dell'applicazione della procedura di identificazione dei simboli e delle relative probabilità di posizione è rappresentato nella Figura 9.13. A questo punto è applicata la procedura per la determinazione dello stato iniziale il cui risultato è riportato nella Figura 9.14.

Come mostrato in Figura 9.14, il primo simbolo ha uno stato iniziale pari a 0 perché il livello di confidenza associato dalla rete neurale al token è maggiore di 0,8, mentre al terzo token, che ha un livello di confidenza minore della soglia, è stato associato lo stato

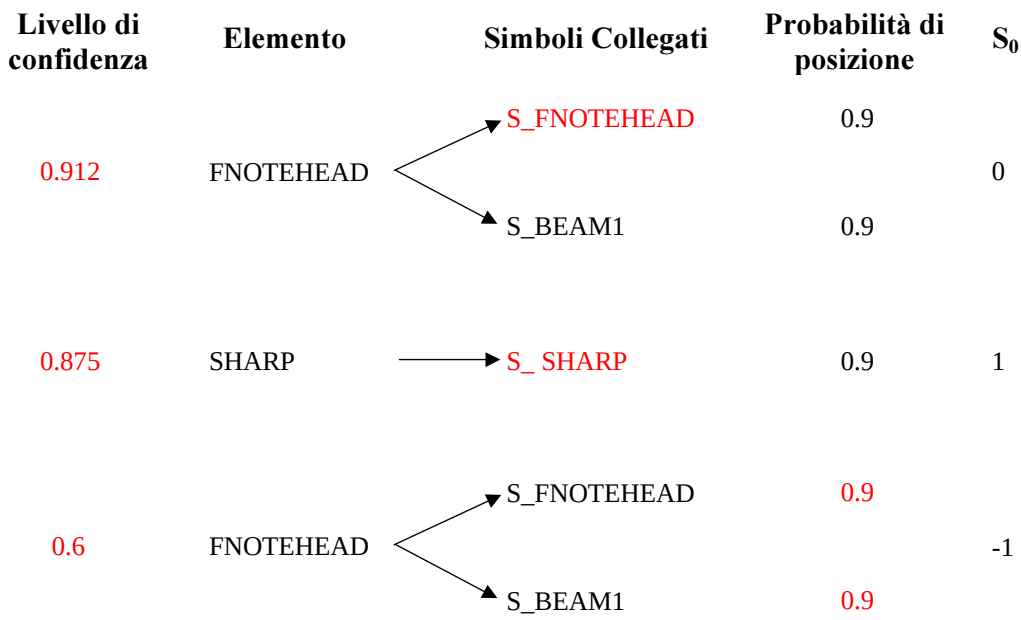


Figura 9.14: Risultato del processo di determinazione dello stato iniziale S_0.

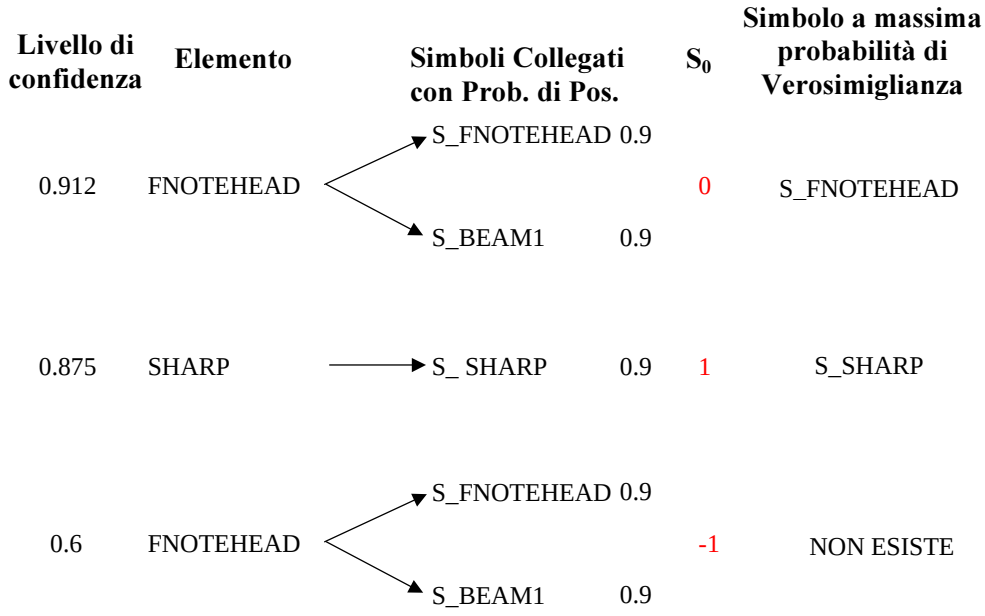


Figura 9.15: Risultato finale del processo di determinazione del simbolo a massima probabilità di verosimiglianza.

iniziale -1. Per quanto riguarda il secondo token questo ha uno stato iniziale pari a 1 perché ha un solo simbolo collegato. Il risultato finale è riportato nella Figura 9.15.

9.7 Condizioni per l'applicazione delle regole

In questa sezione è presentato il criterio per stabilire quale regola utilizzare. La regola da applicare è scelta all'interno dell'insieme $R_c \subset R$, costituito dalle regole selezionate con la chiave. Le condizioni di applicabilità della regola sono definite dalle seguenti condizioni.

Condizione 1 (F_1)
Si ha l'applicazione della regola $r \in R_c$ se verifica la formula:

$$\min_{r \in R_c} \left[\frac{\text{numero di forzature}}{\text{numero elementi usati}} \right] \tag{9.7}$$

dove:

numero di forzature indica il numero dei simboli collegati sostituiti (forzati);

numero elementi usati indica il numero di simboli che compongono lista dei simboli della regola.

Nel caso esistano più regole $r_i \in R_c$ che soddisfino la formula (9.7) si valuta anche la seguente condizione.

Condizione 2 (F_2)
Date le regole $r_i \in R_c$ che soddisfano la (9.7), la regola applicata è quella che verifica la seguente relazione:

$$\max_{r_i} [\text{numero elementi usati}] \tag{9.8}$$

dove:

numero elementi usati indica il numero di simboli che compongono la lista dei simboli della regola.

Le condizioni definite dipendono dal rapporto fra il numero di simboli collegati sostituiti (forzati) e il numero di elementi della strip che fanno parte del simbolo musicale. Nel caso di un'aggregazione senza forzature, non vi sono simboli sostituiti, pertanto il valore minimo realizzabile dalla (9.7) è il valore nullo. In presenza di più regole che consentono l'aggregazione senza forzature, la scelta ricade su quella che aggrega più simboli. Si ritiene che la regola stia aggregando il maggior numero di informazioni interne alla strip. Nel caso di aggregazioni con forzature, la scelta ricade sempre su quella che aggrega il maggior numero di simboli, ma allo stesso tempo è facilmente dimostrabile che questa ha

richiesto un numero minore si sostituzioni. Nel caso particolare in cui vi siano più regole che verificano le due condizioni, la scelta ricade sulla prima valutata.

9.8 L'aggregazione e l'evoluzione dello stato

Come descritto nelle sezioni precedenti il processo di aggregazione inizia dalla configurazione iniziale della strip E_s^0 e si avvale degli stati S_0 di partenza associati ai simboli per determinare la chiave della regola da utilizzare. Individuata la regola, i simboli che concorrono nell'applicazione sono aggregati per generare un simbolo composto. Questo passaggio si configura come un cambiamento dello stato per i simboli coinvolti e la definizione dello stato per il simbolo composto.
Pertanto si definiscono i seguenti stati che caratterizzano l'evoluzione dell'aggregazione:

- **Forzato** ($S = 2$): è lo stato che assume il simbolo soggetto all'aggregazione con forzatura. Il simbolo di partenza è sostituito con quello della lista dei simboli collegati richiesto dalla regola e temporaneamente viene considerato come utilizzato dalla regola $r \in R_c$ per valutare le condizioni di applicabilià (F_1 e F_2), rispetto alle altre regole dell'insieme R_c e che potenzialmente possono essere utilizzate.
- **Composto** ($S = 3$): è lo stato che assume il simbolo quando è stata applicata la regola che ha verificato le condizioni di applicabilità. Il simbolo ottenuto dall'aggregazione prende il nome presente nella risultante della regola.

I simboli che non sono utilizzati dalla regola permangono nello stato originario ad accezione per quelli che temporaneamente sono passati nello stato **Forzato** e per i quali è previsto il ripristino dello stato iniziale S_0 per poter valutare la regola successiva.

Per quanto riguarda le condizioni sulle variazioni dello stato, con riferimento alla Figura 9.16 si hanno le seguenti transizioni.

Siano:

- e_i l'elemento i-esimo della strip E_s^0
- $r \in R_c$ la regola di aggregazione appartenente all'insieme delle regole applicabili.

Transizione di permanenza

$$\{T_{0,0},\ T_{1,1},\ T_{-1,-1}\}: \quad \forall x \in X_i(e_i)\ x \not\rightarrow r \Rightarrow S = S_0^{(i)} \tag{9.9}$$

L'elemento resta nello stato in cui si trova se non può essere utilizzato dalla regola r né senza forzare né forzando. Il simbolo collegato è quello della configurazione iniziale ($X_{Max}^{(i)}$).

Transizione di forzatura o sostituzione simboli

$$\{T_{-1,2},\ T_{0,2}\}: \quad (x \not\rightarrow r) \wedge (\exists \hat{x} \in X_i(e_i) \mid \hat{x} \rightarrow r) \Rightarrow (S = 2) \wedge (x = \hat{x}) \tag{9.10}$$

Se il simbolo può essere sostituito con quello richiesto dalla regola r allora si esegue temporaneamente la sostituzione per la verifica dell'applicabilità della regola. Lo stato assunto è quello **Forzato**.

Transizione di ritorno allo stato iniziale

$$T_{next}: \quad \neg r \Rightarrow S = S_0 \tag{9.11}$$

Se l'elemento è stato forzato, ma la regola non è applicabile, allora si torna nello stato iniziale. Il simbolo collegato è quello della configurazione iniziale ($X^{(i)}_{Max}$) e si cambia regola.

Transizione di applicazione della regola

$$\{T_{-1,3},\ T_{0,3},\ T_{1,3}\ T_{2,3}\}: \quad (x \rightarrow r) \wedge (F_1 \wedge F_2) \Rightarrow S = 3 \tag{9.12}$$

Il simbolo collegato è usato nell'applicazione della regola r. Se avvengono le transizioni $T_{-1,3}$, $T_{0,3}$ e $T_{1,3}$, il simbolo è stato aggregato senza essere forzato. Nel caso di forzatura ($S = 2$) si ha l'applicazione della regola e la transizione $T_{2,3}$. In seguito all'aggregazione si realizza un nuovo simbolo che costituisce l'insieme dei simboli utilizzati dalla regola. Tale simbolo prende il nome dalla risultante della regola e assume lo stato **Composto**.

Transizione di permanenza con simboli composti

$$T_{3,3}: \quad r \Rightarrow S = 3 \tag{9.13}$$

Il simbolo composto è aggregato con altri simboli composti o non ancora aggregati. In seguito all'aggregazione si realizza un nuovo simbolo che costituisce l'insieme dei simboli utilizzati dalla regola. Tale simbolo prende il nome dalla risultante della regola e mantiene lo stato **Composto**.

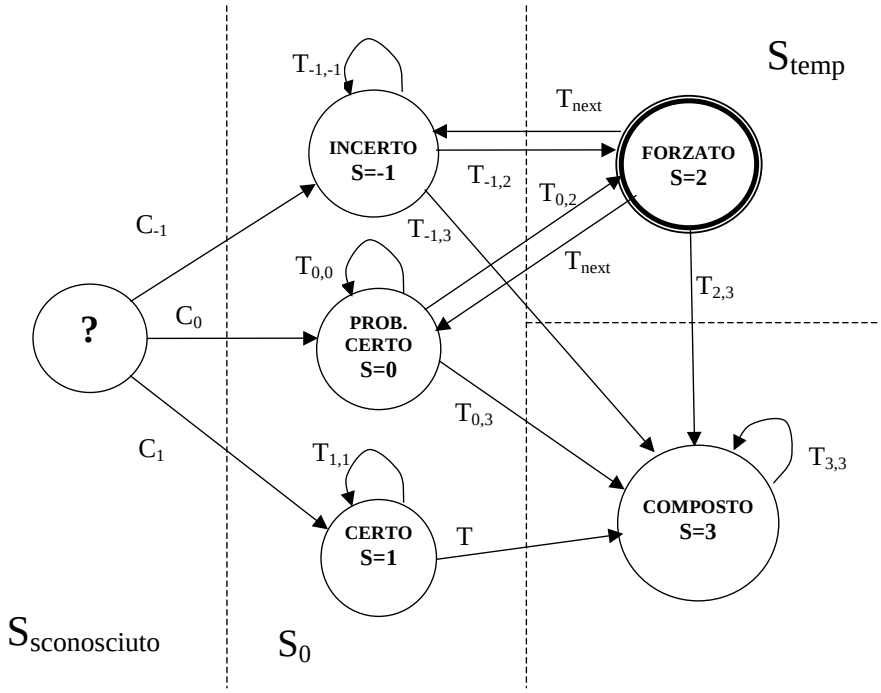


Figura 9.16: Diagramma degli stati ed aggregazione

9.9 Scelta del simbolo chiave e ricerca della regola

Una volta determinata la configurazione iniziale della strip con i simboli collegati più probabili viene definito lo stato iniziale di ciascuno di essi. Durante l'aggregazione lo stato dei simboli evolve verso condizioni che esprimono il tipo aggregazione subito o permane nello stato iniziale. Per identificare la regola da utilizzare è necessario individuare il simbolo che definisce la chiave della regola. Il criterio di scelta deve considerare sia l'evoluzione dell'aggregazione che dello stato assunto dai simboli. Nella fase iniziale dell'aggregazione le uniche indicazioni sono quelle correlate allo stato iniziale S_0. Durante l'aggregazione, l'evoluzione verso lo stato composto è sinonimo che si sta realizzando un'aggregazione di informazioni certe in quanto è frutto di una regola che ha utilizzato dei simboli partendo da quelli più probabili. Per cui il simbolo composto che è stato generato acquista un grado di certezza maggiore. Questa considerazione permette di definire un criterio di selezione in funzione dello stato S e realizzare un meccanismo di scelta prioritario.

Il criterio di scelta definito prevede che la selezione del simbolo chiave sia effettuata scegliendo, in ordine, l'elemento o il simbolo composto con stato pari a:

a. 3 (simbolo composto);

b. 1 (certezza del simbolo);

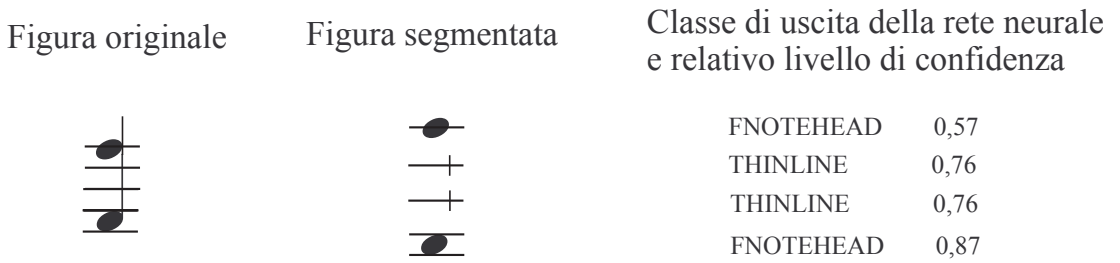


Figura 9.17: Esempio di nota in accordo, prima e dopo l'ultima fase della segmentazione, con le relative uscite della rete neurale.

c. 0 (certezza probabile del simbolo);

d. -1 (incertezza sul simbolo);

Per come è stato definita l'aggregazione, la scelta iniziale è ridotta solo ai valori assunti dallo stato S_0, solo dopo la prima aggregazione si considera lo stato del simbolo composto. Individuato il simbolo, il numero dei posti disponibili sopra e sotto l'elemento chiave è calcolato sulla base degli elementi appartenenti alla configurazione della strip posizionati rispettivamente sopra e sotto l'elemento chiave. Note le caratteristiche che deve avere la regola di aggregazione, è effettuata la ricerca dell'insieme R delle regole che hanno per chiave il simbolo selezionato. Di queste, si considerano soltanto quelle che hanno corrispondenza sul numero dei simboli sopra e sotto la chiave. L'insieme determinato è l'insieme $R_c \subset R$.

9.10 Esempio di aggregazione degli elementi di una strip. Regole Verticali

Si consideri l'esempio della Figura 9.17 nella quale sono riportati: la figura originale, gli elementi ottenuti con la segmentazione e il risultato del riconoscimento. Al termine della fase preliminare di elaborazione degli ingressi, si genera la strip (1) riportata nella Figura 9.19 e rappresentata graficamente nella Figura 9.18. Prima di analizzare in dettaglio le operazioni effettuate dal modulo di aggregazione, è necessario stabilire la posizione delle righe del pentagramma. Si supponga che, in seguito ad un aggiornamento della posizione del pentagramma, le coordinate delle linee siano poste alla distanza di 20 pixel l'una dall'altra e che la riga di riferimento, quella più alta del pentagramma, abbia l'ordinata pari a 20; si ha quindi che la riga centrale del pentagramma ha il valore dell'ordinata pari a 60.

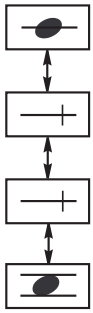


Figura 9.18: Strip ottenuta con gli elementi d'ingresso al ricostruttore.

Il primo passo è l'aggregazione dei simboli elementari. La prima operazione eseguita è l'individuazione dell'elemento chiave per la ricerca delle regole. Con riferimento alla Figura 9.19, l'elemento chiave è dato dal quarto elemento della strip, poiché non esiste alcun elemento con stato iniziale uguale a 1. Si ha quindi che il simbolo da utilizzare come chiave di ricerca è dato da **S_FNOTEHEAD**. Individuata la chiave della regola, il modulo di aggregazione deve calcolare il numero di posti disponibili sopra e sotto l'elemento chiave. Sempre osservando la Figura 9.19, si ha che il numero di posti disponibili sopra l'elemento chiave è pari a 3 mentre quelli sotto l'elemento chiave è pari a 0. Il passo successivo è la ricerca delle regole. Il modulo di aggregazione interroga la grammatica musicale comunicando all'Interfaccia i dati:

1. tipo regole = VRULE;
2. simbolo chiave = S_FNOTEHEAD;
3. simboli sopra l'elemento chiave = 3 e quelli sotto = 0;

L'insieme delle regole R è costituito dalle regole verticali relative alle note e tali da soddisfare il numero dei simboli sopra e sotto la chiave. In particolare si hanno le seguenti regole:

```
VRULE
  [S_STAFLINESTEM]
  S_STAFLINESTEM
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=4);
```

```
VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=8,inBeam);

VRULE
  S_BEAM2(conf>0.1)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=16,inBeam);

VRULE
  S_BEAM3(conf>0.1)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=32,inBeam);

VRULE
  S_BEAM4(conf>0.1)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=64,inBeam);

VRULE
  S_BEAM5(conf>0.1)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=128,inBeam);

VRULE
  S_HOOK1DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=8);

VRULE
  S_HOOK2DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=16);

VRULE
  S_HOOK3DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=32);
```

```
VRULE
  S_HOOK4DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=64);

VRULE
  S_HOOK5DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=128);
```

Le regole individuate hanno tutte la caratteristiche richieste dai dati in ingresso:

- sono tutte di tipo verticale;
- hanno il simbolo S_FNOTEHEAD come chiave;
- non presentano alcun simbolo sotto il simbolo chiave e sopra il simbolo chiave vi sono al massimo 3 simboli.

Una volta ricevuto l'insieme R il modulo di aggregazione controlla se è possibile applicare qualche regola senza eseguire delle forzature. Osservando la Figura 9.19, questo è impossibile perché gli elementi sopra il simbolo chiave non hanno associato nessun simbolo a massima verosimiglianza. Quindi si opera cercando di applicare le regole eseguendo delle forzature in modo da determinare il sottoinsieme R_c. Il sottoinsieme R_c così ottenuto è dato dalle regole:

```
VRULE
  [S_STAFLINESTEM]
  S_STAFLINESTEM
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=4);

VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=8,inBeam);
```

Le altre regole sono state scartate in quanto, per la maggior parte di esse, non è presente uno dei simboli richiesti ed in particolare i simboli: S_BEAM2, S_BEAM3, ..., S_HOOK1DWN, S_HOOK2DWN, etc. Una volta determinato l'insieme R_c si deve determinare quale fra le regole appartenenti a tale insieme deve essere applicata. Come è

richiesto dalla formula (9.7), si deve calcolare il rapporto tra il numero di forzature effettuate e il numero di elementi della strip che fanno parte della regola. Analizzando la prima delle due regole appartenenti all'insieme R_c si ha che:

```
VRULE
  [S_STAFLINESTEM]                                  Numero di Forzature = 2
  S_STAFLINESTEM                  ------------->    Numero di elementi usati = 3
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)    Numero di Forzature / Numero di elementi usati = 0,66
  ==> NOTE(stem:=Up,head:=Fill,durate:=4);
```

Infatti, il numero di elementi totali è dato dalla chiave (S_FNOTEHEAD) e dai due elementi sopra la chiave che sono forzati ad essere degli S_STAFLINESTEM. Per quanto riguarda la seconda regola si ha invece che:

```
VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)               Numero di Forzature = 3
  [S_STAFLINESTEM]                ------------->    Numero di elementi usati = 4
  ::S_FNOTEHEAD(altezzaGreat>1)                     Numero di Forzature / Numero di elementi usati = 0,75
  ==> NOTE(stem:=Up,head:=Fill,durate:=8,inBeam);
```

Concludendo, la regola da applicare è la prima. Il risultato dell'applicazione di tale regola è rappresentato dalla strip (3) della Figura 9.19 e graficamente dalla strip di Figura 9.20, nella quale è rappresentata anche l'evoluzione della strip, indicata dalla strip (2), quando sono eseguite le forzature dei simboli. A questo punto, non essendoci più simboli elementari da utilizzare come chiave, il modulo di aggregazione, come descritto nella Figura 9.4, determina se è possibile effettuare delle aggregazioni utilizzando i simboli composti, prima senza forzatura e poi con forzatura. Nel caso in esame l'unico simbolo composto è rappresentato dalla nota precedentemente determinata. Procedendo come nel caso precedente si ha che i dati da passare all'interfaccia sono, in questo caso:

1. tipo regole = VRULE;

2. simbolo chiave = NOTE;

3. simboli sopra l'elemento chiave = 1 e simboli sotto = 0;

Si noti come i simboli sopra l'elemento chiave siano 1 e non più 3 come nel caso precedente e come riportato nella strip (1) di Figura 9.21. Questo è dovuto al fatto che i tre elementi precedentemente raggruppati sono considerati come un elemento unico.

Una volta ricevuti i dati in ingresso l'interfaccia restituisce al modulo di aggregazione l'insieme delle regole R formato da:

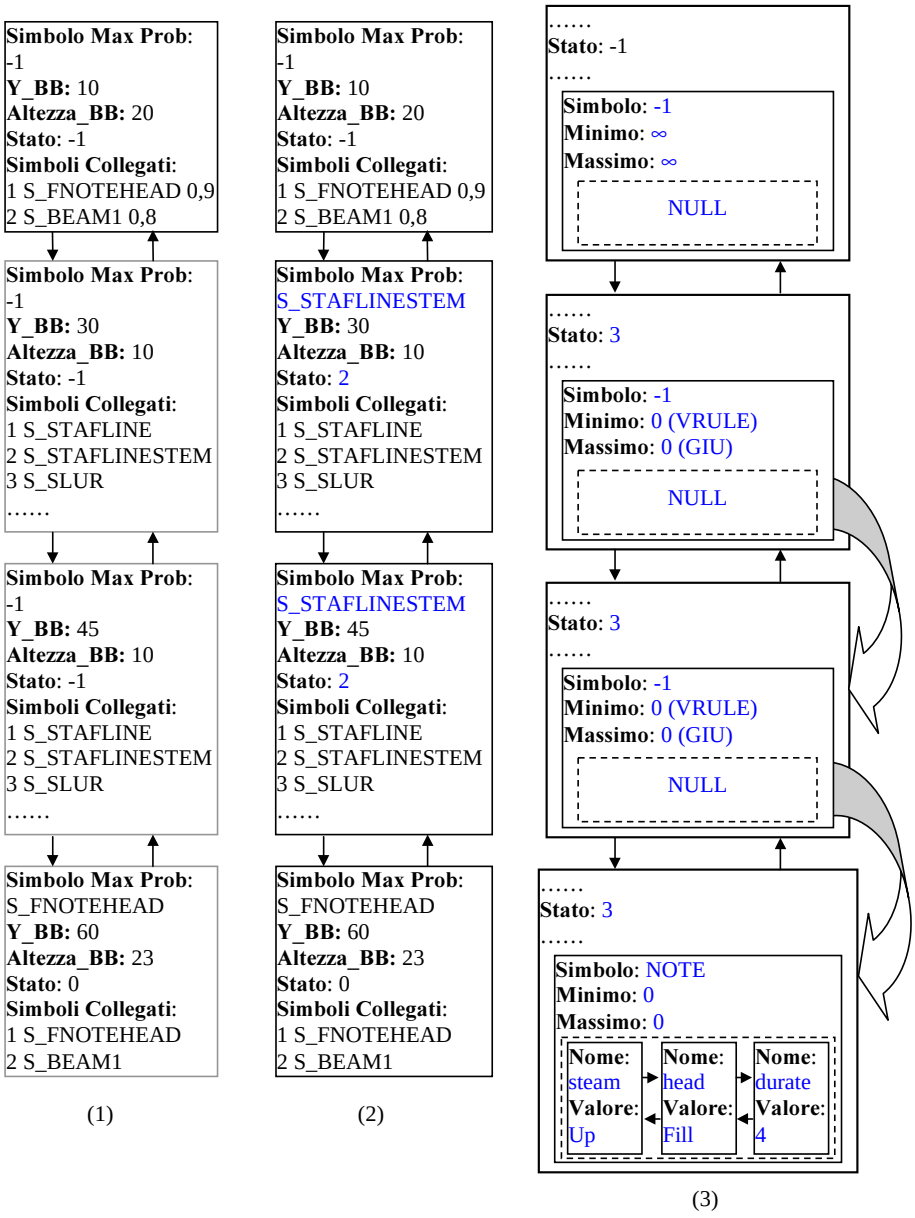


Figura 9.19: Esempio di applicazione di regola verticale forzando alcuni degli elementi ad assumere il valore di un determinato simbolo. Per ogni elemento delle strip sono riportati il simbolo a massima verosimiglianza (Simbolo a Max Prob), l'ordinata e l'altezza del "bounding box", il valore dello stato e i simboli collegati con relative probabilità di posizione. I valori degli attributi riportati sono quelli che subiscono un cambiamento e/o, in generale, sono i simboli appartenenti alla regola. Le frecce bidimensionali denotano il collegamento logico che interviene, tra gli elementi appartenenti alla regola, una volta che questa è stata applicata.

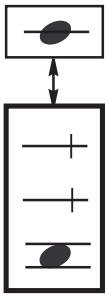


Figura 9.20: Strip ottenuta applicando una regola verticale fra simboli elementi.

```
VRULE
  S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  ::NOTE(stem=Dwn,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  S_CHORD2V
  [S_STAFLINESTEM]
  ::NOTE(stem=Dwn,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  S_CHORD3V
  [S_STAFLINESTEM]
  ::NOTE(stem=Dwn,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  S_CHORD2H
  [S_STAFLINESTEM]
  ::NOTE(stem=Dwn,head=Fill)
  ==> NOTE(inAccordo);
```

```
VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(durate=4,stem=Up)
  ==> NOTE(inBeam, durate:=8);

VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(durate=8,stem=Up)
  ==> NOTE(inBeam, durate:=16);

VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(durate=16,stem=Up)
  ==> NOTE(inBeam, durate:=32);

VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(durate=32,stem=Up)
  ==> NOTE(inBeam, durate:=64);

VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(durate=16,stem=Up)
  ==> NOTE(inBeam, durate:=32);

VRULE
  S_BEAM2(conf>0.1)
  [S_STAFLINESTEM]
  ::NOTE(durate=8,stem=Up)
  ==> NOTE(inBeam, durate:=32);

VRULE
  S_BEAM2(conf>0.1)
  [S_STAFLINESTEM]
  ::NOTE
  ==> NOTE(inBeam, durate:=16);

VRULE
  S_BEAM3(conf>0.1)
  [S_STAFLINESTEM]
  ::NOTE
  ==> NOTE(inBeam, durate:=32);
```

```
VRULE
  S_BEAM3(conf>0.1)
  [S_STAFLINESTEM]
  ::NOTE(durate=8,stem=Up)
  ==> NOTE(inBeam, durate:=64);

VRULE
  S_BEAM4(conf>0.1)
  [S_STAFLINESTEM]
  ::NOTE
  ==> NOTE(inBeam, durate:=64);

VRULE
  S_HOOK1DWN
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(durate:=8);

VRULE
  S_HOOK2DWN
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(durate:=16);

VRULE
  S_HOOK3DWN
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(durate:=32);

VRULE
  S_HOOK4DWN
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(durate:=64);

VRULE
  S_HOOK5DWN
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(durate:=128);

VRULE
  S_TENUTO
  ::NOTE(stem=Dwn)
  ==> NOTE(withTenuto);
```

```
VRULE
  S_STACCATO
  ::NOTE(stem=Dwn)
  ==> NOTE(withStaccato);

VRULE
  SIMBOLO
  [S_STAFLINE]
  ::NOTE(stem=Dwn)
  ==> NOTE;
```

Una volta determinata l'impossibilità di applicare le regole appartenenti all'insieme R senza effettuare delle forzature, il modulo di aggregazione esegue delle forzature riuscendo ad estrarre il sottoinsieme R_c di R. Tale sottoinsieme, determinato sempre eseguendo il procedimento illustrato in precedenza, è composto dalle seguenti regole:

```
VRULE
  S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  S_BEAM1(conf>0.1,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(durate=4,stem=Up)
  ==> NOTE(inBeam, durate:=8);
```

Tra le due regole appartenenti all'insieme R_c, il modulo di aggregazione sceglie quella che soddisfa alla formula (9.7). Nel caso attualmente in esame si ha però che per entrambe le regole tale rapporto vale 0,5. In questo caso, si considera la regola che soddisfa la formula (9.8), ma valutando la formula si ha che il numero di elementi utilizzati è pari a 2 per entrambe le regole. In questo caso particolare, il modulo di aggregazione considera la prima regola appartenente all'insieme R_c data da:

```
VRULE
  S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(inAccordo);
```

Il risultato dell'applicazione di tale regola è rappresentato dalla strip(2) di Figura 9.21 e graficamente dalla Figura 9.22.

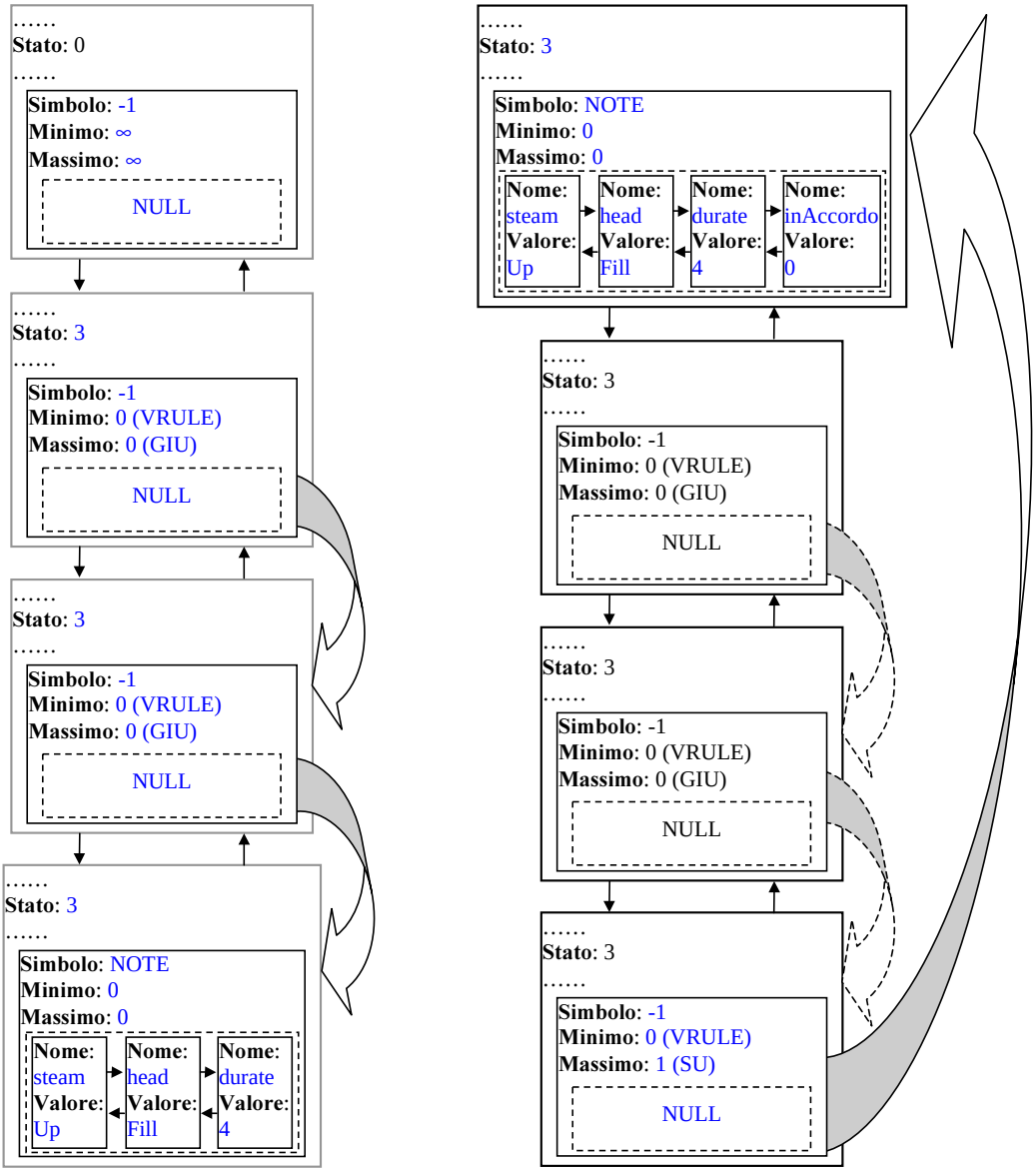


Figura 9.21: Esempio di applicazione di regola verticale sulla strip di Figura 9.19. I valori riportati sono quelli che subiscono un cambiamento e/o, in generale, sono i simboli musicali appartenenti alla regola. Le frecce bidimensionali a tratto continuo denotano il collegamento logico che interviene, tra gli elementi appartenenti alla regola, dopo l'applicazione.

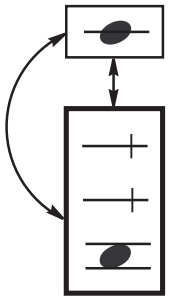


Figura 9.22: Strip finale.

9.11 Esempio di aggregazione degli elementi appartenenti a strip diverse. Regole Orizzontali

Dopo aver considerato l'esempio relativo all'applicazione delle regole verticali, in questa sezione è presentato un esempio di applicazione di regole orizzontali. È opportuno ricordare che le regole orizzontali operano considerando elementi appartenenti a strip diverse e questo fa sì che abbiano delle caratteristiche di distanza contenute nelle condizioni dei simboli che ne fanno parte. In dettaglio, sono stati introdotte le seguenti condizioni:

- **dxMax**, rappresenta la distanza massima lungo l'ascissa;
- **dyMax**, rappresenta la distanza massima lungo l'ordinata.

Si considerino le strip riportate in Figura 9.23 e si ipotizzi di essere posizionati sulla strip B. La modalità di aggregazione è simile al caso delle regole verticali con la differenza che non è necessario calcolare il numero di elementi sopra e sotto la chiave. La prima operazione è quella di determinare l'elemento chiave che nel caso in esame è dato dalla simbolo musicale NOTE della strip B. Una volta comunicato all'interfaccia il simbolo chiave, questa ritorna l'insieme R formato dalle seguenti regole:

```
HRULE
  ALTERAZIONE(dxMax=0.5,dyMax=0.4) ::NOTE ==> NOTE(inAlterazione);

HRULE
  S_STAFLINE(dxMax=0.4,dyMax=0.7) ::SIMBOLO(isPunto) ==> SIMBOLO(isPunto,DotwithStaff);

HRULE
  NOTE(dxMax=1,dyMax=1.2) ::SIMBOLO(DotwithStaff) ==> NOTE;
```

```
HRULE
  ::NOTE(dxMax=0.5,dyMax=0.7) SIMBOLO(isPunto) ==> NOTE;

HRULE
  ::NOTE(isPunto,dyMax=0.7) SIMBOLO(isPunto) ==> NOTE;
```

Per la determinazione dell'insieme R_c si ipotizza che non ci siano strip a destra della strip B. Sotto questa condizione si ha che il sottoinsieme R_c è costituito dalla sola regola:

```
HRULE
  ALTERAZIONE(dxMax=0.5,dyMax=0.4) ::NOTE ==> NOTE(inAlterazione);
```

che è immediatamente applicata. Il risultato dell'aggregazione è riportato nella Figura 9.24.

Questo esempio consente di analizzare più in dettaglio le condizioni sulla distanza che generalmente sono presenti nelle regole orizzontali. Infatti, affinché una regola orizzontale, contenente delle condizioni del tipo dxMax e dyMax, possa essere applicata devono essere verificate le Equazioni 9.14. Ovvero, i baricentri delle ascisse e delle ordinate dei "bounding box" degli elementi in esame, devono avere rispettivamente una distanza inferiore al prodotto della distanza fra due righe del pentagramma, scalata del fattore di scala k, per il valore delle caratteristiche dxMax e dyMax . Il significato delle coordinate presenti nelle Equazioni(9.14) è riportato in Figura 9.25.

$$\begin{array}{ll} |bar_{x_A} - bar_{x_B}| < dxMax * k(dist\_penta) & se\ \exists\ dxMax \\ |bar_{y_A} - bar_{y_B}| < dyMax * k(dist\_penta) & se\ \exists\ dyMax \end{array} \tag{9.14}$$

con:

$$bar_{x_A} = x_A + \frac{larghezza_A}{2}$$
$$bar_{x_B} = x_B + \frac{larghezza_B}{2}$$
$$bar_{y_A} = y_A + \frac{altezza_A}{2}$$
$$bar_{y_B} = y_B + \frac{altezza_B}{2}$$

$dist\_penta$ = distanza tra le righe del pentagramma;
k = fattore di scala;

9.12 Procedura per la determinazione delle legature, del crescendo e del decrescendo

In questa sezione è affrontata la ricostruzione dei simboli musicali caratterizzati da un'evoluzione grafica orizzontale. I simboli presi in considerazione sono rappresentati dalle:

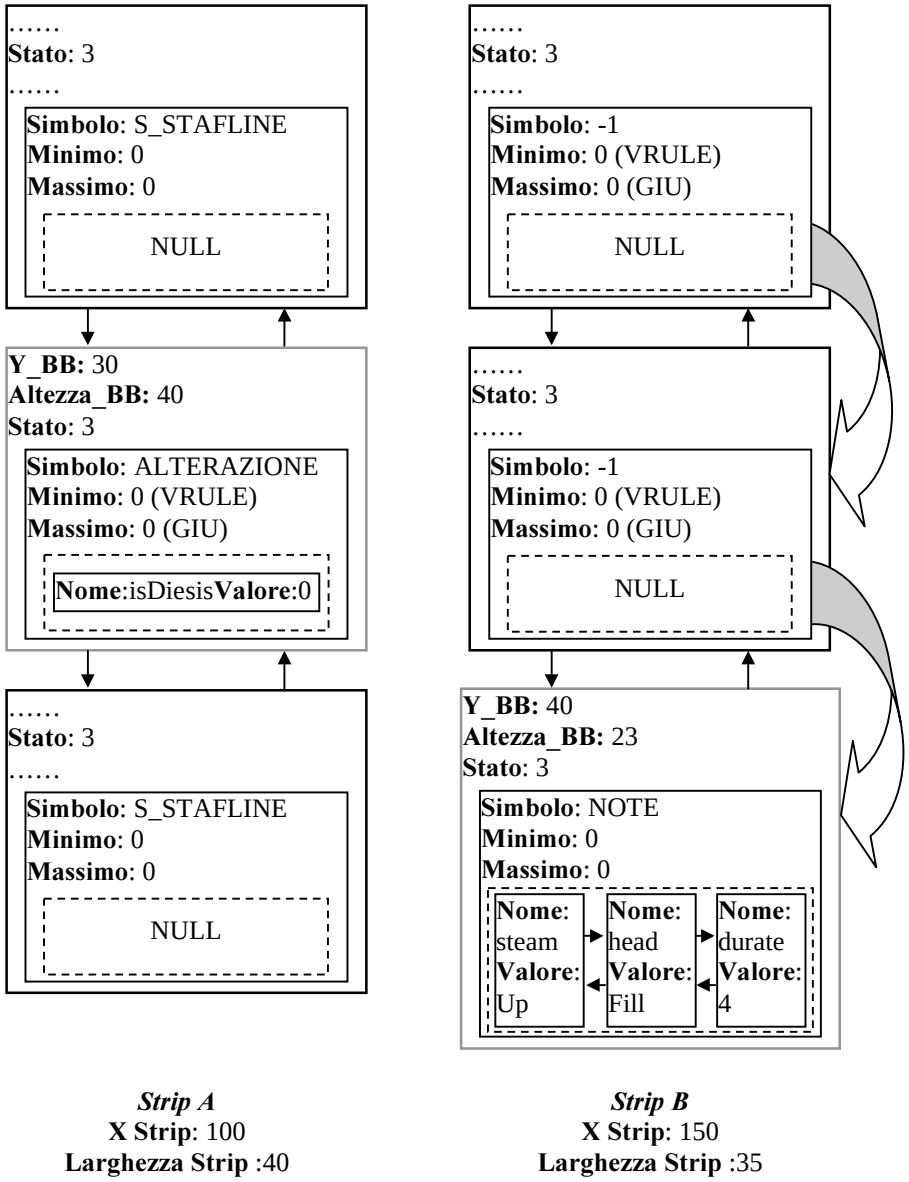


Figura 9.23: Esempio di due strip adiacenti contenenti la strip A (un diesis) e la strip B (una nota). Sono state riportate l'ascissa e la larghezza delle due strip, e l'ordinata e l'altezza degli elementi di interesse.

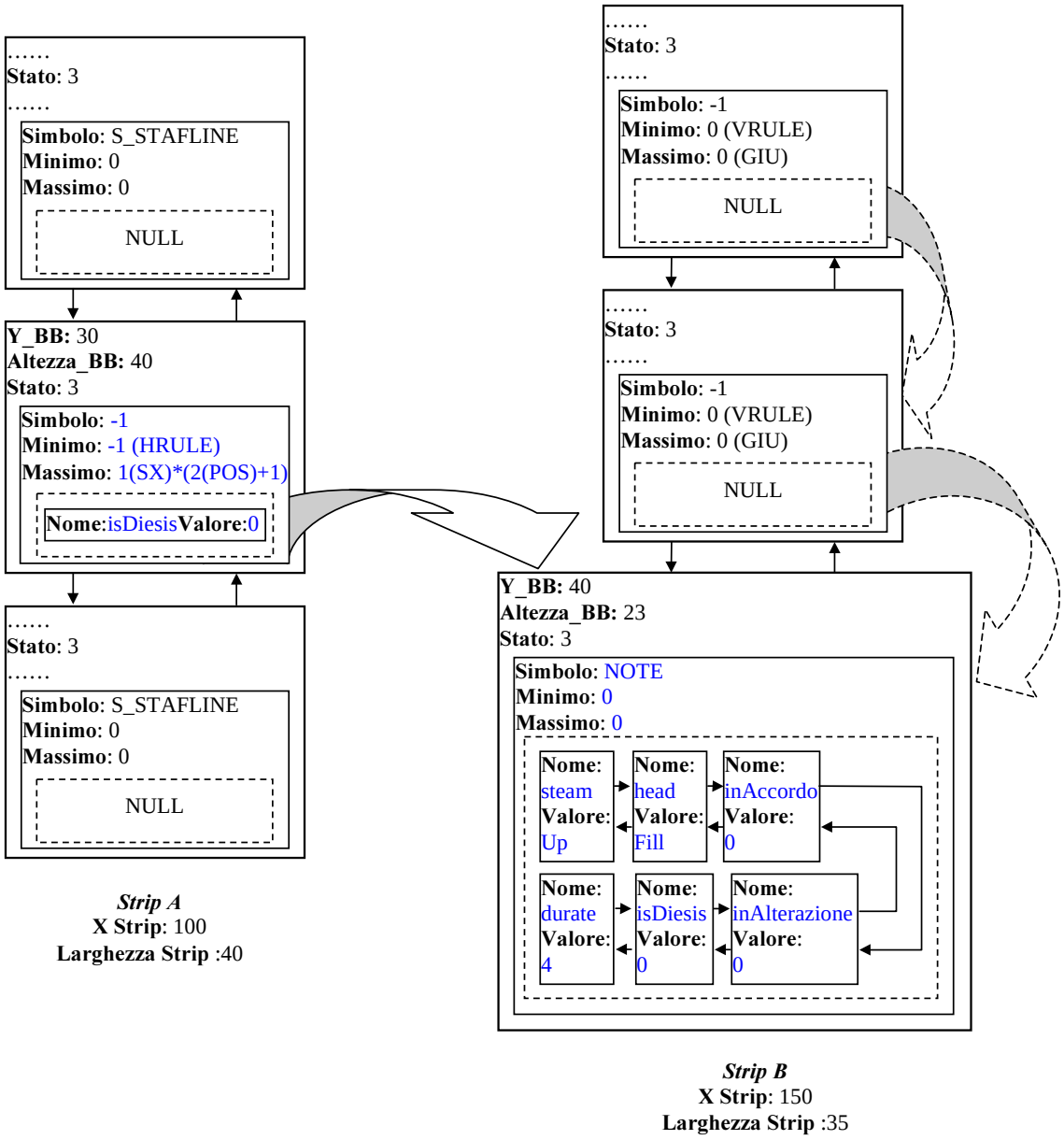


Figura 9.24: Risultato dell'applicazione di una regola orizzontale sulle strip di Figura 9.23. La freccia bidimensionale a tratto continuo rappresenta il collegamento logico che si forma fra gli elementi delle due strip appartenenti alla regola applicata.

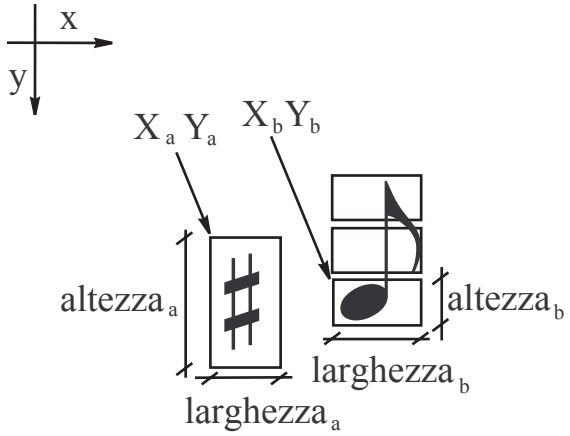


Figura 9.25: Significato delle coordinate del bounding box

- Legature;
- Crescendo;
- Decrescendo.

Questi simboli musicali sono composti da piccoli tratti di linee orizzontali, più o meno oblique, che sono classificati dalla rete neurale con il simbolo THINLINE. Quindi, oltre alla regole orizzontali necessarie per aggregare i tratti di linee, occorre una procedura specifica per la determinazione di tali simboli. Le regole orizzontali da sole sono usate per aggregare tra loro i simboli elementari appartenenti alla classe THINLINE, non ancora utilizzati da altre regole, forzandoli ad assumere il simbolo musicale richiesto dalla regola stessa, se questo è presente fra quelli collegati e se la sua probabilità di posizione è maggiore di 0. Il risultato finale dell'aggregazione è il simbolo LINE e rappresenta una generica retta. Il metodo di aggregazione fra gli elementi di strip adiacenti per mezzo delle regole orizzontali che generano le linee, è lo stesso descritto nel Paragrafo 9.11. Si considerino, ad esempio, le regole:

Regola (1)

```
HRULE
  ::S_SLUR(dyMax=0.4) ::S_SLUR ==> LINE;
```

Regola (2)

```
HRULE
  ::LINE(dyMax=0.4) ::LINE ==> LINE;
```

Regola (3)

```
HRULE
  ::S_CRESCLINE(dyMax=0.4) ::S_CRESCLINE ==> LINE;
```

Figura 9.26: Esempio di legatura con crescendo

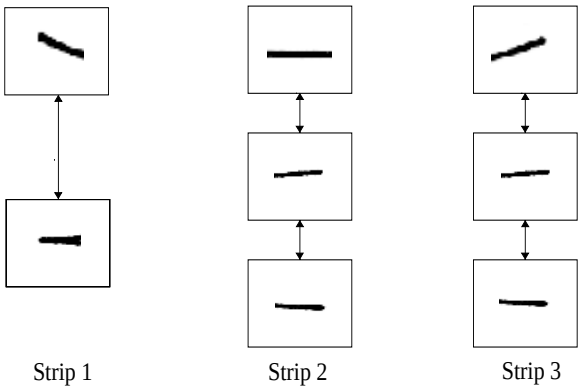


Figura 9.27: Strip relative alle linee della legatura e del crescendo.

estratte dalle regole orizzontali relative alle linee. In queste regole si possono notare due particolarità: la presenza di una sola condizione sulla distanza massima, quella verticale, e la presenza di due elementi chiave. La presenza della sola condizione verticale è stata introdotta in quanto non è possibile conoscere la distanza orizzontale che due tratti di linee possono avere: questa indeterminazione è introdotta dal segmentatore. La seconda caratteristica è stata introdotta per limitare il numero di regole da verificare. Si consideri a questo punto l'esempio di Figura 9.26, la cui rappresentazione in strip è mostrata nella Figura 9.27 e nella quale sono stati riportati solo gli elementi relativi alla legatura e al crescendo. L'applicazione delle regole orizzontali (1), (2) e (3), sulle strip di Figura 9.27, permettono di collegare gli elementi delle strip in esame in modo da formare le linee rappresentate dalle frecce tratteggiate di Figura 9.28. Una volta determinate le linee devono essere associate con i relativi simboli musicali. Per risolvere questo problema è stato creato uno specifico algoritmo che utilizza una regola di scrittura musicale, relativa alla posizione che le legature, i crescendo e i decrescendo assumono relativamente l'uno rispetto all'altro, data da:

- al di sopra del pentagramma possono esserci solo legature;

- sotto il pentagramma le legature sono situate prima di eventuali crescendo o decrescendo.

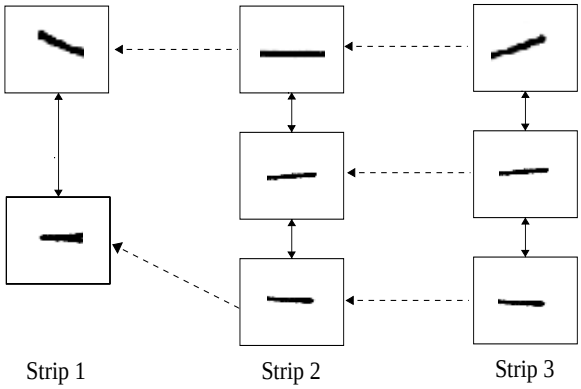


Figura 9.28: Strip relative alle linee della legatura e del decrescendo. Le frecce continue indicano i collegamenti interni alla strip, quelle tratteggiate i collegamenti orizzontali che creano le regole.

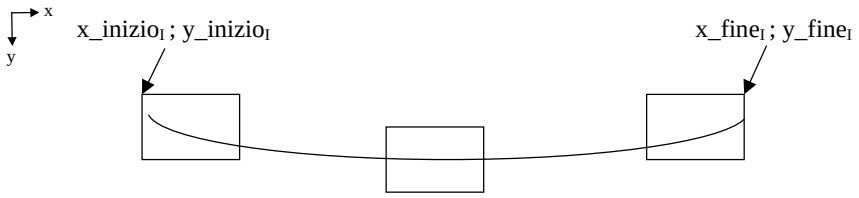


Figura 9.29: Prima linea determinata dall'algoritmo di determinazione dei simboli orizzontali.

Questo algoritmo cerca di associare le righe, determinate dalle regole orizzontali, a questi simboli musicali effettuando un'analisi sulla posizione che queste occupano rispetto al pentagramma e l'una rispetto all'altra. Per analizzare questo algoritmo si consideri l'esempio dato dalla Figura 9.28. Questa procedura inizia esaminando la battuta in esame dalla prima strip fino a quando non trova una riga; nel caso in esame è data dalla legatura come si vede dalla Figura 9.29. Calcolate le coordinate iniziali e finali della riga si determina la posizione che questa occupa relativamente al pentagramma. Nel caso in cui la posizione sia interna o sopra al pentagramma allora è associata ad una legatura mentre nel caso sia posta sotto il pentagramma, come nell'esempio in esame, non è possibile a questo stadio fare nessuna associazione. Nel caso in cui non sia stato possibile determinare il simbolo musicale associato alla linea in esame, si mantengono in memoria le coordinate e si continua ad esaminare la battuta fino a quando non viene determinata una seconda linea. A questo punto, una volta determinate le coordinate iniziali e finali della nuova linea si opera un confronto fra le distanze, iniziali e finali, come raffigurato nella Figura 9.30. Se la distanza fra i punti iniziali che finali è superiore ad una distanza di riferimento, espressa in funzione della distanza fra le righe del pentagramma, allora le due linee non possono formare nè un crescendo nè un decrescendo. L'esclusione di tali simboli musicali permette

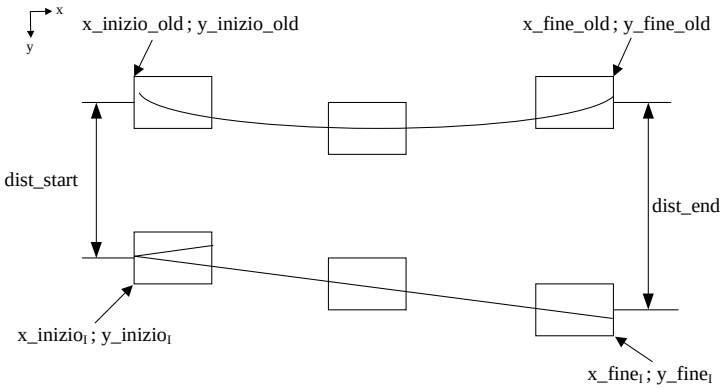


Figura 9.30: Prime due linee messe a confronto dall'algoritmo di determinazione dei simboli orizzontali. I valori delle coordinate che terminano con *old* si riferiscono alla linea precedentemente determinata e alla quale non è stato ancora assegnato nessun simbolo musicale.

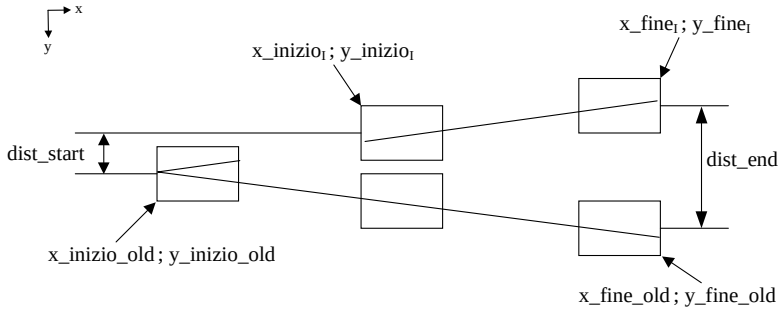


Figura 9.31: Confronto fra le distanze delle ultime linee della battuta in esame.

di associare la linea più vicina al pentagramma ad una legatura, nel caso in esame la prima linea come riportato nella Figura 9.32. La linea rimanente è considerata non assegnata. A questo punto, l'analisi della battuta procede fino a quando non viene determinata una nuova linea o non finisce la battuta stessa. Nell'esempio considerato è determinata una terza linea all'interno della battuta. Calcolate le coordinate iniziali e finali di tale linea si procede nuovamente ad analizzare la distanza iniziale e finale che intercorre fra i punti iniziali e finali delle due righe, come mostrato nella Figura 9.31. In questo caso, poiché le distanze sono minori della distanza di riferimento, si conclude che le due righe, considerate assieme, formano o un crescendo o un decrescendo. In particolare:

- se la distanza iniziale è minore di quella finale allora si ha un crescendo, come evidenziato dalla Figura 9.32;
- se la distanza iniziale è maggiore di quella finale allora il simbolo è un decrescendo.

Di seguito è riportato lo pseudocodice relativo alla procedura per la determinazione dei crescendo, decrescendo e legature.

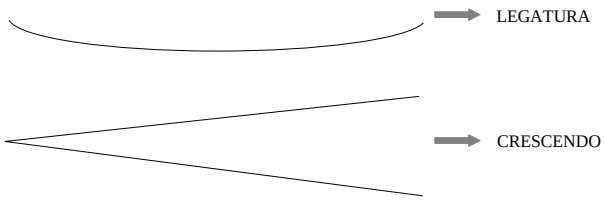


Figura 9.32: Simboli musicali orizzontali assegnati alle linee della battuta in esame.

SE numero_linee_non_assegnate = 0 ALLORA
Determina_Coordinate(x_inizio_I,x_fine_I,y_inizio_I,y_fine_I
x_inizio_{II},x_fine_{II},y_inizio_{II},y_fine_{II})
SE la linea si trova sopra o dentro il pentagramma ALLORA
associa lo SLUR alla line
numero_linee_non_assegnate = 0
ALTRIMENTI
SE (x_inizio_{II} != 0) && (x_fine_{II} != 0) ALLORA
SE ($x\_inizio_I = x\_inizio_{II}$) && ($x\_fine_I = x\_fine_{II}$) ALLORA
SE (dist_inizio > dist_end) && (dist_end < k*0.5) ALLORA
le due linee sono un DECRESCENDO
numero_linee_non_assegnate = 0
SE (dist_inizio < dist_end) && (dist_start < k*0.5) ALLORA
le due linee sono un CRESCENDO
numero_linee_non_assegnate = 0
ALTRIMENTI
SE (x_inizio_I != x_inizio_{II}) || (x_fine_I != x_fine_{II}) ALLORA
Calcola_Nuovi_Estremi(x_inizio_I,x_fine_I,y_inizio_I,y_fine_I)
memorizza i dati:
1 x_inizio_old = x_inizio_I;
2 x_fine_old = x_fine_I;
3 y_inizio_old = y_inizio_I;
4 y_fine_old = y_fine_I;
5 x_linea_old = x_linea.
numero_linee_non_assegnate = 1
ALTRIMENTI
memorizza i dati:
1 x_inizio_old = x_inizio_I;
2 x_fine_old = x_fine_I;
3 y_inizio_old = y_inizio_I;

4 y_fine_old = y_fine_I;
5 x_linea_old = x_linea.
numero_linee_non_assegnate = 1
ALTRIMENTI
Determina_Coordinate(x_inizio_I,x_fine_I,y_inizio_I,y_fine_I
x_inizio_{II},x_fine_{II},y_inizio_{II},y_fine_{II})
SE (x_inizio_{II} != 0) && (x_fine_{II} != 0) ALLORA
SE (x_linea_old != x_linea) ALLORA
SE (x_inizio_I = x_inizio_old) && (dist_start > dist_end)
&& (dist_end < k*0,5) ALLORA
SE x_linea_old < x_linea ALLORA
le due linee sono un DECRESCENDO
numero_linee_non_assegnate = 0
ALTRIMENTI
le due linee sono un CRESCENDO
numero_linee_non_assegnate = 0
ALTRIMENTI
SE (x_fine_I = x_fine_old) && (dist_start < dist_end)
&& (dist_start < k*0,5) ALLORA
SE x_linea_old < x_linea ALLORA
le due linee sono un CRESCENDO
numero_linee_non_assegnate = 0
ALTRIMENTI
le due linee sono un DECRESCENDO
numero_linee_non_assegnate = 0
ALTRIMENTI
SE (y_inizio_old > y_inizio_I) && (y_fine_old > y_fine_I)
si associa lo SLUR alla linea vecchia
numero_linee_non_assegnate = 1
memorizza i dati:
1 x_inizio_old = x_inizio_I;
2 x_fine_old = x_fine_I;
3 y_inizio_old = y_inizio_I;
4 y_fine_old = y_fine_I;
5 x_linea_old = x_linea.
ALTRIMENTI
associa lo SLUR alla line nuova
numero_linee_non_assegnate = 1

ALTRIMENTI

SE (dist_start $>$ k*0,5) && (dist_end $>$ k*0,5) ALLORA

si associa lo SLUR alla linea vecchia

numero_linee_non_assegnate $= 1$

memorizza i dati:

1 x_inizio_old $= x\_inizio_I$;

2 x_fine_old $= x\_fine_I$;

3 y_inizio_old $= y\_inizio_I$;

4 y_fine_old $= y\_fine_I$;

5 x_linea_old $=$ x_linea.

SE (dist_start $>$ dist_end) && (dist_end $<$ k*0,5) ALLORA

le due linee sono un DECRESCENDO

numero_linee_non_assegnate $= 0$

SE (dist_start $<$ dist_end) && (dist_start $<$ k*0,5) ALLORA

le due linee sono un CRESCENDO

numero_linee_non_assegnate $= 0$

ALTRIMENTI

SE ($y\_inizio_I > y\_fine_I$) ALLORA

le due linee sono un DECRESCENDO

numero_linee_non_assegnate $= 1$

ALTRIMENTI

le due linee sono un CRESCENDO

numero_linee_non_assegnate $= 1$

9.13 L'archivio della grammatica musicale

In questo paragrafo è riportata la struttura dell'archivio della grammatica musicale implementato nel MOOR. L'archivio è stato suddiviso in due insiemi di regole: **regole di base** e **regole avanzate**.

BASE	VERTICALI	Chiavi
		Indicazione Temporali
		Note
		Pause
		Alterazioni
		Simboli
	ORIZZONTALI	Linee
AVANZATE	VERTICALI	Accordi
		Note in Beam
		Note con Uncino
		Note con Simboli
	ORIZZONTALI	Collegamenti
		Ricostruzione Simboli

Nelle regole di base sono contenute le regole verticali e orizzontali che consentono l'aggregazione dei simboli musicali completi o parziali. Nelle regole avanzate sono contenute le regole verticali e orizzontali che consentono di assemblare i simboli ottenuti con le regole di base e di realizzare: configurazioni musicali complesse, composizioni e ricostruzione di simboli completi, e gestione delle relazioni orizzontali tra simboli completi.

9.13.1 Le regole verticali di base

Per quanto riguarda le regole verticali queste sono divise in due categorie: quelle relative ai soli simboli elementari e quelle relative al collegamento fra simboli composti con simboli elementari e/o composti.

Chiavi: nelle regole relative alle chiavi il simbolo musicale della risultante è rappresentato da CLEF. Le assegnazioni possibili sono:

isTenore, isContralto, isMezzosoprano e isSoprano rappresentano le CHIAVI DI DO;

isBasso e isBaritono rappresentano le CHIAVI DI FA;

isTreble rappresenta la CHIAVE DI VIOLINO o DI SOL.

Indicatori del Tempo: sono le regole relative agli indicatori temporali (le frazioni). Per questo insieme di regole il simbolo musicale della risultante è rappresentato da TIME e il valore dell'assegnazione relativa alla durata (durate) è espressa mediante una frazione.

Note: per le regole relative alle note, il simbolo musicale della risultante è rappresentato da NOTE e le assegnazioni che caratterizzano la nota sono date da:

stem che rappresenta la direzione che può avere il gambo: *Up* significa che il gambo è rivolto verso l'alto, *Dwn* significa che il gambo è rivolto verso il basso;

head che rappresenta il tipo della testa e può assumere i valori: *Empty* significa che la testa della nota è vuota, *Full* significa che la testa della nota è piena;

durate che rappresenta la durata della nota determinata;

inBeam che quando presente significa che la nota risultante è una nota in beam.

In questo insieme di regole sono gestite tutte le note, a partire dalla semibreve fino alla semibiscroma, con uncino o in beam e indipendentemente dalla direzione del gambo.

Pause: sono le regole relative alle pause e il simbolo musicale della risultante è rappresentato da PAUSE. Il valore dell'assegnazione relativa alla durata (durate) cambia da regola a regola.

Alterazioni: le regole relative alle alterazioni gestiscono i simboli musicali quali il diesis, il bemolle, il bequadro, il doppio diesis e il doppio bemolle. Per questo insieme di regole il simbolo musicale della risultante è rappresentato da ALTERAZIONE mentre le assegnazioni della risultante sono:

isDiesis per rappresentare un DIESIS;

isBemolle per rappresentare un BEMOLLE;

isBequadro per rappresentare un BEQUADRO;

isDdiesis per rappresentare un DOPPIO DIESIS;

isDbemolle per rappresentare un DOPPIO BEQUADRO.

Simboli: questo insieme è costituito dalle le regole relative che gestiscono altri simboli musicali. Per "simboli" si considerano: l'accento, la corona, il tenuto, lo staccato e il punto di valore. Per questo insieme di regole si ha che il simbolo musicale della risultante è rappresentato da SIMBOLO mentre le assegnazioni della risultante sono:

isAccento per rappresentare un ACCENTO;

isTenuto per rappresentare un TENUTO;

isStaccato per rappresentare un STACCATO;

isCorona per rappresentare una CORONA;

isPunto per rappresentare un PUNTO DI VALORE.

9.13.2 Le regole verticali avanzate

In questo paragrafo sono descritte le regole verticali relative all'aggregazione di simboli composti con simboli musicali composti e/o i simboli elementari.

Accordi: nelle regole relative agli accordi il simbolo musicale della risultante è rappresentato da NOTE al quale è aggiunta l'assegnazione *inAccordo*. L'accordo è visto come una configurazione di note con la stessa durata ma con le posizioni delle teste differenti.

Note in beam identificano le regole che gestiscono le note all'interno dei gruppi e connesse dalle travi (beam). Per questo insieme di regole il simbolo musicale della risultante è rappresentato da NOTE, al quale è aggiunto l'assegnazione *inBeam* e la durata espressa dal numero di travi.

Note con uncino: identificano le regole relative che gestiscono le note con uncino. Per questo insieme di regole si ha che il simbolo musicale della risultante è rappresentato da NOTE, al quale è aggiunta la durata espressa dal numero di uncini.

Note con simboli identificano le regole relative a note collegate con SIMBOLI. Per questo insieme di regole il simbolo musicale della risultante è rappresentato da NOTE al quale è aggiunta l'assegnazione del SIMBOLO appartenente alla lista dei simboli musicali della regola.

9.13.3 Le regole orizzontali di base

È l'insieme delle regole orizzontali necessarie alla creazione delle linee che costituiscono le legature, i crescendo e i decrescendo.

Le linee: per questo insieme di regole il simbolo musicale della risultante è rappresentato da LINE. È importante sottolineare la presenza delle condizioni di distanza orizzontale (dxMax) e verticale (dyMax).

9.13.4 Le regole orizzontali avanzate

È l'insieme delle regole orizzontali necessarie alla realizzazione dei collegamenti tra simboli musicali composti e alla ricostruzione di simboli.

I collegamenti: includono le regole necessarie per:

- assegnare le alterazioni e i punti di valore alle note,
- costruire note in configurazione di accordo,
- assemblare l'armatura di chiave (chiave e alterazioni).

Nelle condizioni dei simboli appartenenti alla lista dei simboli musicali della regole sono presenti le condizioni di distanza orizzontale (dxMax) e verticale (dyMax).

Ricostruzione simboli: includono regole che permettono di ricostruire simboli frammentati dalla segmentazione e con componenti su strip adiacenti. Nelle condizioni dei simboli appartenenti alla lista dei simboli musicali della regole sono presenti le condizioni di distanza orizzontale (dxMax) e verticale (dyMax).

9.13.5 Il significato delle condizioni e delle assegnazioni

In questo paragrafo sono descritti gli attributi utilizzati nelle regole verticali e orizzontali. Prima di elencare tutti gli attributi è opportuno precisare che il valore numerico assegnato agli attributi rappresenta la percentuale di un valore preso come riferimento e rappresentato dalla distanza delle righe del pentagramma. Se, ad esempio, si ha una condizione del tipo:

altezzaGreat>0.8

questa condizione significa che l'altezza del simbolo musicale deve essere maggiore di 0.8 volte la distanza delle righe del pentagramma (80%). In questo modo la regola assume una valenza generica e gli attributi costituiscono dei parametri rispetto ad un valore caratteristico dello spartito in esame. Si risolve così il problema della variabilità delle dimensioni relative al font.

Di seguito sono riportati gli attributi presenti nella grammatica:

- *isTenor* indica una Chiave di Tenore e non può avere nessun valore assegnato;

- *isContralto* indica una Chiave di Contralto e non può avere nessun valore assegnato;
- *isMezzosoprano* indica una Chiave di Mezzosoprano e non può avere nessun valore assegnato;
- *isSoprano* indica una Chiave di Soprano e non può avere nessun valore assegnato;
- *isBasso* indica una Chiave di Basso e non può avere nessun valore assegnato;
- *isBaritono* indica una Chiave di Baritono e non può avere nessun valore assegnato;
- *isTreble* indica una Chiave di Violino e non può avere nessun valore assegnato;
- *isDiesis* indica il Diesis e può essere associato un valore intero positivo che ne rappresenta il numero;
- *isBequadro* indica il Bequadro e può essere associato un valore intero positivo che ne rappresenta il numero;
- *isBemolle* indica il Bemolle e può essere associato un valore intero positivo che ne rappresenta il numero;
- *isDdiesis* indica il doppio Diesis e può essere associato un valore intero positivo che ne rappresenta il numero;
- *isDbemolle* indica il doppio Bemolle e può essere associato un valore intero positivo che ne rappresenta il numero;
- *isCorona* indica la Corona e non può avere nessun valore associato;
- *isAccento* indica l'Accento e non può avere nessun valore associato;
- *isTenuto* indica il simbolo di Tenuto e non può avere nessun valore associato;
- *isStaccato* indica il simbolo di Staccato e non può avere nessun valore associato;
- *isPunto* indica il Punto di Valore e può essere associato un valore intero positivo che ne rappresenta il numero;
- *isDoublePunto* indica i due punti verticali e può essere associato un valore intero positivo che ne rappresenta il numero;
- *isPiano* indica che il simbolo rappresenta un'indicazione dinamica di PIANO;
- *isForte* indica che il simbolo rappresenta un'indicazione dinamica di FORTE;
- *withPiano* indica che la strip contiene l'indicazione dinamica di PIANO e non può avere nessun valore associato;

- *withForte* indica che la strip contiene l'indicazione dinamica di FORTE e non può avere nessun valore associato;
- *withTenuto* indica che la strip contiene il Tenuto e non può avere nessun valore associato;
- *withStaccato* indica che la strip contiene lo Staccato e non può avere nessun valore associato;
- *withAccento* indica che la strip contiene l'Accento e e può assumere i valori:
  - *Up* indica che l'accento è posizionato sopra la testa della nota;
  - *Dwn* indica che l'accento è posizionato sotto la testa della nota.
- *withCorona* indica che la strip contiene la Corona e non può avere nessun valore associato;
- *stem* indica il gambo di una nota e può assumere i valori:
  - *Up* indica che il gambo è rivolto verso l'alto;
  - *Dwn* indica che il gambo è rivolto verso il basso.
- *duration* indicatore della durata e del tempo e può assumere i valori:
  - c che indica il tempo di 4/4;
  - C che indica il tempo di 2/2;
  - 2/2 che indica il tempo di 2/2;
  - 2/4 che indica il tempo di 2/4;
  - 3/4 che indica il tempo di 3/4;
  - 4/4 che indica il tempo di 4/4;
  - 3/8 che indica il tempo di 3/8;
  - 6/8 che indica il tempo di 6/8;
  - 9/8 che indica il tempo di 9/8.
  - 8/4 che indica la durata di 8/4;
  - 4/4 che indica la durata di 4/4;
  - 2 che indica la durata di 1/2;
  - 4 che indica la durata di 1/4;
  - 8 che indica la durata di 1/8;
  - 16 che indica la durata di 1/16;

  - 32 che indica la durata di 1/32;
  - 64 che indica la durata di 1/64;
  - 128 che indica la durata di 1/128.

- *pos* indica la posizione dell'elemento relativamente al pentagramma e può assumere i valori:
  - *UprHalfStaff* indica che la posizione è sopra la terza riga del pentagramma;
  - *LwrHalfStaff* indica che la posizione è sotto la terza riga del pentagramma.
  - *OnStaffLineX* indica la posizione sulla linea X del pentagramma. X assume i valori da 1 a 5.
  - *OnSpaceX* indica la posizione sullo spazio X tra le linee del pentagramma. X assume i valori da 0 a 5. Il valore 0 e il valore 5 sono riferiti ai due spazi fuori dal pentagramma e compresi rispettivamente tra il primo taglio addizionale superiore e il primo taglio addizionale inferiore.
  - *outStaff* indica una qualunque posizione fuori dal pentagramma.
  - *insideStaff* indica una qualunque posizione dentro il pentagramma.
  - *Rest1pos* indica la posizione occupata dalla pausa di semibreve.
  - *Rest2pos* indica la posizione occupata dalla pausa di minima.

- *head* si riferisce alla testa della nota e può assumere i valori:
  - *Empty* indica che la testa è vuota;
  - *Fill* indica che la testa è piena.

- *inBeam* indica che la nota è in beam e non può avere nessun valore assegnato;

- *inAccordo* indica che la nota è in accordo e non può avere nessun valore assegnato;

- *altezzaLess* indica che l'altezza dell'elemento deve essere inferiore del valore percentuale assegnato;

- *altezzaGreat* indica che l'altezza dell'elemento deve essere maggiore del valore percentuale assegnato;

- *larghezzaLess* indica che la larghezza dell'elemento deve essere inferiore del valore percentuale assegnato;

- *larghezzaGreat* indica che la larghezza dell'elemento deve essere maggiore del valore percentuale assegnato;

- *conf* indica il valore minimo del livello di confidenza che può avere il simbolo a cui si riferisce e può assumere valori nell'intervallo 0-1;
- *inAlterazione* indica che alla nota è legata un alterazione e può essere associato un valore intero positivo che ne rappresenta il numero;
- *DotwithStaff* indica un Punto di valore collegato con una line di pentagramma e non può avere nessun valore assegnato;
- *dxMax* indica la distanza massima lungo l'asse delle ascisse;
- *dyMax* indica la distanza massima lungo l'asse delle ordinate.
- *isRepeat* indica che la barra di suddivisione della battuta è un segno di ripetizione e non può avere nessun valore associato.
- *isEnd* indica che la barra di suddivisione è una barra di fine spartito o di fine ritornello e non può avere nessun valore associato.
- *isStart* indica che la barra di suddivisione è una barra di inizio ritornello e non può avere nessun valore associato.
- *isSingle* indica che la barra di suddivisione è una barra singola (normale) e non può avere nessun valore associato.
- *isDouble* indica che la barra di suddivisione è una barra doppia e non può avere nessun valore associato.
- *notBeamed* indica che l'elemento o il simbolo non appartiene ad un gruppo note ovvero la strip di appartenenza non è compresa all'interno dei riferimenti strutturali in ingresso BEGIN e END. Non può avere nessun valore associato.
- *isBeamed* indica che l'elemento o il simbolo appartiene ad un gruppo note ovvero la strip di appartenenza è compresa all'interno dei riferimenti strutturali in ingresso BEGIN e END. Non può avere nessun valore associato.
- *isAlone* indica che l'elemento è l'unico elemento presente nella strip e non può avere nessun valore associato.
- *notAlone* indica che l'elemento non è l'unico l'elemento presente nella strip e non può avere nessun valore associato.
- *withNote* indica che nella strip esiste una NOTA e non può avere nessun valore associato.
- *withoutNote* indica che nella strip non esiste alcuna NOTA e non può avere nessun valore associato.

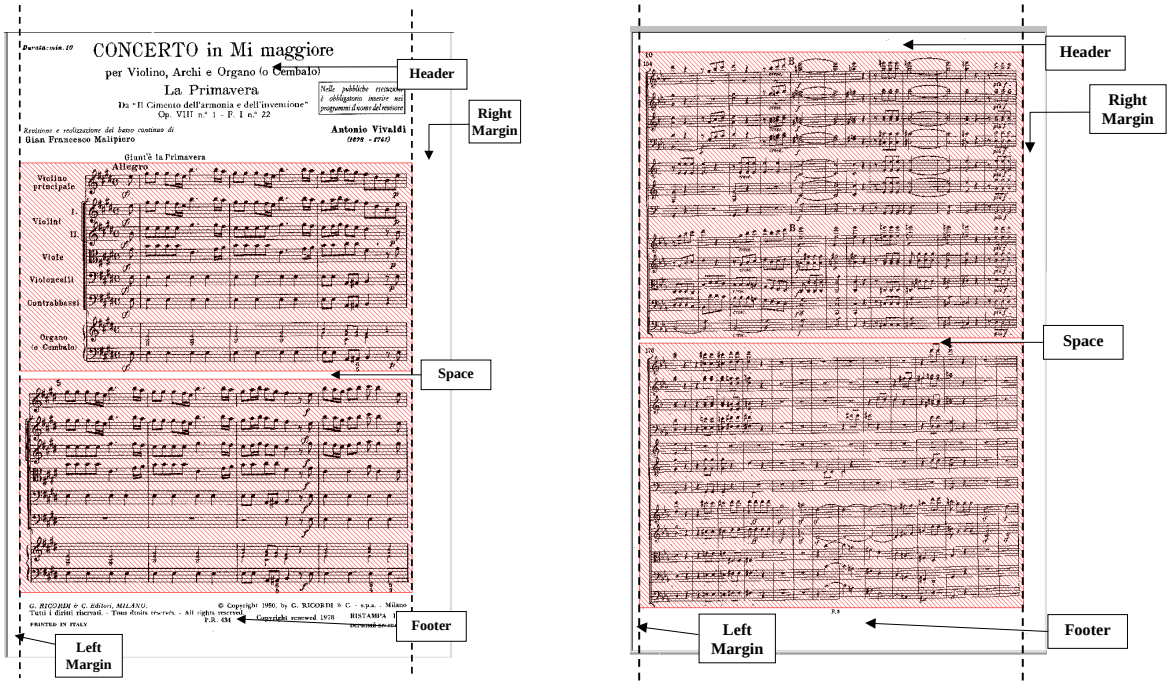
Capitolo 10

Estrazione e indicizzazione automatica di immagini di spartiti musicali

Nel corso del progetto WEDELMUSIC è stato sviluppato un segmentatore e un indicizzatore automatico di immagini digitalizzate di partiture musicali. Obiettivo di tale applicazione è generare una sequenza di immagini contenenti i pentagrammi e l'informazione necessaria affinché possano essere visualizzati nel giusto ordine all'interno del WEDELMUSIC editor e consentire la ricostruzione e la stampa delle pagine originarie da tale editor. La realizzazione di tale sistema è stata eseguita usando alcune delle tecniche OMR descritte nei capitoli precedenti.

10.1 Il processo di segmentazione automatica

Analizzando il layout degli spartiti musicali, si possono individuare due principali strutture: sistemi di pentagrammi e pentagrammi singoli. La prima struttura è quella tipica degli spartiti direttoriali e degli strumenti polifonici quali il pianoforte, l'arpa, l'organo e il clavicembalo; la seconda è quella tipica di tutti gli altri strumenti musicali. Nel primo caso i pentagrammi sono raggruppati e tenuti insieme da una parentesi, che può essere graffa oppure quadra, estesa dal primo all'ultimo pentagramma del gruppo e collocata sul margine sinistro. Nel secondo caso, invece, i pentagrammi non sono connessi tra loro. In seguito a questa distinzione sono stati realizzati due procedure di segmentazione, una per gli spartiti direttoriali (*main score*) e una per quelli delle parti singole (*single parts*) ed è l'utente, che selezionando il tipo di spartiti che desidera segmentare, guida il segmentatore nell'applicare la procedura più opportuna.



(a) (b)

Figura 10.1: Decomposizione e struttura di uno spartito direttoriale: (a) prima pagina e (b) pagina successiva

Nella figura 10.1 è riportata la struttura di una pagina di uno spartito musicale nella quale sono messe in evidenza:

- Aree relative ai margini destro (*right margin*) e sinistro (*left margin*) e gli spazi di separazione tra i pentagrammi e i sistemi di pentagrammi (*space*)
- Aree che rappresentano il frontespizio (*header*) e il fondo pagina (*footer*). Tali aree possono contenere indicazioni testuali (titolo, autore, casa editrice, raccomandazioni, ecc.).
- Aree con contenuto musicale. I pentagrammi per le parti singole o i sistemi di pentagrammi per le parti direttoriali.

Sulla base di questa struttura il segmentatore estrae tutte le aree sopra definite, ed in particolare le immagini di spartiti direttoriali saranno decomposte in sistemi, mentre quelle delle parti singole in pentagrammi (si veda figura 10.2). Alle immagini con contenuto musicale il segmentatore associa le informazioni relative:

- Al nome dello strumento oppure se si tratta di uno spartito direttoriale.
- Al numero della battuta all'inizio del pentagramma o del sistema.

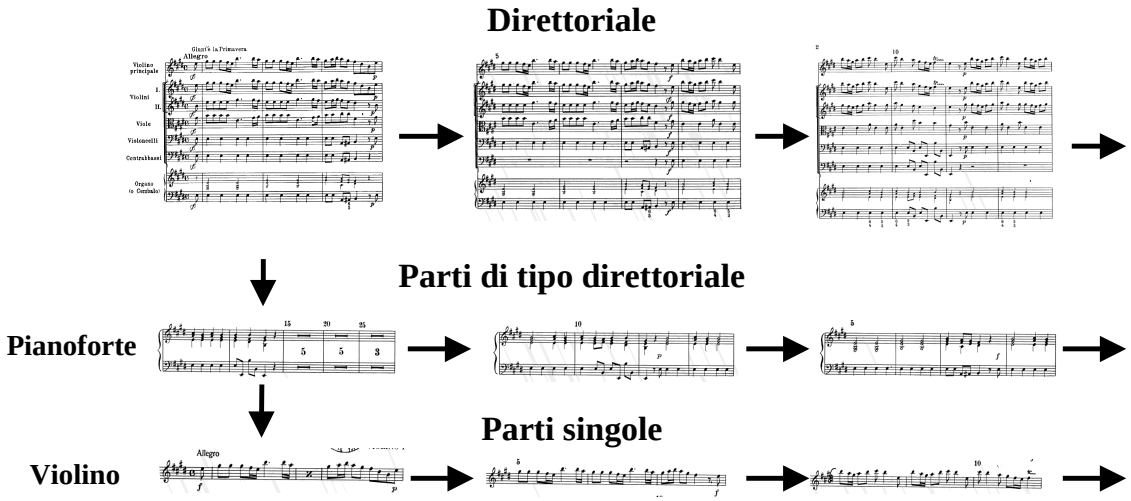


Figura 10.2: Aree con contenuto musicale

- Al numero di battute contenute nel pentagramma.
- Al numero di pagina cui fa parte.

Queste informazioni consentono di indicizzare e archiviare le immagini ottenute, raggruppandole per tipo. Le modalità di archiviazione saranno descritte in dettaglio nella sezione 10.5.

10.2 Metodi di segmentazione per uno spartito direttoriale

Con riferimento alla struttura di pagina evidenziata nel paragrafo precedente, la decomposizione avviene applicando in successione i metodi descritti di seguito.

Ricerca dei pentagrammi – L'algoritmo usato è quello presentato nel Capitolo 5. Le coppie di coordinate, che definiscono le aree di inclusione dei pentagrammi, sono calcolate senza effettuare l'estensione definita dalla relazione 5.6 ($\epsilon = 0$);

Ricerca dei margini esterni – La regolarità nella disposizione e l'allineamento dei pentagrammi consente di ricorrere al metodo delle proiezioni ed in particolare a quella verticale. Non è necessario calcolare tutto il profilo dell'immagine per tutta la larghezza, ma è sufficiente considerare il primo e l'ultimo quarto di pagina. In questi settori viene calcolata la proiezione-X e sottoposta ad una operazione di soglia, con valore pari al 2% del valore massimo della proiezione stessa. Le coordinate per i margini esterni corrispondono a quelle dei tagli più esterni individuati dopo l'operazione di soglia (si veda figura 10.3).

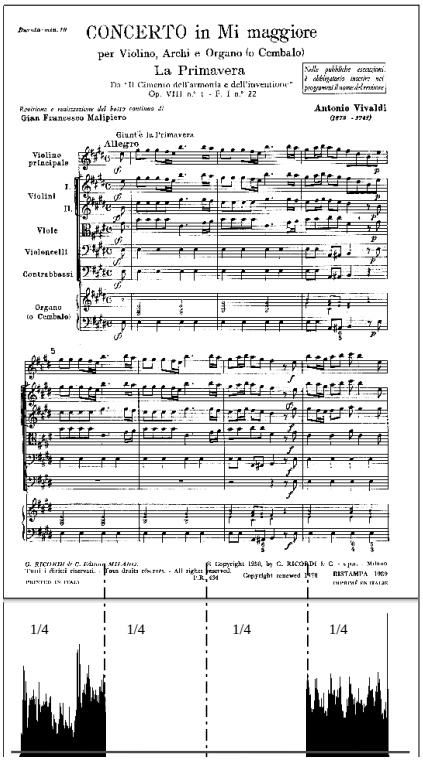


Figura 10.3: Proiezione-X e ricerca dei margini

Ricerca dei sistemi di pentagrammi – Anche in questo caso valgono le considerazioni fatte per la ricerca dei margini. A queste si deve aggiungere un'ulteriore osservazione: come è già stato detto in precedenza i sistemi sono costituiti da un insieme di pentagrammi connessi tra loro da una parentesi graffa o quadra, in più presentano una suddivisione in battute tale che la barra di suddivisione nel caso migliore attraversa tutto il sistema unendo i pentagrammi. Questi elementi grafici, in termini di contributi alla proiezione-Y, concorrono nel generare un offset e nel produrre dei profili che si estendono dal primo all'ultimo pentagramma, per ogni sistema presente. Il testo, presente soprattutto nella prima pagina di uno spartito (titolo e altre indicazioni), contribuisce nella proiezione introducendo dei profili di ampiezza variabile ed inferiore a quella generata dai sistemi. La procedura di identificazione è stata strutturata nel modo seguente:

- Calcolo della proiezione-Y di tutta la pagina e applicazione di una soglia pari all'1% del valore massimo della proiezione.
- Tra tutte le coppie di coordinate ottenute dall'operazione di soglia, si selezionano quelle che definiscono delle regioni con ampiezza tre volte superiore a quella di un pentagramma e contenenti le coordinate dei pentagrammi calcolate in precedenza.

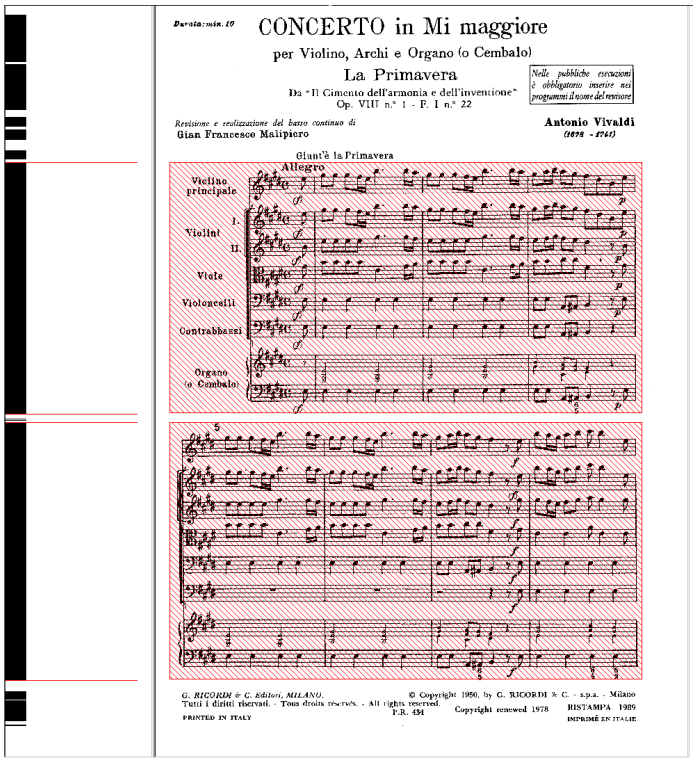


Figura 10.4: Proiezione-Y e ricerca sistemi

Nella figura 10.4 è riportato il risultato della ricerca dei sistemi di pentagrammi. Noti i margini e le coordinate di taglio per i sistemi, vengono ritagliate le aree rispecchiando così la struttura della pagina descritta in precedenza.

10.3 Segmentazione di spartiti con parti singole

Per quanto concerne gli spartiti relativi alle parti singole, l'obiettivo principale e isolare tutti i pentagrammi contenuti nella pagina. Il procedimento si avvale di alcuni dei metodi definiti per il caso di spartiti direttoriali. I passi della procedura sono i seguenti:

- Ricerca dei pentagrammi con estensione delle aree di inclusione dei pentagrammi.
- Ricerca dei margini esterni.
- Aggiustamento delle coordinate di estrazione dei pentagrammi.

L'aggiustamento delle coordinate è necessario per poter identificare l'intestazione e il fondo pagina. Le coordinate interessate sono quelle relative al primo e all'ultimo pentagramma,

che delimitano le aree da identificare. In particolare, l'aggiustamento coinvolge la coordinata superiore del primo e quella inferiore dell'ultimo pentagramma. Queste coordinate vengono ricalcolate, sulla base dell'ampiezza media che l'algoritmo di ricerca dei pentagrammi associa ai pentagrammi compresi tra i due sotto esame. Infine, noti i margini e le coordinate di taglio per ciascun pentagramma, vengono ritagliate le aree rispecchiando così la struttura della pagina descritta in precedenza.

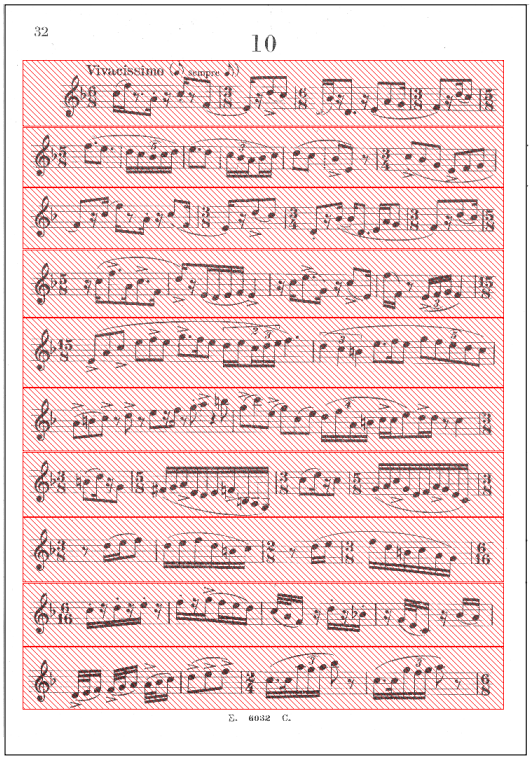


Figura 10.5: Segmentazione parti singole

10.4 Determinazione e calcolo del numero delle battute in spartiti direttoriali

Le linee di suddivisione delle battute si presentano verticali al pentagramma e la loro lunghezza è legata al tipo di spartito scelto. Nel caso di musica monofonica la scrittura musicale avviene avvalendosi di un solo pentagramma e le linee di battuta hanno una lunghezza pari all'altezza del pentagramma. Nella musica per orchestra e polifonica, invece, la scrittura musicale si presenta su più pentagrammi e le barre possono coinvolgere un pentagramma alla volta oppure gruppi di pentagrammi o tutti i pentagrammi. In questo

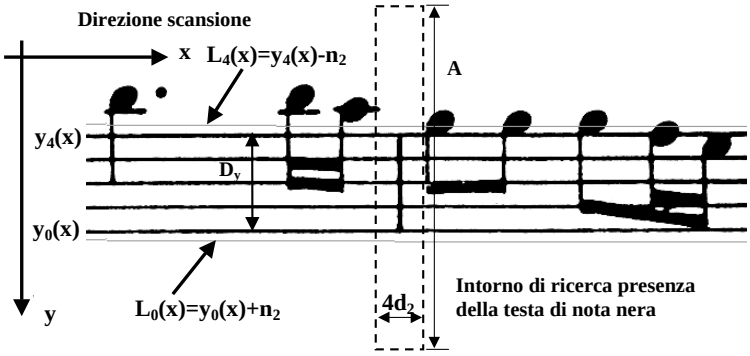


Figura 10.6: Procedura di ricerca barre di suddivisione

contesto la varietà della rappresentazione delle barre di suddivisione rende necessario la definizione di un metodo che abbia valenza generale. Il metodo si basa sul presupposto che le linee da ricercare sono verticali e che devono essere escluse quelle associate al gambo della nota. Per ogni sistema si procede nella costruzione di una particolare proiezione $P(x)$ il cui profilo sia principalmente definito dal contributo delle barre di suddivisione. Tale profilo è ottenuto iterando i seguenti passi per ogni pentagramma appartenente al sistema considerato.

Costruzione intorno di inseguimento del pentagramma Per ogni pentagramma contenuto in un segmento di immagine di altezza A si procede alla determinazione delle curve di riferimento:

$$L_4(x) = y_4(x) - n_2 \quad (10.1)$$

$$L_0(x) = y_0(x) + n_2 \quad (10.2)$$

dove:

- n_2 è lo spessore massimo di una linea di pentagramma e consente di definire un valore di tolleranza
- $y_4(x)$ e $y_0(x)$ sono due curve relative rispettivamente all'andamento dei baricentri delle linee superiore ed inferiore del pentagramma e costruite sulla base delle coordinate verticali individuate usando l'algoritmo di ricerca dei gruppi ed elementi isolati descritto nella sezione 5.3.1 del Capitolo 5.

Le due curve di riferimento consentono di definire una regione di ricerca e costituisce un intorno di inseguimento del pentagramma.

Costruzione proiezione-X relativa alle barre di suddivisione Definite le due curve di riferimento si procede nella costruzione della proiezione-X. Procedendo con una scansione a passo unitario e da sinistra a destra, si considera la colonna di pixel delimitata

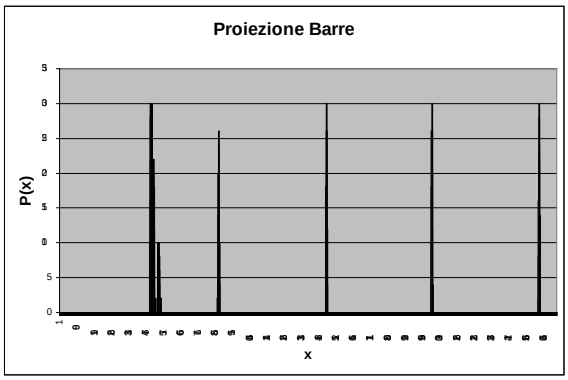


Figura 10.7: Proiezione-X relative alle barre di suddivisione per un sistema di pentagrammi

dalle curve L_0 e L_4. All'interno della colonna si ricerca una linea verticale (segmento di pixel neri) con lunghezza N uguale o maggiore del 95% della distanza $D_y = y_4(x)$ - $y_0(x)$. Se non viene rilevata alcuna linea si prosegue nella scansione senza aggiornare la proiezione $P(x)$ considerando la colonna successiva. In caso affermativo, occorre stabilire se il segmento individuato appartenga alla barra di suddivisione oppure al gambo di una nota. La discriminazione può essere condotta valutando la presenza della testa di nota nera all'interno di una finestra centrata sulla colonna in esame, ampia $4d_2$ (due teste di note) e alta A. L'algoritmo usato per condurre la ricerca è quello descritto nella sezione 5.3.2 del Capitolo 5. Se non viene rilevata alcuna testa di nota, si procede con l'aggiornamento della proiezione $P(x)$ nel modo seguente:

$$P(x+i) = P(x+i) + \alpha \qquad \text{con } i = -2, -1, 0, 1, 2 \tag{10.3}$$

con:

$$\alpha = \begin{cases} 1, & \text{se } \frac{N}{D_y} < 1; \\ 2, & \text{se } \frac{N}{D_y} \geq 1 \end{cases} \tag{10.4}$$

I valori definiti per α consentono di gestire in modo differente il contributo dovuto alle barre di suddivisione che si estendendo da un pentagramma ad un altro. Poiché, queste sono sicuramente più lunghe di D_y, per come sono stati scelti i riferimenti L_0 e L_4, il rapporto $\frac{N}{D_y}$ può assumere valori maggiori di 1, evidenziando così l'uscita della barra di suddivisione fuori dal pentagramma.

Nella costruzione della proiezione si realizza un meccanismo di sovrapposizione con i valori calcolati per il pentagramma precendente che permette di amplificare i contributi delle barre esaltando il picco nel centro stesso della barra come mostrato nella figura 10.7. Allo stesso tempo si osserva il contributo dovuto alle parentesi (graffe oppure quadre) di raggruppamento dei pentagrammi evidenziato dal primo picco.

Estrazione delle battute Al termine della costruzione della proiezione $P(x)$, si procede con l'estrazione dei picchi relativi alle barre per mezzo di una soglia definita dalla seguente relazione:

$$soglia \quad = \quad \frac{Numero\ pentagrammi\ del\ sistema}{Massimo\ della\ proiezione} \tag{10.5}$$

Per ogni coppia di coordinate, che delimitano i picchi, si considera il punto medio, in questo modo le coordinate sono centrate in corrispondenza del centro delle barre (si veda figura 10.8).

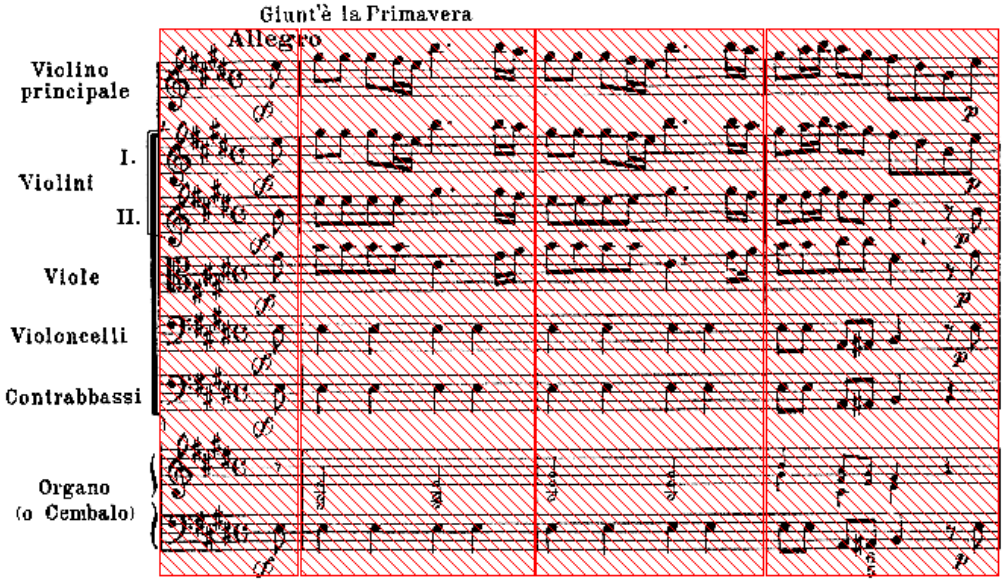
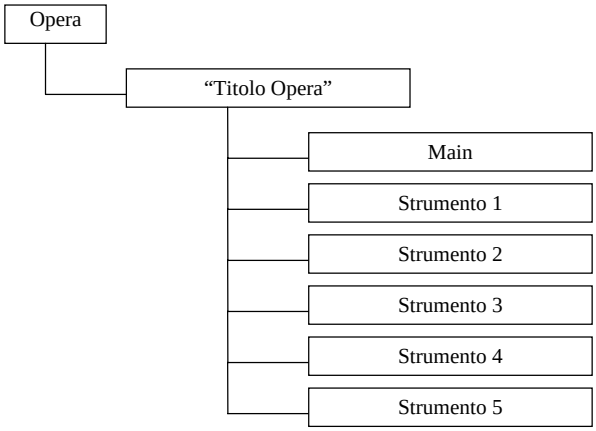


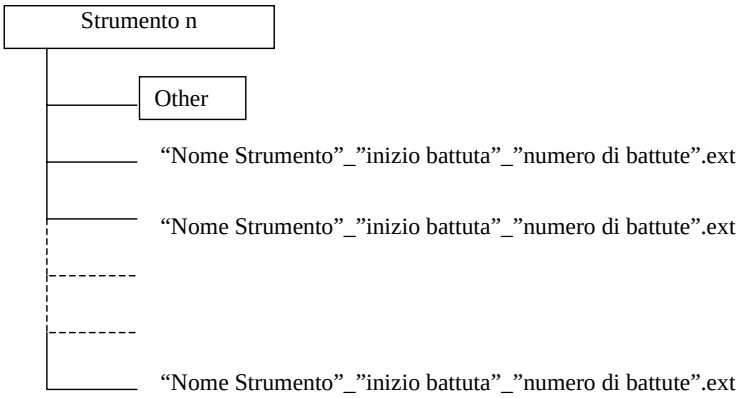
Figura 10.8: Esempio di identificazione battute

10.5 Archiviazione immagini segmentate

Tutte le immagini prodotte dal processo di segmentazione automatica vengono distribuite e memorizzate in una cartella principale chiamata *Opera*. In tale cartella viene generata una directory col nome dell'opera nella quale verranno distribuite e memorizzate le immagini provenienti dalla segmentazione. Per lo scopo, vengono generate una cartella per le immagini relative allo spartito direttoriale e una per ogni singola parte recante il nome dello strumento che essa rappresenta. In questo modo, se per esempio, l'opera è composta da un direttoriale e cinque parti singole si otterrà la seguente struttura:



Per ciascuna delle cartelle presenti è invece prevista la seguente struttura e la modalità di denominazione dei file che essa deve accogliere:



Nel caso di immagini relativi al direttoriale, il nome *Nome Strumento* viene sostituito con *Main*. L'attributo *inizio battuta* riporta il numero della prima battuta del pentagramma, mentre l'attributo *numero di battute* riporta il numero di battute presenti. La cartella **Other** consente di collezionare le immagine residue ottenute dal processo di segmentazione e rappresentano le aree relative all'intestazione, al fondo pagina, ai margini sinistro e destro ed infine agli interspazi tra i pentagrammi (o sistemi di pentagrammi). Per ciascuno di tali file è prevista una denominazione definita come segue:

- Per le immagini relative al frontespizio, il fondo pagina e i margini:

$$< Nome\ Strumento >\_< Tipo >\_< n^{\circ}\ di\ pagina > . < ext >$$

  L'attributo $< Tipo >$ può assumere uno dei seguenti valori: **Header**, **Footer**, **L_Margin** e **R_Margin**.

- Per le immagine relative agli spazi:

$< NomeStrumento >\_< Tipo >\_< n°dipagina > \_ < numerodiordine > . < ext >$

Il numero di ordine esprime la posizione dello spazio nella struttura della pagina e l'attributo $< Tipo >$ assume il valore **Space**.

Tutte le immagini di questo tipo sono elencate in un file indice memorizzato nella cartella *Other*. Lo scopo di tale file è quello di tenere traccia dell'ordine delle immagini e per ogni pagina di musica (immagine) vengono fornite le dimensioni reali (larghezza w e altezza h). Queste informazioni consentono di mantenere l'ordine logico delle immagini, quando queste verranno successivamente importate e gestite dal WEDELMUSIC editor, e di ricostruire la pagina. Un estratto di file di indice relativa ad una decomposizione di uno spartito direttoriale è riportato di seguito:

```
#1 w=2735 h=4019
main_L_Margin_001.png
main_R_Margin_001.png
main_Header_001.png
main_001_004.png
main_Space_001_001.png
main_005_004.png
main_Footer_001.png
#2 w=2735 h=3858
main_L_Margin_002.png
main_R_Margin_002.png
main_Header_002.png
main_009_004.png
main_Space_002_001.png
main_013_004.png
main_Footer_002.png
#3 w=2735 h=3858
main_L_Margin_003.png
main_R_Margin_003.png
main_Header_003.png
main_017_003.png
main_Space_003_001.png
main_020_003.png
main_Footer_003.png
```

La riga riferita dal simbolo # indica l'inizio di una nuova pagina dello spartito, riporta il numero di pagina e le dimensioni grafiche espresse da w e h.

Capitolo 11

Prestazioni del riconoscimento di spartiti monofonici

In questo Capitolo sono affrontate le problematiche relative alla valutazione delle prestazioni di un sistema di riconoscimento degli spartiti musicali. L'assenza di una terminologia standard e di un database di immagini da utilizzare come riferimento nei test non consente una facile e corretta valutazione dei risultati. Per colmare questa lacuna è stato condotto uno studio per la definizione di modelli di valutazione attraverso i quali misurare le prestazioni dei sistemi di riconoscimento OMR. I risultati ottenuti sono stati comparati con quelli ottenuti valutando con gli stessi criteri due dei più potenti software. La tipologia di spartiti considerati nei test di valutazione è limitata alla musica monofonica.

11.1 Valutazione di spartiti monofonici

Uno degli obiettivi di questa tesi è stato definire dei criteri e dei modelli di misurazione delle prestazioni dei sistemi di riconoscimento automatico degli spartiti musicali. Le soluzioni adottate prevedono di distinguere due tipologie di valutazioni:

1. Valutazione del riconoscimento dei simboli di base: l'obiettivo è misurare il riconoscimento dei singoli simboli di base considerandoli indipendenti.

2. Valutazione della ricostruzione dei simboli musicali completi e delle relazioni: l'obiettivo è valutare il riconoscimento dei simboli musicali completi ovvero considerare la sintassi musicale, per cui il simbolo musicale è la realizzazione finale della ricostruzione delle relazioni che intercorrono tra i simboli di base.

In assenza di un database di immagini di riferimento, sono state selezionate sette immagini di spartiti dall'archivio utilizzato come base di sviluppo del sistema di riconoscimento. Gli spartiti presi come campione hanno le seguenti caratteristiche:

- musica monofonica
- variazione delle dimensioni del font
- copertura dei simboli più frequenti nel repertorio di musica classica
- densità di simboli musicali variabile
- presenza di gruppi irregolari (terzine, quartine, quintine e sestine)
- presenza di simboli musicali di dimensioni ridotte (appoggiature, acciaccature, alterazioni)
- varietà di simboli di suddivisione delle battute (inzio ritornello, fine ritornello, fine spartito, etc.)
- cambi chiave e cambi di tempo

Gli spartiti sono stati sottoposti al riconoscimento e alla ricostruzione con il sistema sviluppato ed i software SharpEye 2 (distibuito da Visiv) e SmartScore 2.0.3 (distribuito da MusiTek).

11.2 Valutazione del riconoscimento attraverso i simboli di base

Traendo spunto dall'insieme dei simboli di base definito per il sistema di riconoscimento descritto nei capitoli precedenti, è stato definito un criterio di misurazione delle prestazioni basato su un'analisi di basso livello. Ad un simbolo di base è associata un'informazione elementare ed il suo corretto riconoscimento costituisce il primo passo verso la ricostruzione dei simboli musicali. Una prima valutazione delle prestazioni può essere condotta analizzando la capacità di riconoscimento delle informazioni elementari e considerando ogni simbolo di base in modo indipendente. A questo scopo è stato definito l'insieme di simboli di base analizzando gli spartiti campione e per ciascuna categoria di simboli sono state definite le seguenti metriche:

- il numero di occorrenze E_i per i simboli attesi (*Expected*) ovvero il numero simboli della categoria i-esima presenti sullo spartito originale;
- il numero di occorrenze T_i per i simboli corretti (*True*) ovvero il numero di simboli della categoria i-esima riscontrabili nello spartito originale;
- il numero di occorrenze F_i per i simboli confusi (*Fault*) ovvero il numero di simboli della categoria i-esima aggiunti e sbagliati nello spartito ricostruito.

- il numero di occorrenze M_i per i simboli mancanti (*Miss*) ovvero il numero di simboli della categoria i-esima presenti nello spartito originale, ma non in quello ricostruito;

La misurazione di ciascuna metrica è effettuata attraverso il conteggio dei simboli di base presenti nello spartito originale e successivamente ripetendo il conteggio su quello ricostruito.
Indicato con:

- N il numero di categorie di simboli di base prese in considerazione;
- $E_{tot} = \sum_{i=1}^{N} E_i$, ovvero il numero totale di simboli di base attesi,

sono stati definiti i seguenti indici di valutazione:

1. *Tasso di Riconoscimento percentuale* espresso come:

$$T_R = \frac{\sum_{i=1}^{N} T_i}{E_{tot}} \cdot 100 \tag{11.1}$$

2. *Tasso di Riconoscimento pesato percentuale*:

$$TP_R = \frac{1}{E_{tot}} \frac{\sum_{i=1}^{N} p_i T_i}{\sum_{i=1}^{N} p_i} \cdot 100 \tag{11.2}$$

   dove:

   - p_i rappresenta il grado di importanza (peso) dato a ciascun simbolo di base. I pesi sono stati considerati nell'intervallo di valori interi [1, 10].

3. *Tasso di Confusione percentuale* o *Rumore di riconoscimento*:

$$TC_R = \frac{\sum_{i=1}^{N} F_i}{E_{tot}} \cdot 100 \tag{11.3}$$

Di seguito è riportato l'insieme dei simboli di base definito per condurre la valutazione:

```
Testa di nota vuota semibreve
Testa di nota vuota minima
Testa di nota piena altre note
Pausa di semibreve
Pausa di minima
Pausa di seminima
Pausa di croma
Pausa di semicroma
```

Pausa di biscroma
Pausa di semibiscroma
Barra singola
Barra doppia
Barra di fine
Barra di inizio ritornello
Barra di fine ritornello
Diesis
Bemolle
Bequadro
Doppio diesis
Doppio bemolle
Chiave di violino
Chiave di basso
Chiave di Baritono
Chiave di tenore
Chiave di Contralto
Chiave di mezzo soprano
Chiave di soprano
Uncino singolo (durata 1/8)
Uncino doppio (durata 1/16)
Uncino triplo (durata 1/32)
Uncino quadruplo (durata 1/64)
Trave singola (durata 1/8)
Trave doppia (durata 1/16)
Trave tripla (durata 1/32)
Trave quadrupla (durata 1/64)
Punto di valore
Accento >
Numero 1
Numero 2
Numero 3
Numero 4
Numero 5
Numero 6
Numero 8
Numero 9
Numero 12

```
Numero 16
Legatura, bend
Piano indicazione di dinamica
Forte indicazione di dinamica
Prendi fiato (virgola)
Tempo C
Punto sulla nota
Corona
Mordenti
Gruppetti
Notine (appoggiatura, acciaccatura)
Trilli
Tenuto
```

L'insieme dei simboli non è esaustivo, ma è indicativo per condurre un'analisi delle prestazioni per spartiti monofonici. Può essere esteso per considerare altre tipologie di simboli oppure alcuni dei simboli definiti possono essere suddivisi aumentando la finezza della valutazione.

11.3 Valutazione della ricostruzione attraverso i simboli musicali completi

In questa sezione è proposto un criterio di valutazione che considera l'informazione musicale completa nei termini di relazioni e correttezza dei simboli musicali. In questo senso il criterio descritto vuole rappresentare un'estensione del precedente. Alcuni simboli di base sono allo stesso tempo dei simboli musicali completi, mentre altri sono dei componenti elementari di un simbolo musicale. Il riconoscimento di un singolo componente non significa che il simbolo musicale sia corretto, in quanto dal punto di vista del risultato quello che conta è che il simbolo musicale completo sia identificato nelle sue proprietà musicali. L'identificazione di una testa di nota non è indicativa del fatto che sia stata riconosciuta la nota nella sua interezza, poiché occorre aggiungere alla valutazione l'altezza e la durata, se questa ha un'alterazione ed è stata associata correttamente, se fa parte di un gruppo di note. Il riconoscimento di un accento acquista valore se questo viene associato alla nota cui appartiene, mentre la realizzazione di un gruppo di note connesse dalle travi è un'indicazione della ricostruzione della struttura e delle relazioni che esistono tra le note appartenenti al gruppo. Queste sono alcune delle considerazioni che giustificano il passaggio ad una valutazione che non si limiti solo alla misurazione del riconoscimento dei simboli considerati indipendenti, perché solo alcuni possono considerarsi tali, altri invece

concorrono nel generare relazioni più complesse. Pertanto, nell'ottica di una valutazione delle prestazioni è necessario considerare i legami reciproci tra simboli musicali che si ottengono dalla ricostruzione.

È stato individuato un insieme di valutazioni costituito da simboli musicali completi e relazioni per ciascuno dei quali, è stato definito il seguente insieme di metriche:

1. il numero di occorrenze N_i per i simboli o le relazioni tra simboli attese (*Expected*) ovvero i simboli o le relazioni presenti sullo spartito originale relative alla valutazione i-esima,
2. il numero di occorrenze $n_t^{(i)}$ di simboli o di relazioni corretti/e (True) relative alla valutazione i-esima e riferite allo spartito ricostruito,
3. il numero di occorrenze $n_a^{(i)}$ di simboli o di relazioni aggiunti/e (Add) relative alla valutazione i-esima e riferite allo spartito ricostruito,
4. il numero di occorrenze $n_f^{(i)}$ di simboli o di relazioni non corretti/e (Fault) relative alla valutazione i-esima e riferite allo spartito ricostruito,
5. il numero di occorrenze $n_m^{(i)}$ di simboli o di relazioni mancanti (Miss) relative alla valutazione i-esima e riferite allo spartito ricostruito,

e per le quali sussiste la seguente relazione:

$$N_i = n_t^{(i)} + n_f^{(i)} + n_m^{(i)} \tag{11.4}$$

La misurazione è effettuata attraverso il conteggio dei simboli o delle relazioni attesi per lo spartito originale e successivamente il conteggio dei simboli e delle relazioni ottenute o omesse nello spartito ricostruito.

Indicato con:

- M il numero totale di simboli e di relazioni che costituiscono l'insieme di valutazione,
- $E_{tot} = \sum_{i=1}^{M} N_i$, ovvero il numero totale delle relazioni e dei simboli musicali attesi,

si definiscono i seguenti indici di valutazione:

1. *Tasso di Ricostruzione percentuale* espresso come:

$$T_{Ric} = \frac{\sum_{i=1}^{M} n_t^{(i)}}{E_{tot}} \cdot 100 \tag{11.5}$$

2. *Tasso di Ricostruzione pesato percentuale*:

$$TP_{Ric} = \frac{1}{E_{tot}} \frac{\sum_{i=1}^{M} p_i n_t^{(i)}}{\sum_{i=1}^{M} p_i} \cdot 100 \tag{11.6}$$

dove:

- p_i rappresenta il grado di importanza (peso) dato a ciascun simbolo di base. I pesi sono stati considerati nell'intervallo di valori interi [1, 10].

3. *Errore di ricostruzione percentuale*:

$$E_{Ric} = \frac{\sum_{i=1}^{M} n_f^{(i)} + n_a^{(i)} + n_m^{(i)}}{E_{tot}} \cdot 100 \tag{11.7}$$

Il criterio descritto valuta il simbolo musicale e le relazioni che esso ha con altri simboli. Un simboli musicale o la relazione è corretta se sono corretti i simboli di base e i simboli musicali che concorrono alla realizzazione della relazione. Per cui si ha una valutazione più severa rispetto al criterio basato sui soli simboli di base, è quindi prevedibile un calo nei valori degli indici di prestazione. L'insieme di valutazione e il significato specifico per ciascuna metrica sono state definite come segue.

Note con pitch e durata – L'obiettivo è valutare la correttezza della ricostruzione delle note considerando le proprietà: altezza e durata. Una nota è considerata non corretta se uno dei due requisiti non sono soddisfatti. Una nota è considerata mancante se è assente, è considerata aggiunta se non è presente nello spartito.

Pause – L'obiettivo è valutare il riconoscimento delle pause. Una pausa è considerata corretta se è ha il valore di durata corretto. È considerata mancante se è assente, è considerata aggiunta se non è presente nello spartito, non corretta se la durata è sbagliata.

Note con alterazione associata – L'obiettivo è valutare l'associazione di un'alterazione (diesis, bemolle, bequadro, etc.) alla nota. L'associazione è considerata corretta se il simbolo di alterazione è corretto, mancante se è assente, non corretta se non è il simbolo di alterazione richiesto, aggiunta se è stato associato ad una nota che non ha alterazione.

Gruppi di note connesse con travi – L'obiettivo è valutare la ricostruzione dei gruppi di note collegate con le travi (beam). La valutazione considera la realizzazione del raggruppamento per cui è considerato corretto se è stato realizzato il gruppo, mancante se non è presente la connessione tra le note, aggiunto se non è previsto la connessione delle note. Il caso di non correttezza non è stato definito in quanto è legata alla presenza delle note e valutata dalla metrica sulle note.

Frazione e cambi tempo – È valutata l'identificazione e la rappresentazione della frazione indicativa del tempo della battuta. L'obiettivo è valutare il riconoscimento dei caratteri numerici che costituiscono la frazione. È considerata corretta se il numeratore e denominatore sono esatti, mancante se è assente, non corretta se uno dei due numeri è errato, aggiunta se non è prevista.

Armatura di chiave e cambi armatura – È valutata l'identificazione e la rappresentazione dell'armatura di chiave indicativa della tonalità. È considerata corretta se il numero di diesis o bemolli (e gli eventuali bequadri nei cambi di tonalità durante l'esecuzione) è esatto, mancante se è l'armatura è assente, non corretta se il numero dei simboli è errato, aggiunta se non è prevista.

Simboli sopra/sotto la nota/pausa – L'obiettivo è valutare l'identificazione è l'assegnazione dei simboli musicali di accento, staccato, corona, tenuto, gruppetti, mordenti, trilli. Il simbolo è ritenuto corretto se è quello previsto, mancante se è assente, non corretto se non è errato, aggiunto se non è previsto.

Notine – L'obiettivo è valutare la capacità di riconoscimento delle note di abbellimento (acciaccature, appoggiature e gruppetti di note). In questo caso si considera la nota completa nei termini di tipo di abbellimento, altezza e durata. Per cui la nota è corretta se sono soddisfatti i tre requisiti, non corretta se uno dei requisiti non è soddisfatto, aggiunta se non è prevista, mancante se è non è stata riconosciuta.

Legature e bend – È valutata la ricostruzione dei simboli ad evoluzione orizzontale: legature, bend e legature di riferimento per i gruppi irregolari. Il simbolo è corretto se la nota di inizio e di fine sono le note reali di partenza e arrivo del simbolo oppure nel caso di legature di riferimento è posta nella posizione corretta. Non è corretto se non sono soddisfatte le condizioni precedenti, aggiunto se non è previsto, mancante se è assente.

Punti di valore – L'obiettivo è valutare l'associazione dei punti di valore alle note. L'associazione è corretta se il punto di valore è associato alla nota, mancante se il punto non esiste, aggiunta se il punto non è previsto. La condizione di assegnazione non corretta non è definita in quanto equivalente alla aggiunta.

Chiavi – Valuta il riconoscimento delle chiavi e dei cambi chiave. La presenza di simboli uguali per indicare chiavi diverse comporta la considerazione della posizione relativa al pentagramma. Ad esempio la chiave di Do può essere collocata in quattro posizioni e ciascuna rappresenta rispettivamente la chiave di tenore, contralto, mezzo soprano e soprano. Per cui il simbolo è corretto se oltre a rappresentare il simbolo della chiave e collocato nella posizione corretta. Il simbolo è non corretto se uno dei dure requisiti non è soddisfatto, mancante se è assente, aggiunto se non è previsto.

Gruppi irregolari – Valuta l'identificazione dei gruppi irregolari ovvero la capacità di riconoscere terzine, quartine, etc. Si ricorda che i gruppi irregolari sono insieme di note,

non necessariamente connesse da simboli di travi, identificate dalla presenza di un'indicazione numerica che definisce se il gruppo è una terzina, quartina, etc. L'identificazione è corretta se il gruppo di note è effettivamente un gruppo irregolare e ha associato la relativa indicazione numerica. Non è corretta se l'indicazione numerica è sbagliata, è mancante se non è stato individuato il gruppo irregolare, è aggiunto se il gruppo individuato non è un gruppo irregolare.

Numero di battute nella pagina – Valuta il numero di battute in relazione alla barre verticali di suddivisione. Si considerano corrette le battute che sono identificabili sullo spartito originale, mancante se non vi è corrispondenza con l'originale (il mancato riconoscimento di una barra di suddivisione può portare alla fusione di due battute e quindi alla perdita di una), aggiunta se non è individuabile un'analoga sullo spartito originale. La condizione di non corretta è inclusa nella condizione di mancante.

Numero di pentagrammi – Valuta il numero di pentagrammi in relazione alla struttura dello spartito. Per cui il numero corretto di pentagrammi indica anche la distribuzione corretta delle battute. Un pentagramma è mancante se non è stato identificato, è aggiunto se non è nello spartito originale. La non correttezza di un pentagramma è associata al numero di linee e spazi.

Queste insieme di valutazione non è da considerarsi esaustivo per la valutazione di tutti gli spartiti musicali. Esso è comunque in grado di caratterizzare gli spartiti monofonici del repertorio classico e di descrivere gli aspetti, le relazioni e i simboli più importanti presenti nella scrittura musicale.

11.4 Considerazioni sui pesi

Con l'introduzione dei pesi si vuole esprimere un'importanza del riconoscimento dei simboli in funzione della loro presenza e rilevanza nella partitura. È ammesso che il sistema introduca degli errori, ma è richiesto che questi siano limitati ai simboli di minore importanza dal punto di vista del lavoro di editing necessario per la correzione. Per questa ragione, ad esempio, una testa di nota o la nota completa o una pausa ha una rilevanza nel riconoscimento superiore rispetto ad un simbolo di mordente, poiché è il simbolo musicale principale. Un altro aspetto da tenere in considerazione è il formato utilizzato per la rappresentazione della notazione musicale in forma simbolica (si veda il paragrafo 1.8). Alcune limitazioni sul riconoscimento possono dipendere dalla capacità del formato di rappresentare i simboli musicali. È quindi introdotta una discriminazione sui simboli musicali gestiti dal linguaggio di rappresentazione della notazione scelto.

Da un punto di vista operativo, la definizione dei valori da assegnare ai pesi può essere realizzata su base statistica attraverso un'indagine condotta su un numero significativo di utenti in grado di definire quali simboli siano più importati da riconoscere sulla base dell'esperienza nell'utilizzo di programmi di video scrittura e di riconoscimento ottico, e del successivo lavoro di editing per le correzioni.

11.5 Guadagno nel riconoscimento automatico

Dalle considerazioni effettuate sui pesi, emerge che la valutazione delle prestazioni di un sistema di riconoscimento, non è solo un'operazione di conteggio di simboli, ma deve tenere in considerazione il lavoro di correzione da fare al termine del riconoscimento automatico. Il lavoro di correzione è effettuato attraverso l'editing e può richiedere molte risorse se i simboli da correggere, da eliminare o da aggiungere sono coinvolti in relazioni complesse. In particolare il costo maggiore è imputabile alla ricerca e verifica della correttezza dei simboli musicali, delle relazioni e delle strutture, per capire quale azione intraprendere. Nasce allora la domanda se convenga o no utilizzare un sistema di riconoscimento automatico e quando questo è effettivamente conveniente. Queste considerazioni trovano una formalizzazione definendo un *guadagno di riconoscimento* nel modo seguente:

$$G = \frac{C_0}{C_{OMR}} \tag{11.8}$$

dove:

- C_0 esprime il costo necessario per trascrive uno spartito in formato elettronico per mezzo di un editor di video scrittura musicale;
- C_{OMR} esprime il costo di correzione richiesto per trascrivere uno spartito in formato elettronico adottando un sistema di riconoscimento ottico degli spartiti musicali.

Affiché si abbia convenienza è necessario che il rapporto sia maggiore dell'unità o più in generale di un valore desiderato $\gamma \geq 1$. Pertanto la (11.8) può essere riscritta come:

$$G = \frac{C_0}{C_{OMR}} \geq \gamma \tag{11.9}$$

Per valutare la ricostruzione di uno spartito è necessario comparare lo sparito originale con quello ottenuto per ricercare errori, omissioni o introduzione involontarie di simboli musicali e di relazioni tra i simboli. Il controllo è necessario anche se il riconoscimento fosse del 100% come operazione di validazione della ricostruzione, poiché non vi è garanzia che ciò che è stato ricostruito sia corretto. Pertanto tutti i simboli musicali e le relazioni devono

essere controllate. Se ad ogni operazione di correzione attribuiamo un costo, questo dovrà tenere in considerazione il costo imputabile alla ricerca e alla valutazione della correttezza o meno dei simboli musicali riconosciuti. Sulla base di queste considerazioni, si definiscono i seguenti costi:

- C_i il *costo medio di editing* per inserire un simbolo musicale o creare una relazione tra simboli attraverso l'editor di video scrittura musicale;
- C_v il *costo medio di verifica* e ricerca dei simboli e delle relazioni
- $C_f = \hat{C}_f + C_v$ il *costo medio di correzione* di un errore di riconoscimento, è dato dal *costo proprio di correzione* $\hat{C}_f$ più il costo medio di verifica.
- $C_a = \hat{C}_a + C_v$ il *costo medio di cancellazione* di un simbolo o una relazione introdotta dal riconoscimento, è dato dal *costo proprio di cancellazione* $\hat{C}_a$ più il costo medio di verifica.
- $C_m = \hat{C}_m + C_v$ il *costo medio di ripristino* di un simbolo od una relazione non riconosciuta, è dato dal *costo proprio di ripristino* $\hat{C}_m$ più il costo medio di verifica.
- $C_t = C_v$ il costo necessario per validare le relazioni ed i simboli musicali corretti e si può considerare uguale al solo costo di verifica.

I costi $\hat{C}_f$, $\hat{C}_a$ e $\hat{C}_m$ in quanto costi di editing possono essere resi proporzionali al costo medio di editing C_i, introducendo le costanti di proporzionalità α_f, α_a e α_m. I costi sono quindi espressi come segue:

$$\hat{C}_f = \alpha_f C_i \qquad \hat{C}_a = \alpha_a C_i \qquad \hat{C}_m = \alpha_m C_i \tag{11.10}$$

Sulla base delle definizioni introdotte i costi C_0 e C_{OMR} sono definiti nel modo seguente:

- $C_0 = kNC_i$ con N numero di simboli musicali e relazioni da trascrivere e $k > 1$ costante di proporzionalità che tiene in considerazione attività di editing aggiuntive (la definizione della pagina, il numero di pentagrammi, il numero di battute, etc.) e la lettura del simbolo musicale da scrivere.
- $C_{OMR} = C_f + C_a + C_m + C_t$

Indicando con:

- n_t il numero di relazioni e simboli correttamente riconosciuti;
- n_f il numero di relazioni e simboli non correttamente riconosciuti;
- n_a il numero di relazioni e simboli aggiunti ma non esistenti sullo spartito originale;

- n_m il numero di relazioni e simboli non riconosciuti e quindi mancanti;

la relazione (11.9) assume la seguente forma:

$$\frac{kNC_i}{C_f n_f + C_a n_a + C_m n_m + C_t n_t} \geq \gamma \tag{11.11}$$

e considerando che N esprime il numero di simboli musicali e relazioni da trascrivere, si ottiene che:

$$N = n_t + n_f + n_m \tag{11.12}$$

Sostituendo le definizioni dei singoli costi e utilizzando le relazioni (11.10) e (11.12) si ottiene la relazione:

$$\frac{kNC_i}{(n_f\alpha_f + n_a\alpha_a + n_m\alpha_m)C_i + (N + n_a)C_v} \geq \gamma \tag{11.13}$$

Nel caso di riconoscimento ideale del 100% la (11.13) diventa:

$$\frac{kC_i}{C_v} \geq \gamma \tag{11.14}$$

da cui si deduce che la convenienza di un sistema di riconoscimento si ha quando:

$$C_v \leq \frac{kC_i}{\gamma} \tag{11.15}$$

Pertanto, se si desidera un guadagno pari a k allora il costo di verifica per simbolo o relazione deve risultare inferiore al costo di editing. Nel caso migliore, il limite alla convenienza è legato al solo costo di verifica ed è vincolato da grandezze che dipendono dall'editor utilizzato e dalla modalità di video scrittura. Tale limite può essere indicato con:

$$C_v^0 = \frac{kC_i}{\gamma} \tag{11.16}$$

Nel caso reale la relazione per il costo di verifica è dato da:

$$C_v \leq \frac{kNC_i}{\gamma(N + n_a)} - \frac{(n_f\alpha_f + n_a\alpha_a + n_m\alpha_m)C_i}{N + n_a} \tag{11.17}$$

ed infine introducendo i tassi percentuali:

$$k_f = \frac{n_a}{N} \qquad k_m = \frac{n_a}{N} \qquad k_a = \frac{n_a}{N} \tag{11.18}$$

ed utilizzando la (11.16), si ottiene:

$$C_v \leq C_v^0 \frac{1}{(1+k_a)} - \frac{(k_f\alpha_f + k_a\alpha_a + k_m\alpha_m)C_i}{1+k_a} \tag{11.19}$$

Nella relazione completa sono presenti tutti i contributi legati al riconoscimento. Pertanto ogni contributo migliorativo al riconoscimento comporta un aumento della tolleranza sul costo di verifica fino al limite ideale ottenuto per un riconoscimento del 100%. Ma ogni contributo migliorativo sull'editing abbassa il limite della convenienza. È interessante quantificare la percentuale di riconoscimento media che un sistema deve avere affichè si possano ottenere dei benefici nell'utilizzo del riconoscimento automatico. L'individuazione di tale valore è legata alla stima dei costi e dei parametri introdotti, che possono essere diversi da editor ad editor. Tali valori possono essere stimati su base statistica quantificando i costi medi per trascrivere uno spartito.

Infine, aumentare la percentuale di riconoscimento anche di pochi decimi percentuali richiede molte risorse a causa del problema della scalabilità dei sistemi di riconoscimento, che in certi casi può significare lo sviluppo di un nuovo sistema. Lo sforzo, allora, può essere diretto alla realizzazione di strumenti in grado di agevolare il controllo dei simboli musicali (segnalare le battute temporalmente inconsistenti, inserire dei marcatori individuabili velocemente nei punti in cui si ritiene possa esserci un errore o un simbolo mancante) e ridurre così il tempo di verifica. Questi strumenti possono essere inseriti direttamente nel sistema di riconoscimento o nell'editor musicale utilizzato.

11.6 Risultati

In questa sezione, sono riportati i risultati ottenuti con il sistema di riconoscimento descritto e la misura delle prestazioni, ottenuta attraverso i due metodi descritti. Sono stati effettuati dei test di comparazione misurando con gli stessi criteri i risultati ottenuti con due software ritenuti più potenti: SharpEye 2 e SmartScore. Il numero di immagini campione è pari a sette e sono indicate con il riferimento *Esempio* e il numero associato. Per ogni esempio è stato riportato lo spartito originale (Figure 11.1, 11.3, 11.5, 11.7,11.9, 11.11, 11.13) e la rappresentazione mediante l'editor WEDELMUSIC dello spartito ricostruito dal sistema O^{3}MR (Figure 11.2, 11.4, 11.6, 11.8,11.10, 11.12, 11.14).

Valutazione del riconoscimento attraverso i simboli di base

Nelle Figure 11.15, 11.16 e 11.17 sono riportate rispettivamente le tabelle per SmartScore, SharpEye 2 e O^{3}MR contenenti i risultati delle misurazione per ciascun esempio considerato. La tabella è stata suddivisa in sotto tabelle e ciascuna in due sezioni: una relativa ai valori attesi (colonna **Attesi**) per ciascun simbolo di base considerato e ottenuti dal conteggio sullo spartito originale ed una relativa ai valori conteggiati nello spartito ricostruito, indicato con **Trovati**. In particolare in questa sezione sono riportati:

- nella colonna **Totale** il numero di occorrenze per ciascun simbolo di base,
- nella colonna **True** il numero di occorrenze per i simboli di base corretti (T_i).
- nella colonna **Fault** il numero di occorrenze per i simboli di base confusi (F_i).
- nella colonna **Miss** il numero di occorrenze per i simboli mancanti (M_i).

Nella colonna **Pesi** sono riportati i valori dei pesi associati a ciascun simbolo di base. Nell'ultima riga (**TOTALE**) della tabella sono riportati il numero di occorrenze totali per ciascuna colonna. Nella Figura 11.19 sono riportati i valori degli indici di prestazione valutati per ogni esempio e per ogni software.

Osservando le tabelle si ha che:

- SmartScore è soggetto alla perdita di informazione, infatti in ogni esempio il numero di simboli mancanti è maggiore dei simboli confusi. Il simbolo di accento non è stato mai riconosciuto, si riscontrano difficoltà nell'identificazione dei numeri delle frazioni relative alle indicazioni del tempo, delle pause di durata inferiore all'ottavo e delle legature. Per quanto riguarda la confusione di simboli si osserva la tendenza ad aggiungere note di semibreve, punti di valore e simboli di staccato. Alcuni simboli sembrano non gestiti: mordenti, trilli e corona.
- SharpEye 2 mostra una prestazione nel riconoscimento superiore, ma allo stesso tempo una tendenza alla perdita di informazione. Rispetto ai due software, però, aggiunge un numero minore di simboli. Da sottolineare la capacità di ricostruzione delle legature e la gestione dei simboli di trillo, corona e delle notine anche se non sempre corrette. Alcune difficoltà sono state riscontrare nell'identificazione della chiave di Baritono.
- O^{3}MR è soggetto a perdita di informazione, in particolare questa è dovuta alla mancata gestione dei simboli di corona, di trillo e di staccato nella versione attuale, e la difficoltà nel riconoscimento delle legature alle quali è imputabile l'alto valore di confusione.

Dai grafici in Figura 11.19, si deduce che SharpEye 2 è il programma che ha realizzato le prestazioni migliori. In relazione al *Tasso di Riconoscimento pesato percentuale* e al *Tasso di Riconoscimento percentuale*, SharpEye 2 ha subito una diminuizione delle prestazioni nell'Esempio 6 sia rispetto a SmartScore che O^{3}MR (si veda Figura 11.11) dovuta principalmente alle difficoltà manifestate nel riconoscere correttamente le notine, questa perdita è compensata dall'alto *Tasso di Confusione* realizzato nello stesso esempio da SmartScore e O^{3}MR. Per quanto riguarda il sistema sviluppato, in generale mostra capacità simili a SmartScore ed una tendenza ad aggiungere meno simboli, solo in un caso si è dimostrato migliore di SharpEye 2 come evidenziano i valori dei Tassi di Riconoscimento sia pesato che percentuale dell'Esempio 5, tuttavia è penalizzato da un Tasso di Confusione di poco superiore a SharpEye 2. Infine, tutti e tre i software hanno avuto problemi con lo spartito dell'Esempio 6 come dimostrano i valori degli indici.

Valutazione della ricostruzione attraverso i simboli musicali completi

Nella Figura 11.18 sono riportate rispettivamente le tabelle per SmartScore, SharpEye 2 e O^{3}MR contenenti i risultati delle misurazioni per ciascun esempio considerato. Ogni tabella è stata suddivisa in sotto tabelle e in ciascuna sono riportati:

- nella colonna **Attesi** il numero di occorrenze attese conteggiato nello spartito originale per ciascun simbolo musicale o relazione tra i simboli (N_i),
- nella colonna **True** il numero di occorrenze conteggiato nello spartito ricostruito per i simboli musicali o le relazioni corretti ($n_t^{(i)}$),
- nella colonna **Add** il numero di occorrenze conteggiato nello spartito ricostruito per i simboli musicali o relazioni aggiunti ($n_a^{(i)}$),
- nella colonna **Fault** il numero di occorrenze per i simboli musicali o le relazioni sbagliate ($n_f^{(i)}$),
- nella colonna **Miss** il numero di occorrenze per i simboli musicali o le relazioni mancanti ($n_m^{(i)}$).

Nella colonna **Pesi** sono riportati i valori dei pesi associati a ciascun voce dell'insieme di valutazione. Nell'ultima riga (**TOTALE**) della tabella sono riportati il numero di occorrenze totali per ciascuna colonna. Nella Figura 11.20 sono rappresentati gli andamenti dei valori degli indici di prestazione valutati per ogni esempio e per ogni software.

Osservando le tabelle si ha che:

- SmartScore ha la tendenza ad introdurre degli errori di ricostruzione delle note ed aggiungere note non previste. Incontra delle difficoltà nell'identificazione dei gruppi

irregolari (terzine, quartine, etc.), in particolare la tendenza è quella di considerare irregolari gruppi di note che non lo sono. Questo fatto lascia presupporre che il sistema implementi un meccanismo di valutazione dei gruppi irregolari basata sull'analisi della durata della battuta o sulla struttura del gruppo di note e quindi trascurando o non identificando il simbolo numerico che caratterizza il gruppo irregolare. Sono confermate i problemi di ricostruzione delle legature e sono evidenziate le difficoltà nell'identificazione dei cambi di tempo e la ricostruzione dell'armatura di chiave.

- SharpEye 2 mostra una scarsa attitudine all'introduzione di note e nel riconoscere i gruppi irregolari, tuttavia rispetto a SmartScore non vi sono gruppi irregolari aggiunti. Questo lascia presupporre che l'identificazione dei gruppi irregolari sia condotto riconoscendo l'identificativo numerico. Sono confermati i problemi anche se minori nella ricostruzione delle legature e di identificazione delle notine, in quest'ultimo caso l'attitudine è quella di non discriminare le appoggiature dalle acciaccature ed a considerare un solo simbolo per la notina: l'appoggiatura.

- i limiti manifestati da O^{3}MR sono riconducibili alla ricostruzione delle legature, alla non gestione dei gruppi irregolari e delle notine. È osservabile una tendenza all'introduzione di pause, questo comportamento è dovuto ad errori di riconoscimento nei simboli di base generati dalla segmentazione e classificati come pause. Infine, come nella valutazione basata sui simboli di base, è confermata la tendenza ad introdurre meno simboli rispetto a SmartScore.

I grafici in Figura 11.20, mostrano che in generale SharpEye 2 è il programma che ha realizzato le prestazioni migliori. In relazione al *Tasso di Ricostruzione pesato percentuale* e al *Tasso di Ricostruzione percentuale*, SharpEye 2 ha subito una perdita di prestazione nell'Esempio 6 (si veda Figura 11.11) a vantaggio di SmartScore, le cause sono identificabili principalmente nelle difficoltà manifestate nel riconoscere correttamente le notine e nell'avere omesso un numero di note superiore a SmartScore, questa perdita è comunque compensata dall'alto *Tasso di Confusione* realizzato nello stesso esempio da SmartScore e O^{3}MR a causa del maggior numero di simboli aggiunti. Per quanto riguarda il sistema sviluppato, anche con la valutazione sui simboli musicali e sulle relazioni è evidenziato un comportamento simile a SmartScore.

11.6.1 Valutazione del sistema O^{3}MR

Dalle considerazioni precedenti il sistema sviluppato nella versione attuale presenta alcuni problemi nella ricostruzione delle legature, non gestisce simboli come lo staccato, la corona, il trillo, il mordente, le notine e non riconosce i gruppi irregolari. Tuttavia, uno degli obiettivi principali è stato quello di realizzare un sistema robusto nell'identificazione delle

Note con pitch e durata	SmartScore	SharpEye 2	O^{3}MR
TT_{Ric}	95.68%	96.67%	97.24%
TF_{Ric}	2.29%	1.25%	1.72%
TM_{Ric}	2.03%	2.08%	1.04%
TA_{Ric}	2.44%	0.21%	1.20%

Tabella 11.1: Tassi di Ricostruzione percentuali per le note

Pause	SmartScore	SharpEye 2	O^{3}MR
TT_{Ric}	38.54%	81.77%	94.79%
TF_{Ric}	0.00%	2.60%	0.52%
TM_{Ric}	61.46%	15.63%	4.69%
TA_{Ric}	8.85%	0.00%	8.85%

Tabella 11.2: Tassi di Ricostruzione percentuali per le pause

note e delle pause in partiture monofoniche. Per valutare le prestazioni ristrette a queste categorie di figure si riportano i tassi percentuali complessivi valutati su tutti gli esempi (utilizzando i valori numerici riportati nelle tabelle di Figura 11.18) e definiti nel modo seguente:

$$
\begin{aligned}
TT_{Ric} &= \frac{\sum_{i=1}^{7} n_t^{(i)}}{\sum_{i=1}^{7} N_i} \cdot 100 && \text{Tasso di simboli corretti (True)} \\
TF_{Ric} &= \frac{\sum_{i=1}^{7} n_f^{(i)}}{\sum_{i=1}^{7} N_i} \cdot 100 && \text{Tasso di simboli non corretti (Fault)} \\
TM_{Ric} &= \frac{\sum_{i=1}^{7} n_m^{(i)}}{\sum_{i=1}^{7} N_i} \cdot 100 && \text{Tasso di simboli mancanti (Miss)} \\
TA_{Ric} &= \frac{\sum_{i=1}^{7} n_a^{(i)}}{\sum_{i=1}^{7} N_i} \cdot 100 && \text{Tasso di simboli aggiunti (Add)}
\end{aligned}
$$

I valori ottenuti (si veda le Tabelle 11.1 e 11.2) mettono in evidenza un'ottima prestazione nella gestione delle note e delle pause, in particolare è da sottolineare l'elevata capacità nel riconoscimento delle pause (TT_{Ric}) con una differenza del 13.02% rispetto a SharpEye 2 e del 56.25% rispetto a SmartScore. Anche se meno accentuato è presente uno scostamento nel riconscimento delle note pari allo 0.57% con SharpEye 2 e all'1.56% con SmartScore. Infine, in relazione al successivo lavoro di correzione si nota che il sistema O^{3}MR presenta un comportamento comparabile, in alcuni casi migliore, di SharpEye 2. Le pause aggiunte dal sistema O^{3}MR (TA_{Ric}) sono imputabili ad un'errata segmentazione che ha prodotto dei simboli di base non corretti o non appartenenti all'insieme definito. Questo ha inciso sulla successiva fase di riconoscimento e di ricostruzione.

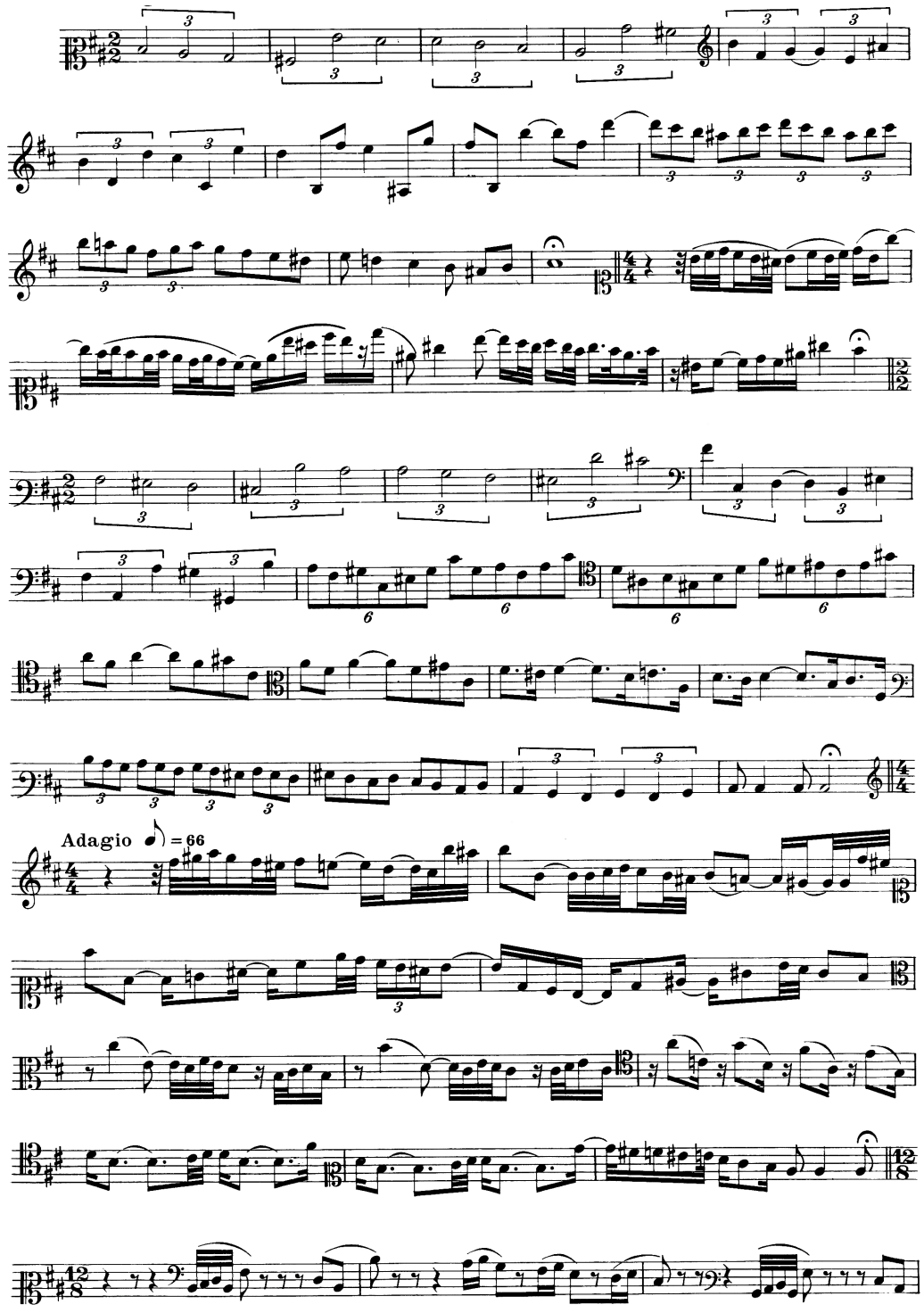


Figura 11.1: Esempio 1, immagine dello spartito originale

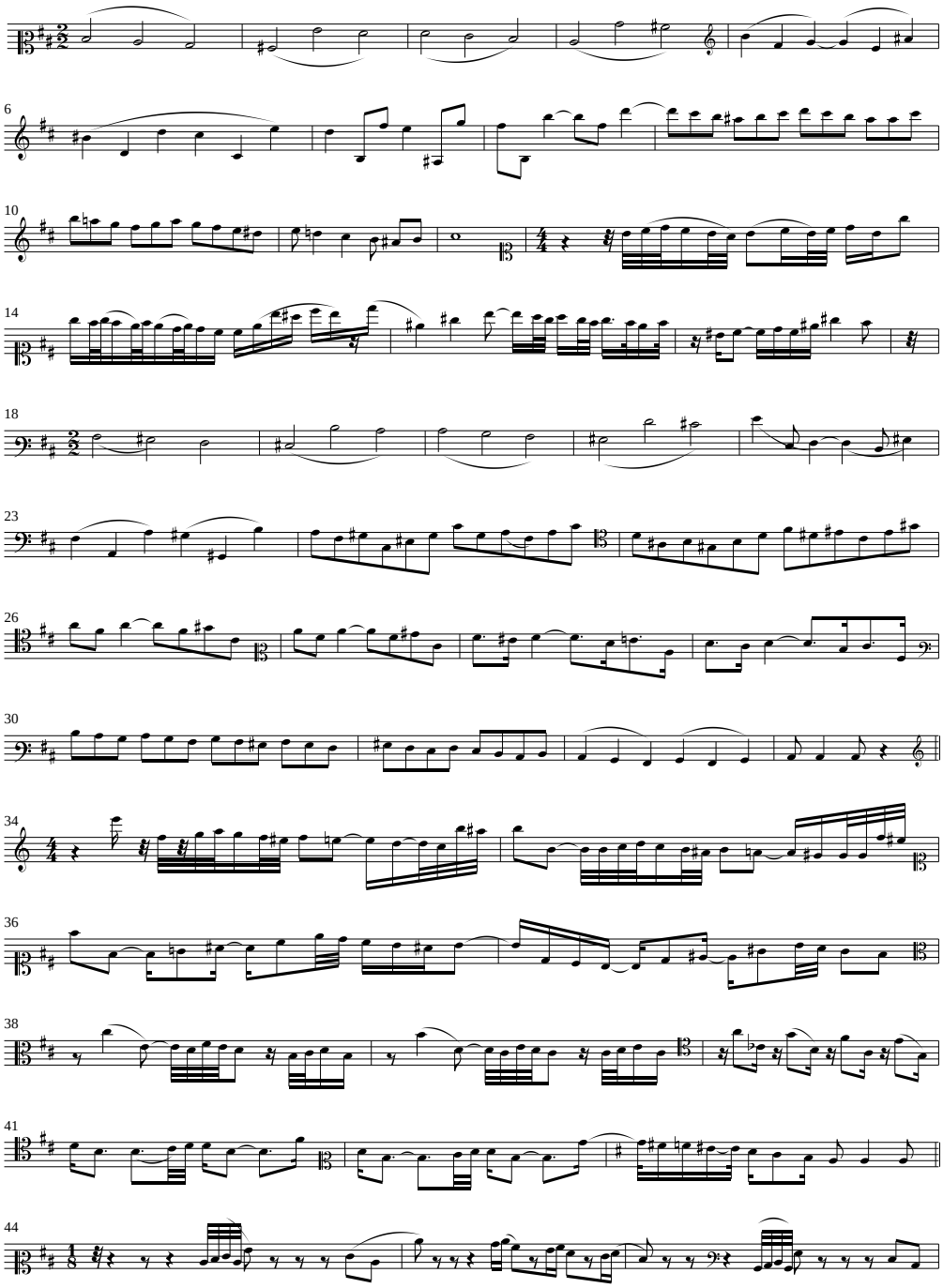


Figura 11.2: Esempio 1, immagine dello spartito ricostruito con O^{3}MR e visualizzato con l'editor musicale WEDELMUSIC

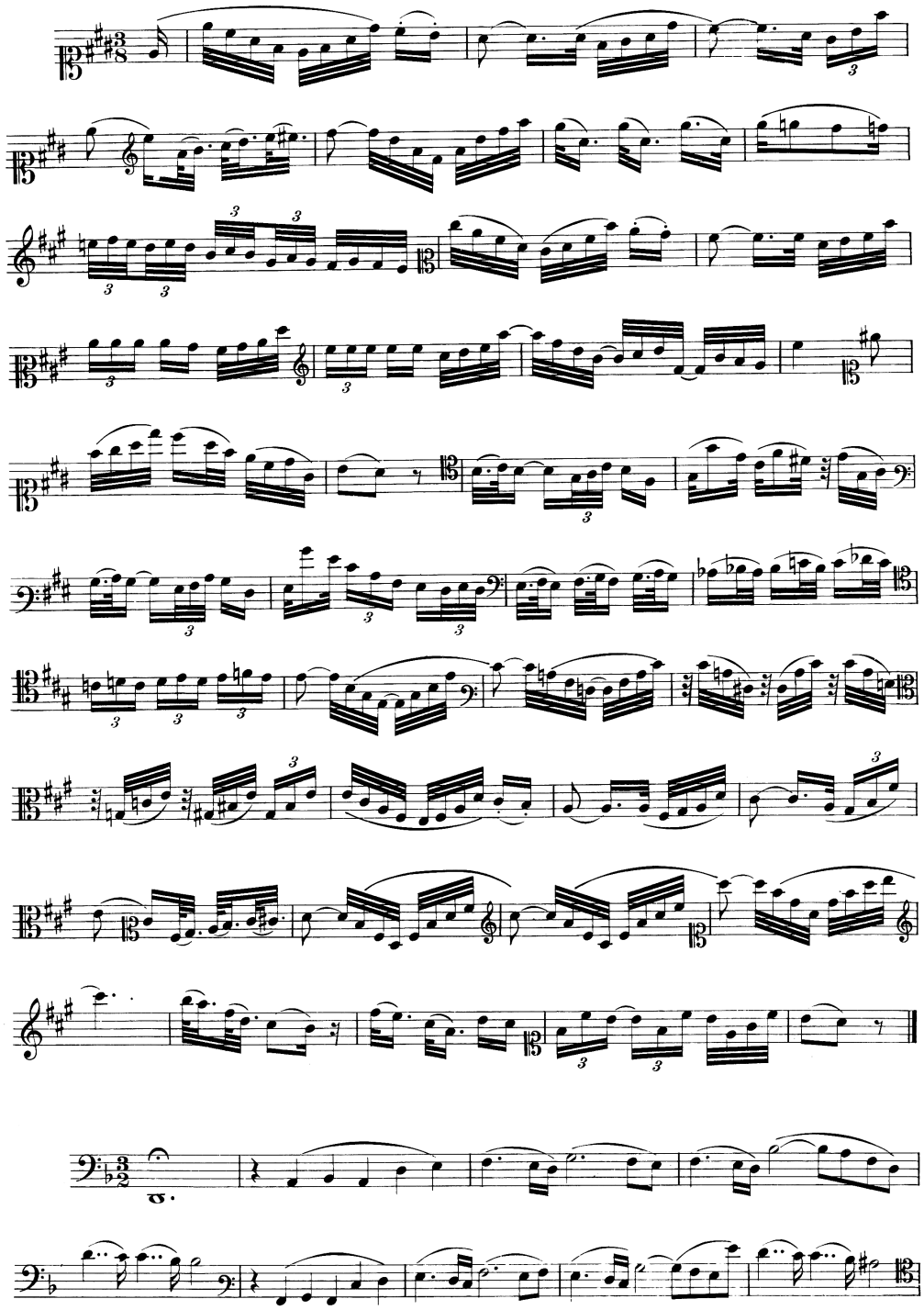


Figura 11.3: Esempio 2, immagine dello spartito originale

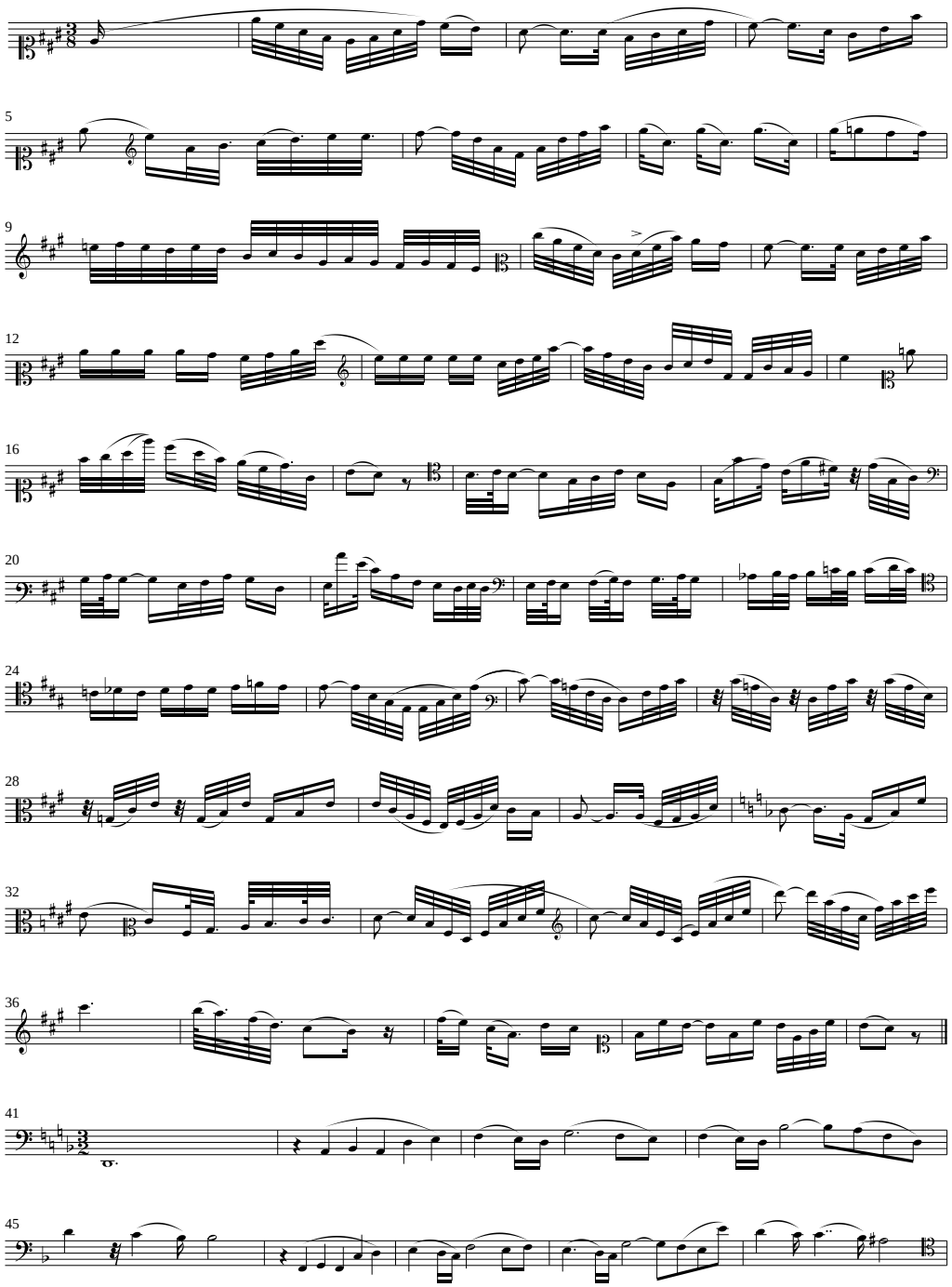


Figura 11.4: Esempio 2, immagine dello spartito ricostruito con O^{3}MR e visualizzato con l'editor musicale WEDELMUSIC

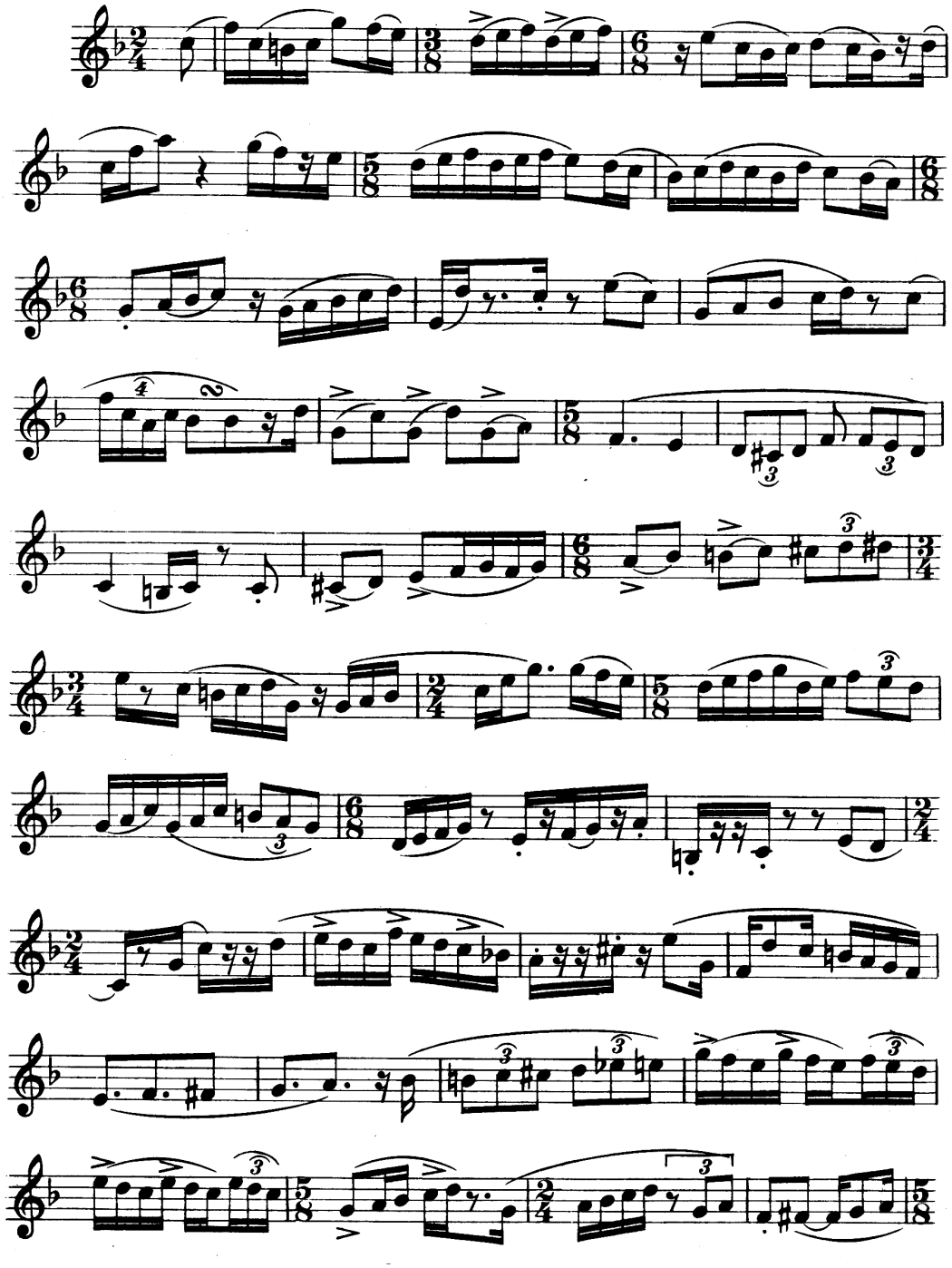


Figura 11.5: Esempio 3, immagine dello spartito originale

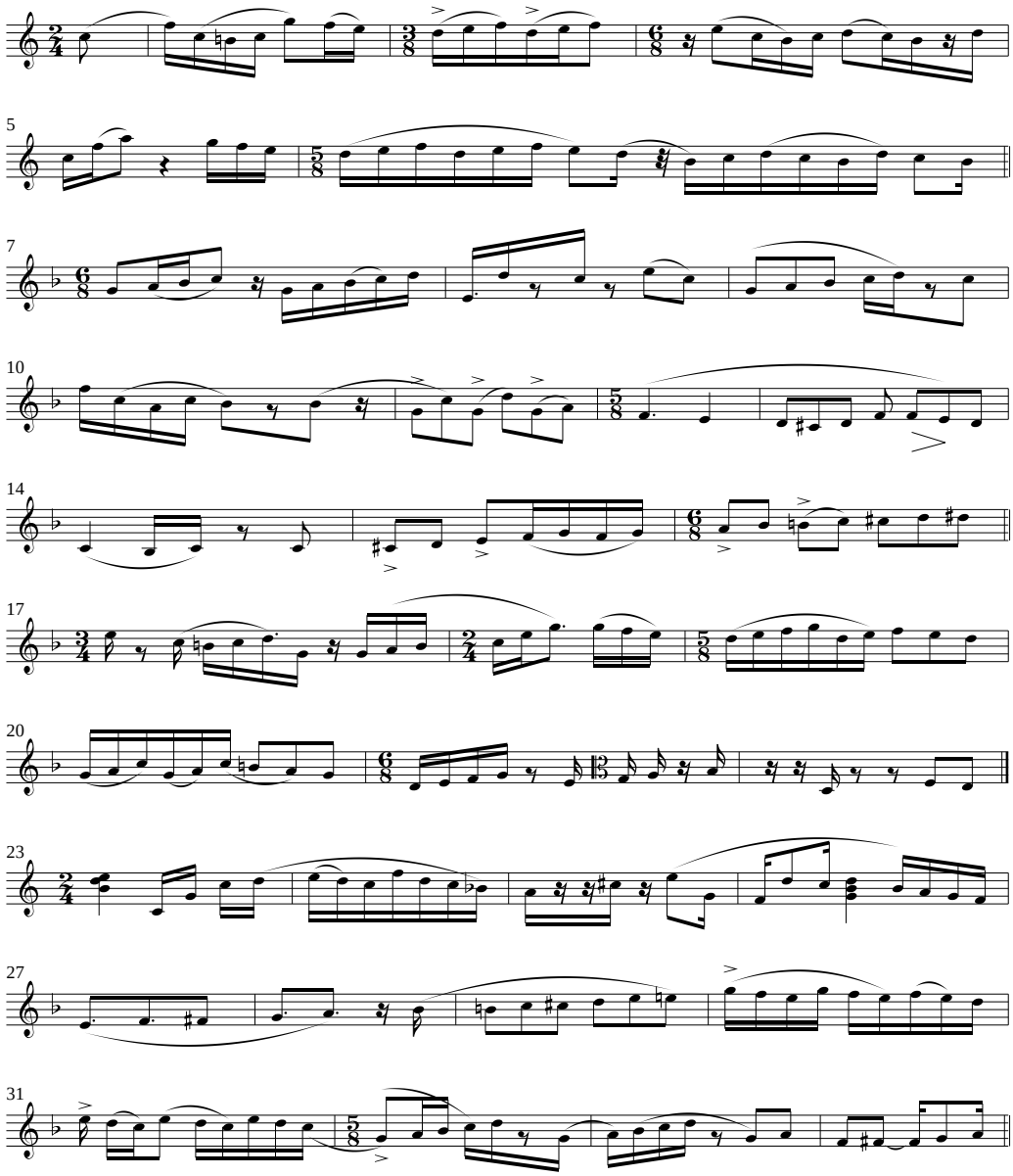


Figura 11.6: Esempio 3, immagine dello spartito ricostruito con O^{3}MR e visualizzato con l'editor musicale WEDELMUSIC

.

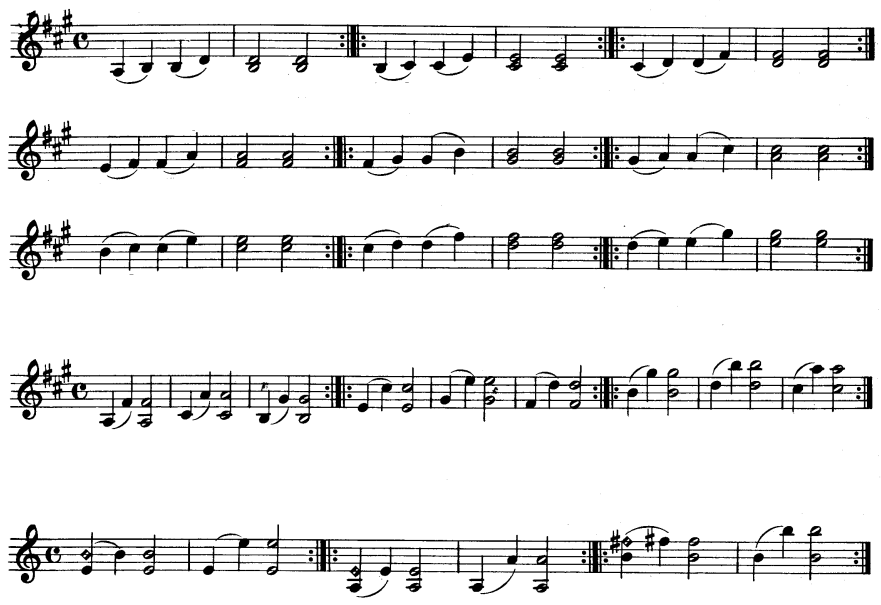


Figura 11.7: Esempio 4, immagine dello spartito originale

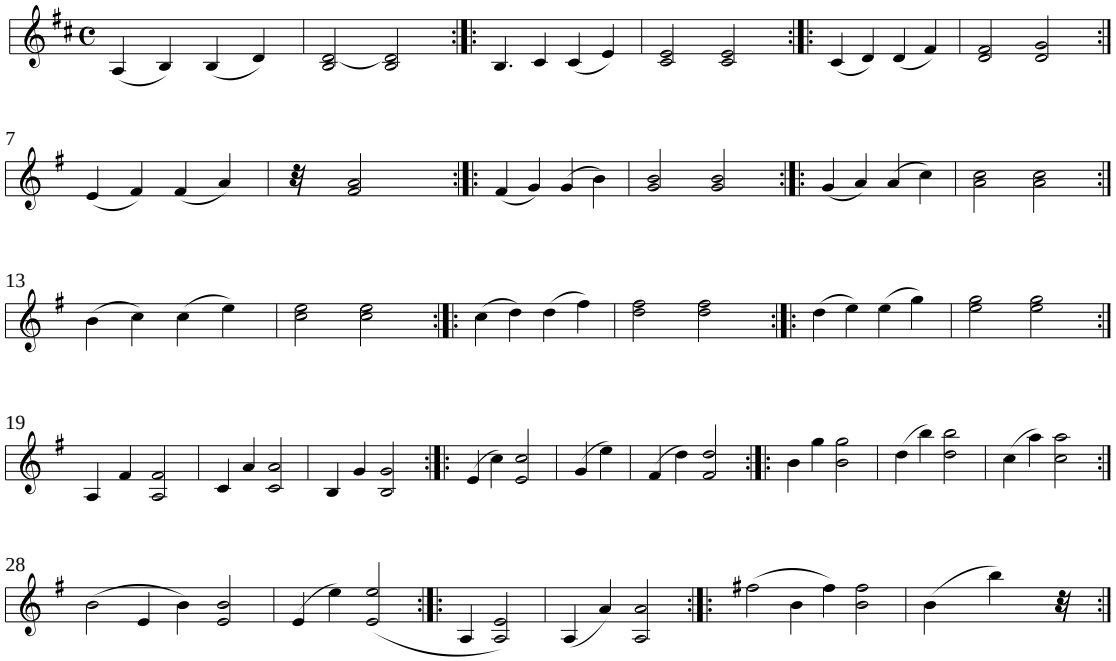


Figura 11.8: Esempio 4, immagine dello spartito ricostruito con O^{3}MR e visualizzato con l'editor musicale WEDELMUSIC

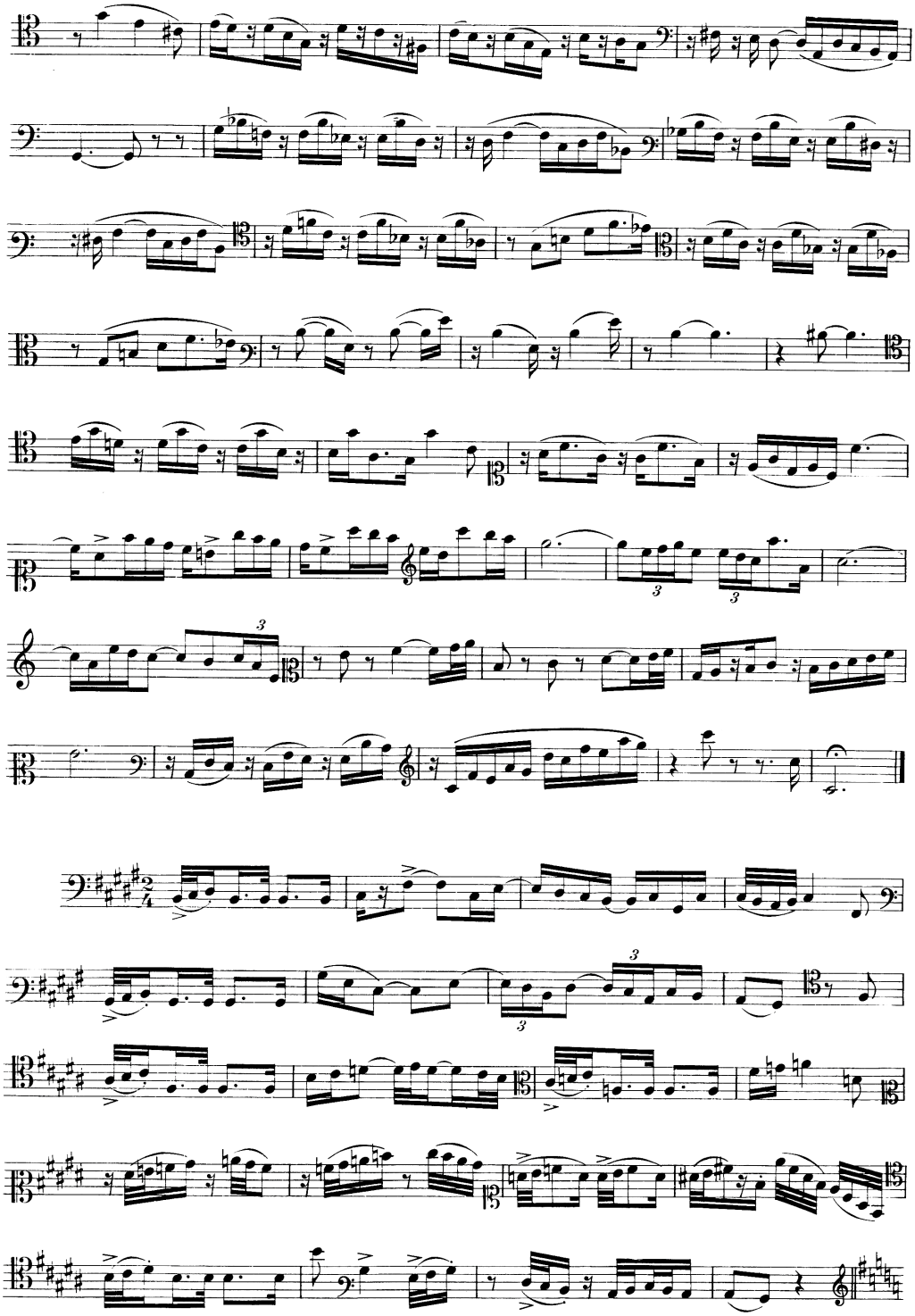


Figura 11.9: Esempio 5, immagine dello spartito originale

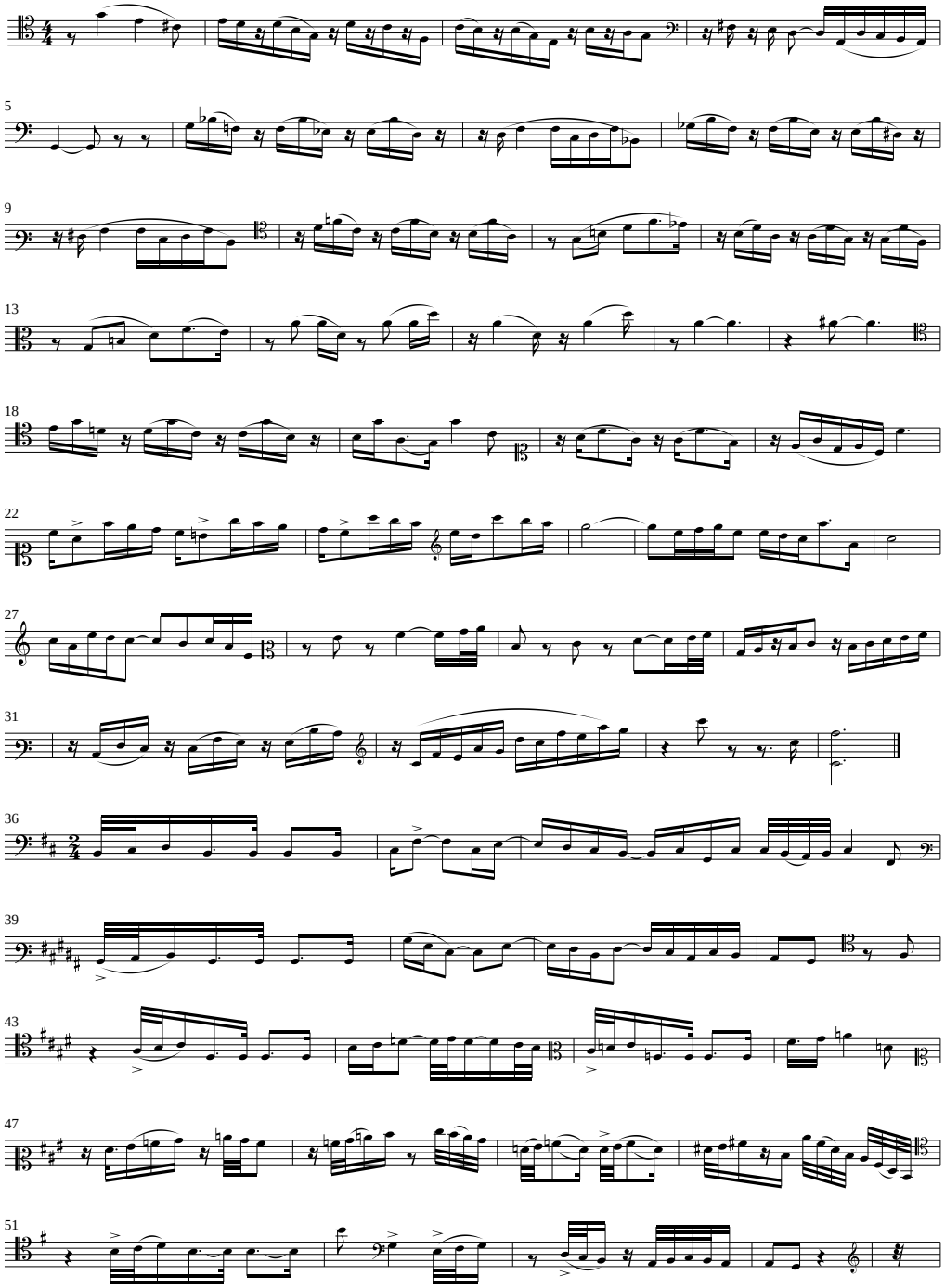


Figura 11.10: Esempio 5, immagine dello spartito ricostruito con O^{3}MR e visualizzato con l'editor musicale WEDELMUSIC

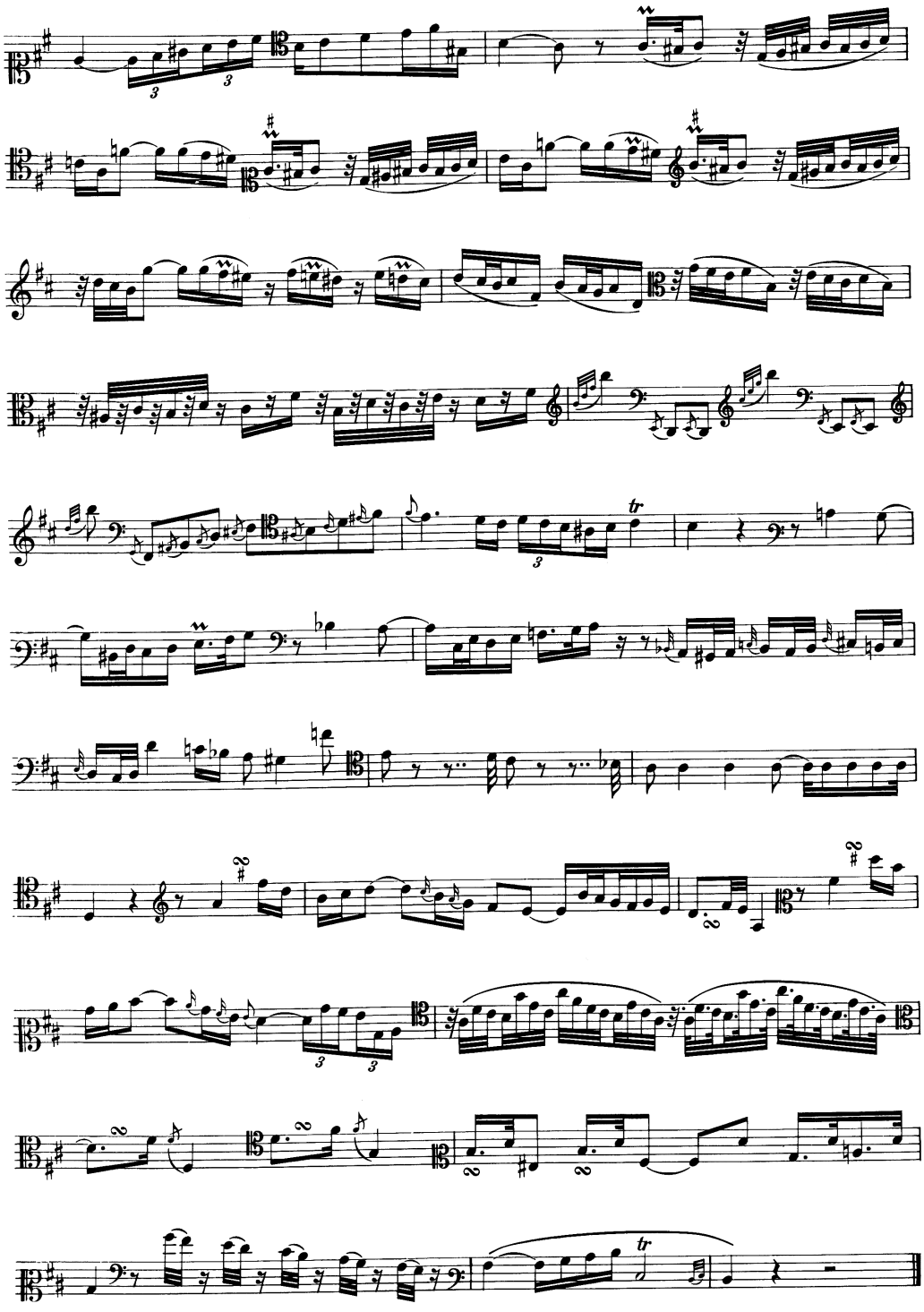


Figura 11.11: Esempio 6, immagine dello spartito originale

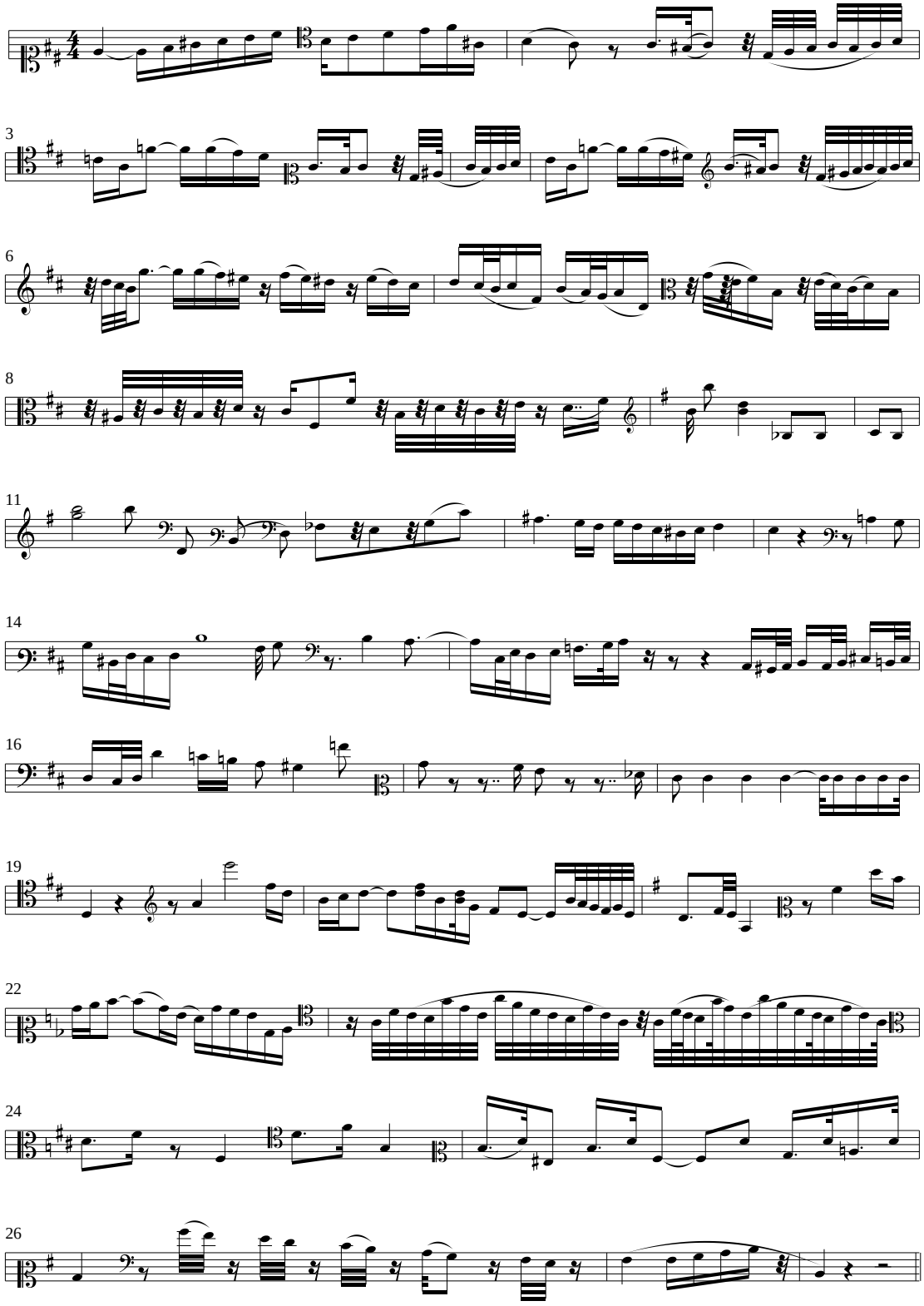


Figura 11.12: Esempio 6, immagine dello spartito ricostruito con O^{3}MR e visualizzato con l'editor musicale WEDELMUSIC

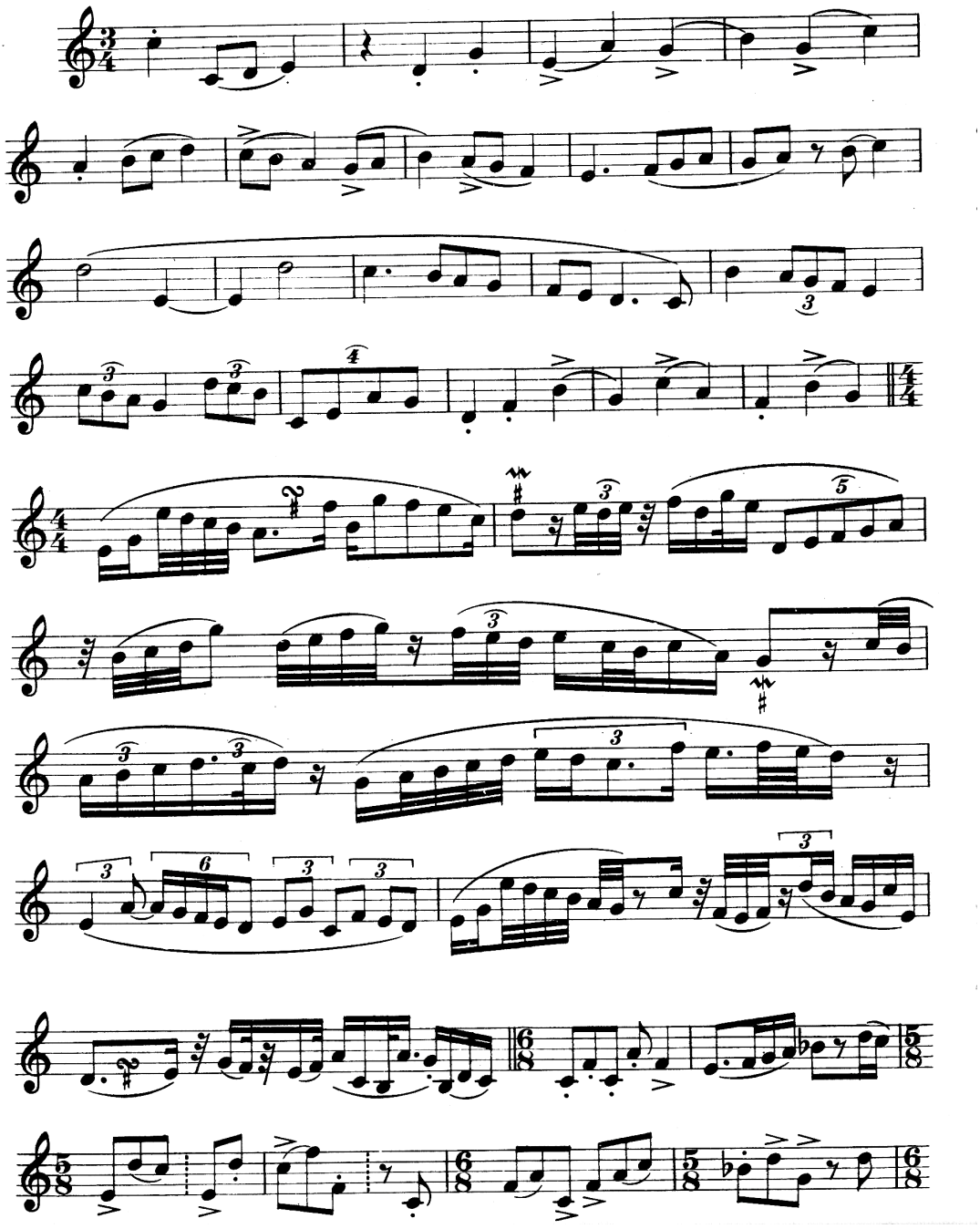


Figura 11.13: Esempio 7, immagine dello spartito originale

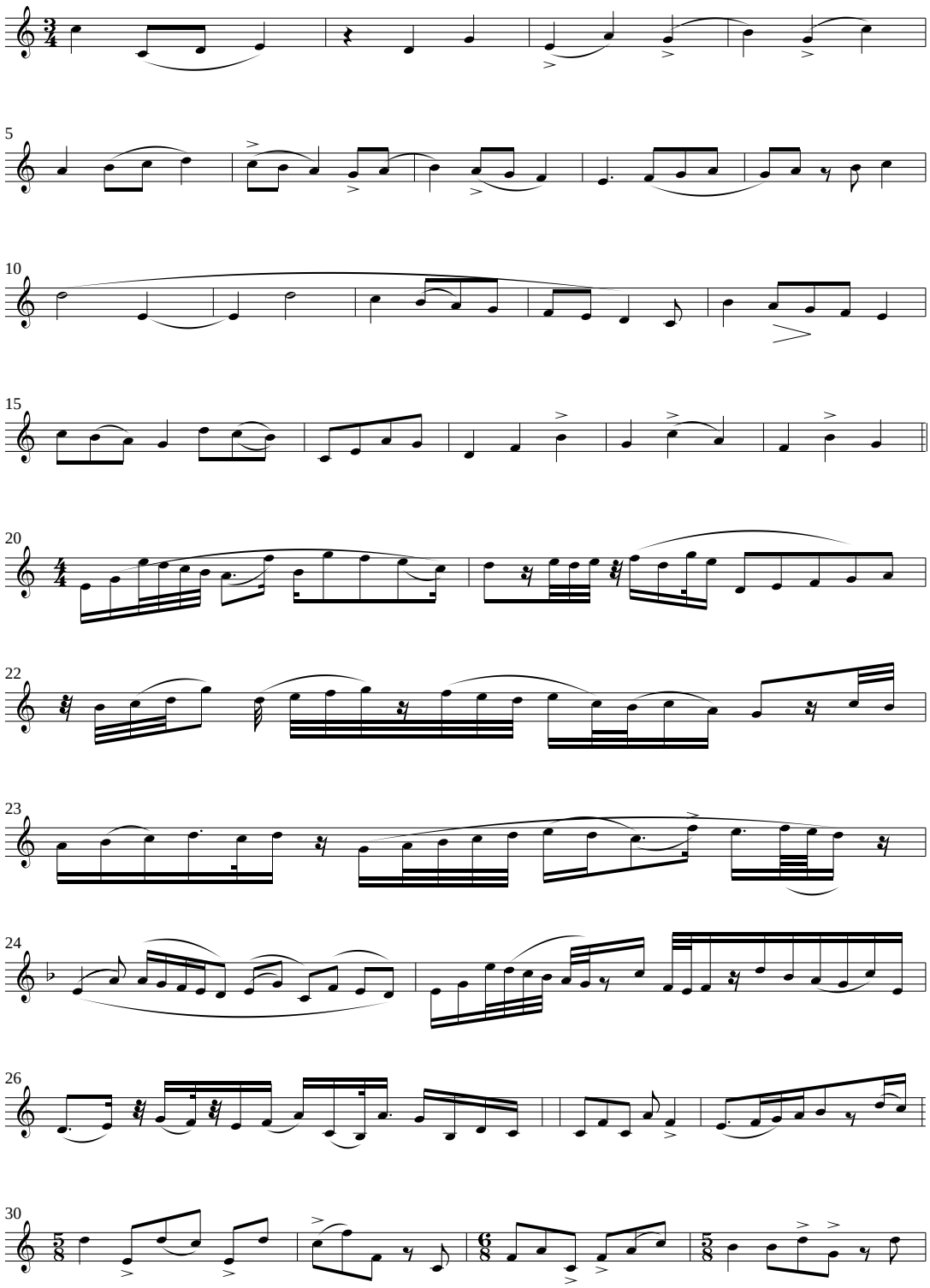


Figura 11.14: Esempio 7, immagine dello spartito ricostruito con O^{3}MR e visualizzato con l'editor musicale WEDELMUSIC

Figura 11.15: Valutazione simboli di base: SmartScore

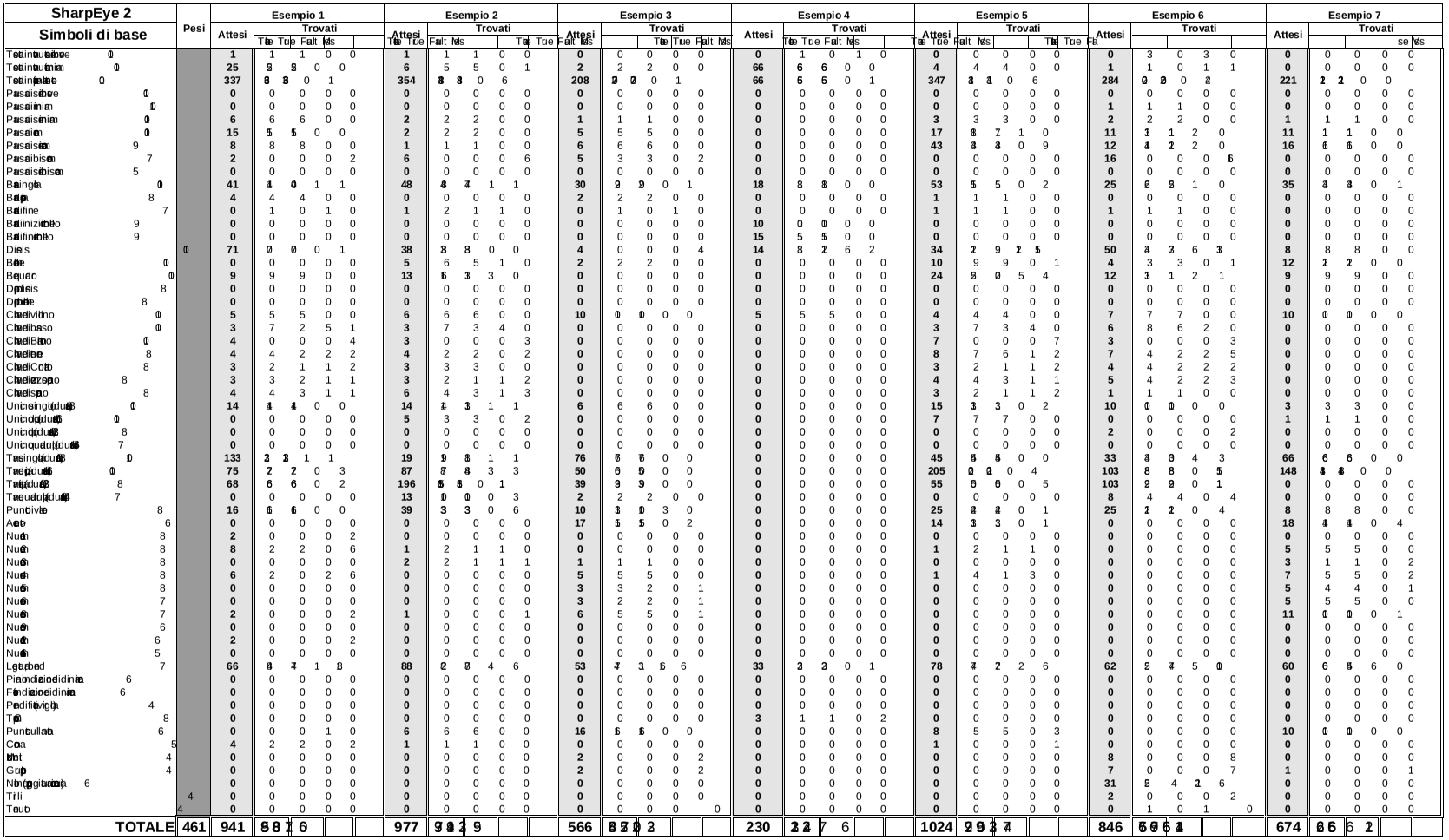


Figura 11.16: Valutazione simboli di base: SharpEye 2

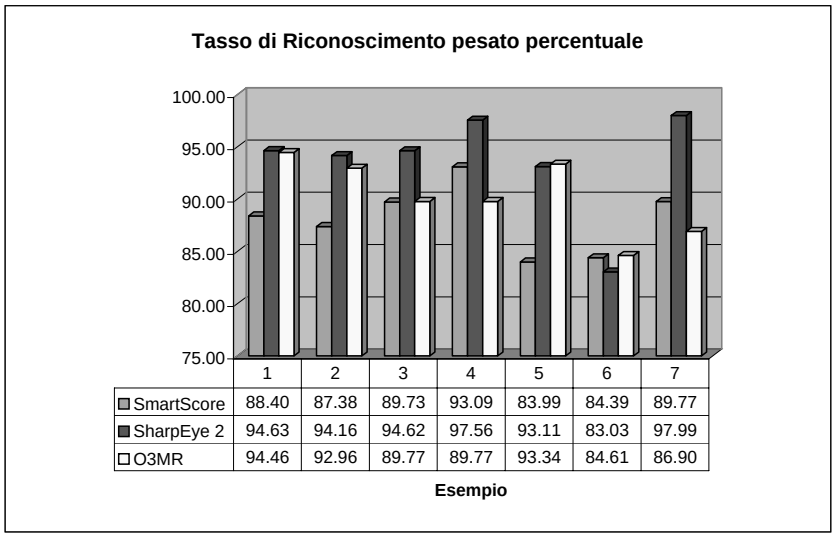
Figura 11.17: Valutazione simboli di base: O^{3}MR

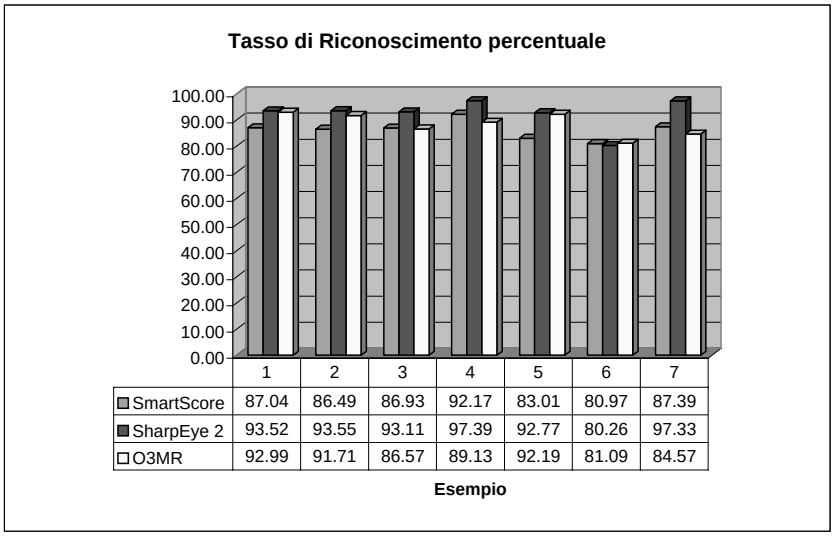
SmartScore	Pesi	Esempio 1					Esempio 2					Esempio 3					Esempio 4					Esempio 5					Esempio 6					Esempio 7				
Simboli musicali completi e relazioni		Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal						t	Ms	Atsi	TrueAdd	Fal		Ms
Nobu	10	363	3	1	7	6	361	3	7	5	9	210	0	4	0	3	132	2	2	0	4	351	3	8	3	0	285	2	2	9	5	221	2	3	0	2
Nobaio	10	54	0	0	2	2	24	2	0	0	3	2	2	0	0	0	1	0	0	0	1	38	0	1	2	6	34	1	8	0	3	18	0	0	0	0
Nuligu	8	82	0	0	0	0	99	0	0	0	2	48	0	0	0	1	0	0	0	0	0	85	0	0	0	0	68	0	0	1	0	64	0	1	0	2
Pase	10	31	0	1	0	0	11	2	2	0	9	17	0	3	0	7	0	0	0	0	0	63	6	1	0	0	42	0	6	0	0	28	0	4	0	0
Fa	6	8	1	0	0	7	2	0	0	0	2	9	4	0	1	4	3	3	1	0	0	1	0	0	0	1	0	0	0	0	0	18	5	0	5	8
Ar	6	13	0	0	0	0	12	2	0	0	0	10	9	0	1	0	5	0	0	5	0	13	3	0	0	0	11	0	0	0	0	10	0	0	0	0
Bi	6	4	0	4	0	0	7	6	3	0	1	37	0	3	0	0	0	0	0	0	0	23	7	2	0	0	17	0	1	0	0	29	0	3	0	0
No	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0
Le	7	66	3	0	0	0	88	0	0	0	0	53	0	5	7	0	33	3	0	0	0	78	0	4	0	0	62	0	6	0	0	60	0	0	3	0
Puntdiv	8	16	5	1	0	1	39	0	4	0	4	10	0	5	0	0	0	0	4	0	0	25	0	0	0	0	25	0	0	0	0	8	0	0	0	1
Cl	9	26	0	1	0	0	28	0	2	0	0	10	0	0	0	0	5	5	0	0	0	32	0	4	0	0	33	0	1	0	0	10	0	0	0	0
Gru	7	33	0	2	0	0	18	0	3	0	0	15	2	5	0	0	0	0	0	0	0	5	0	4	0	0	5	0	4	0	0	11	5	5	0	0
Nu	10	45	5	0	0	0	49	0	0	0	0	32	0	0	0	0	33	0	0	0	0	55	0	0	0	0	26	0	0	0	0	35	0	0	0	0
Nu	10	13	0				12	0				10	0				5	0				13	0				11	0				10	0			
Totale	112	754	0	0	0	0	750	0	0	0	0	463	0	0	0	0	217	0	0	0	0	782	0	0	0	0	650	0	0	0	0	522	0	0	0	0

SharpEye 2	Pesi	Esempio 1					Esempio 2					Esempio 3					Esempio 4					Esempio 5					Esempio 6					Esempio 7				
Simboli musicali completi e relazioni		Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal						t	Ms	Atsi	TrueAdd	Fal		Ms
Nobu	10	363	3	0	5	1	361	3	0	3	7	210	0	0	0	1	132	1	1	0	1	351	0	0	5	6	285	0	3	1	2	221	2	0	0	0
Nobaio	10	54	5	0	0	1	24	2	0	0	0	2	2	0	0	0	1	0	0	0	1	38	3	0	1	1	34	0	0	0	4	18	3	0	0	0
Nuligu	8	82	0	0	0	2	99	0	0	1	4	48	3	0	0	0	0	0	0	0	0	85	0	0	0	1	68	0	0	0	7	64	0	0	0	0
Pase	10	31	0	0	0	2	11	5	0	0	6	17	5	0	0	2	0	0	0	0	0	63	0	0	1	8	42	0	0	4	2	28	0	0	0	0
Fa	6	8	0	0	1	7	2	1	0	1	0	9	8	0	0	1	3	1	0	0	2	1	1	1	0	0	0	0	1	0	0	18	5	0	0	3
Ar	6	13	1	0	0	0	12	2	1	0	0	10	0	0	0	0	5	4	3	1	0	13	9	0	4	0	11	9	4	2	0	10	0	0	0	0
Bi	6	4	2	1	0	2	7	7	0	0	0	37	1	0	0	6	0	0	0	0	0	23	1	0	0	5	17	0	0	0	1	29	2	0	0	5
No	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	4	0	2	6	0	0	0	0	0
Le	7	66	7	0	1	1	88	3	0	6	4	53	1	0	6	6	33	2	0	0	1	78	2	0	2	4	62	7	0	5	0	60	5	0	6	0
Puntdiv	8	16	6	1	0	0	39	3	0	0	6	10	0	3	0	0	0	0	0	0	0	25	2	0	0	1	25	2	0	0	4	8	8	0	0	0
Cl	9	26	5	5	5	6	28	3	0	6	4	10	0	0	0	0	5	5	0	0	0	32	1	0	8	6	33	2	0	8	5	10	0	0	0	0
Gru	7	33	5	0	8	0	18	1	0	0	5	15	3	0	0	2	0	0	0	0	0	5	3	0	0	2	5	0	1	0	5	11	5	0	0	6
Nu	10	45	5	1	0	0	49	0	1	0	0	32	2	0	0	0	33	3	1	0	0	55	5	0	0	2	26	2	1	0	0	35	3	0	0	0
Nu	10	13	1				12	2				10	0				5	5				13	1				11	1				10	0			
Totale	112	754	6	3	0	9	750	6	2	7	6	463	8	3	6	8	217	2	5	1	5	782	7	1	2	3	650	5	0	4	0	522	9	0	6	4

O3MR	Pesi	Esempio 1					Esempio 2					Esempio 3					Esempio 4					Esempio 5					Esempio 6					Esempio 7				
Simboli musicali completi e relazioni		Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal		Ms	Atsi	TrueAdd	Fal						t	Ms	Atsi	TrueAdd	Fal		Ms
Nobu	10	363	3	0	5	1	361	3	0	9	1	210	0	2	0	0	132	2	2	1	7	351	0	1	1	1	285	2	4	6		221	1	6	3	4
Nobaio	10	54	0	1	1	3	24	1	0	2	1	2	0	0	0	2	1	0	0	0	1	38	2	0	0	0	34	5	0	1	8	18	3	0	0	5
Nuligu	8	82	0	0	1	1	99	0	0	0	0	48	4	0	2	2	0	0	0	0	0	85	0	0	0	0	68	0	0	1	3	64	0	0	0	4
Pase	10	31	1	3	0	0	11	1	1	0	0	17	0	0	0	1	0	0	2	0	0	63	0	3	0	1	42	0	6	1	2	28	2	2	0	5
Fa	6	8	4	0	1	3	2	2	0	0	0	9	5	0	0	4	3	1	0	0	2	1	1	0	1	0	0	0	1	0	0	18	1	0	0	5
Ar	6	13	2	1	1	0	12	2	1	0	0	10	9	0	1	0	5	0	0	5	0	13	9	0	4	0	11	8	0	3	0	10	7	0	3	0
Bi	6	4	0	0	0	4	7	0	0	0	7	37	1	1	0	0	0	0	0	0	0	23	2	0	0	1	17	0	0	0	1	29	2	0	0	1
No	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0
Le	7	66	0	4	2	0	88	5	3	2	1	53	6	7	2	4	33	2	1	4	4	78	4	6	1	2	62	1	8	6	3	60	2	5	6	2
Puntdiv	8	16	1	0	0	3	39	2	1	0	1	10	8	3	0	2	0	0	1	0	0	25	0	2	0	5	25	1	5	0	9	8	6	2	0	2
Cl	9	26	2	0	1	2	28	2	0	1	3	10	0	0	0	0	5	5	0	0	0	32	1	0	8	6	33	2	0	3	4	10	0	0	0	0
Gru	7	33	0	0	0	3	18	0	0	0	1	15	0	0	0	5	0	0	0	0	0	5	0	0	0	5	5	0	0	0	5	11	0	0	0	1
Nu	10	45	5	1	0	0	49	0	0	0	0	32	2	1	0	0	33	3	0	0	0	55	1	1	0	1	26	2	2	0	0	35	3	0	0	1
Nu	10	13	1				12	2				10	0				5	5				13	1				11	1				10	0			
Totale	112	754	6	1	0	0	750	5	6	0	6	463	7	1	0	0	217	2	1	6	0	782	0	1	1	2	650	4	3	9	5	522	4	5	2	5

Figura 11.18: Valutazione simboli musicali completi e relazioni





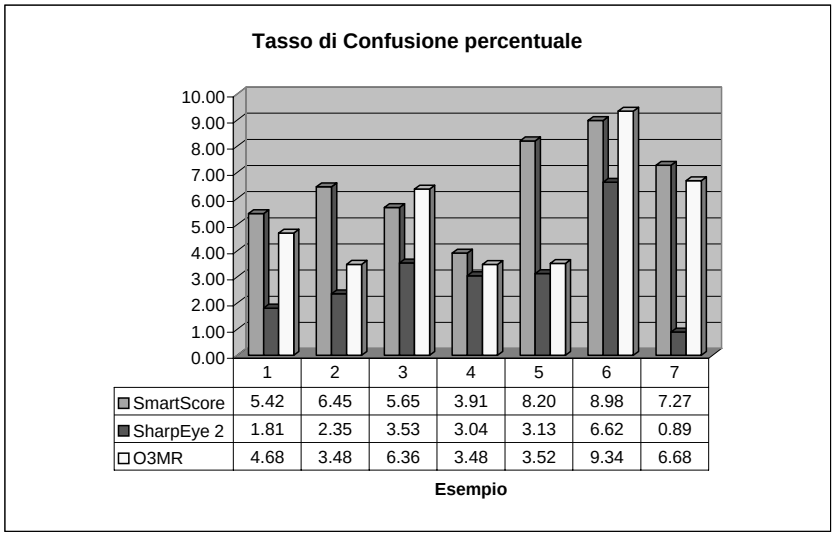
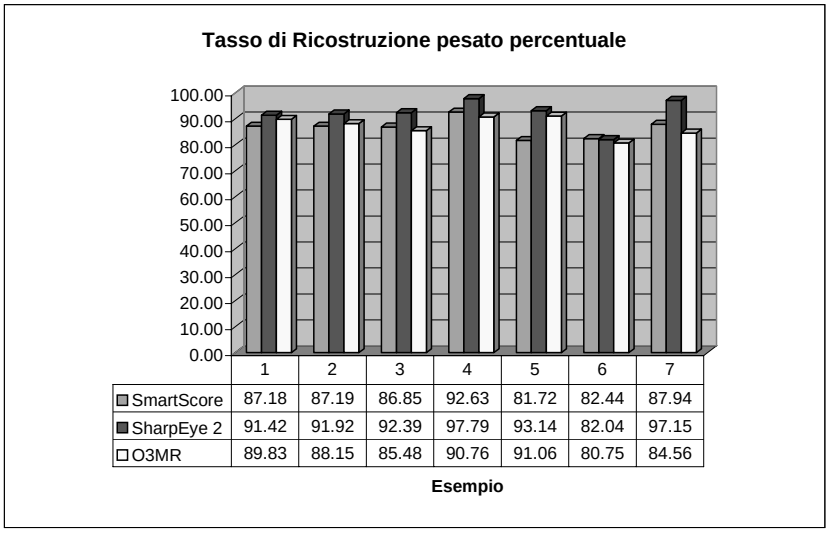
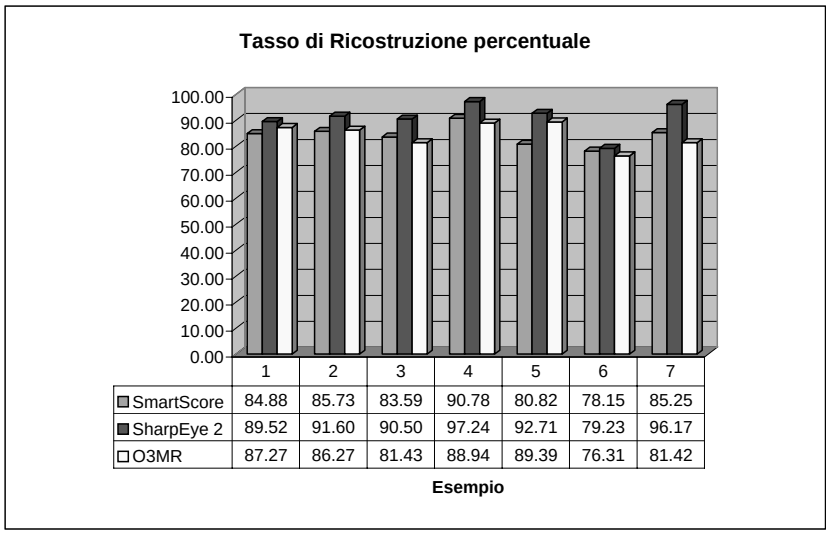


Figura 11.19: Valutazione simboli di base: indici di prestazione





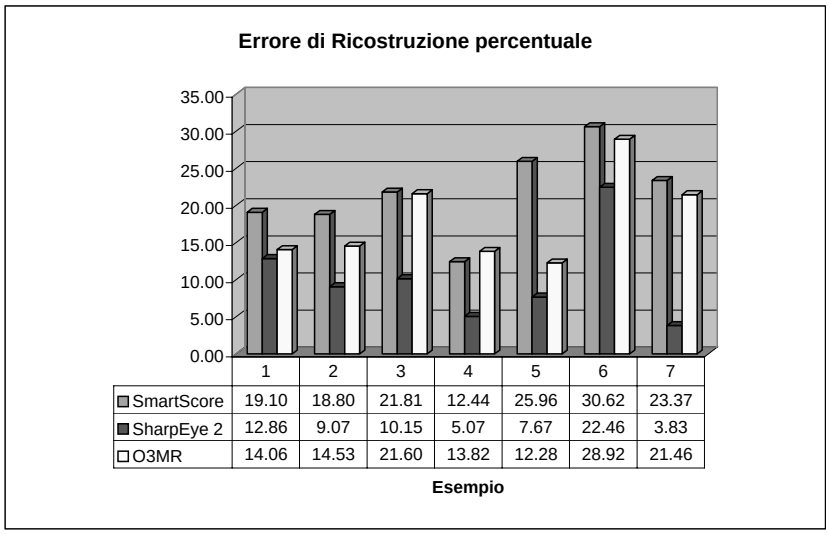


Figura 11.20: Valutazione simboli musicali e relazioni: indici di prestazione

Capitolo 12

Conclusioni

In questa tesi, è stato affrontato il problema dell'analisi e dello sviluppo di metodi e strumenti per il riconoscimento e l'indicizzazione automatica di spartiti musicali e la valutazione delle prestazioni di un sistema di riconoscimento ottico di caratteri musicali. Di seguito si riporta un'analisi conclusiva sul lavoro svolto e sui risultati ottenuti e gli sviluppi futuri.

12.1 Il riconoscimento automatico di spartiti musicali

Il sistema è stato scomposto in tre moduli principali seguendo l'architettura classica: segmentazione, riconoscimento e ricostruzione. Seguendo tale struttura, si riportano i risultati conclusivi.

Segmentazione

Il processo di segmentazione è stato progettato per individuare informazioni topologiche e proprietà grafiche direttamente dall'immagine dello spartito in modo da realizzare un'operazione di setup automatico rispetto allo spartito da riconoscere e costruire una conoscenza come guida del processo di frammentazione. Il processo di decomposizione dell'immagine in elementi grafici segue un criterio di localizzazione progressiva: identificazione dei pentagrammi (livello 0), estrazione di segmenti verticali di immagine contenenti i simboli musicali (livello 1), frammentazione dei segmenti in primitive grafiche (livello 2). Durante le fasi di decomposizione, si estraggono informazioni sul contesto di carattere strutturale (identificazione dei gruppi di note connesse con travi e fine pentagramma) e informativo (presenza o meno di una testa di nota nera). Analizzando singolarmente i risultati ottenuti dalla fase di segmentazione si traggono le seguenti conclusioni.

Identificazione dei pentagrammi – Il metodo sviluppato per la ricerca dei pentagrammi all'interno di uno spartito musicale ha fornito risultati soddisfacenti e ha confermato la robustezza nei confronti dell'inclinazione e della variabilità di spessore delle linee del pentagramma.

Estrazione di segmenti verticali – È la fase che ha presentato le difficoltà maggiori dovute principalmente alla variabilità e alla complessità della notazione e della scrittura musicale. Anche in questo caso, è stato definito un procedimento di localizzazione progressiva, che individua una prima scomposizione in gruppi di note musicali e figure musicali isolate ed indipendenti, e riduce il problema all'analisi di sotto parti dell'immagine contente il pentagramma. Sono state sviluppate delle elaborazioni che operando direttamente sui segmenti di immagine permettono di testare la presenza della testa di nota piena e di estrarre i segmenti che la contengono. Successivamente, per differenza sono estratti i segmenti contenenti gli altri simboli musicali. I risultati della scomposizione in gruppi sono da ritenersi soddisfacenti e consentono di ricostruire le strutture e le relazione tra i gruppi di note. L'identificazione della testa di nota si è dimostrata efficiente anche nelle configurazioni con un'alta densità di simboli musicali. Queste considerazioni sono confermate dalle valutazioni condotte sui simboli di base relative alle teste di nota. Il metodo sviluppato per l'individuazione degli altri elementi grafici, sebbene sia in grado di risolvere alcune scomposizioni, si ipotizza sia responsabile di frammentazioni indesiderate che nella decomposizione finale sono responsabili della generazione di primitive grafiche che non fanno parte nell'insieme dei simboli di base scelto.

Estrazione delle primitive – Questa fase del processo di segmentazione ha richiesto la definizione di più metodi di scomposizione specifici poiché in sede di analisi è stata constatata la difficoltà di generalizzazione dovuta alla variabilità grafica dei simboli musicali. Sulla base delle informazioni individuate nelle fasi precedenti è stato possibile definire metodi di decomposizione per segmenti contenenti la testa di nota nera, segmenti contenenti la nota da 1/4 e un metodo più generale per tutti gli altri simboli. Le primitive grafiche sono estratte in modo uniforme e hanno consentito la definizione di un insieme di simboli di base. Tuttavia, in corrispondenza dei segmenti contenenti la testa nera è stato riscontrata la perdita dei simboli di staccato e di alcune partenze ed arrivi delle legature. Questa perdita di informazione è imputata alla procedura di segmentazione finale.

Riconoscimento

È stato osservato che il classificatore neurale utilizzato nella fase di riconoscimento può essere addestrato direttamente con le immagini dei simboli di base del database di addestramento senza l'utilizzo di *feature* aggiuntive. La rete neurale ha confermato la capacità

di generalizzare, in quanto è in grado di classificare immagini provenienti da spartiti diversi nelle dimensioni dei e tipo di font. Nonostante questo, il classificatore introduce un certo grado di imprecisione legata alla confusione nel riconoscere simboli apparentemente simili, è necessario intervenire per ridurre l'incertezza sul risultato in uscita. Per questo scopo è stato affiancato un meccanismo di riclassificazione di quei simboli, che sulla base delle informazioni di contesto e topologiche individuate nella fase di segmentazione e sul basso livello di confidenza associato dalla rete neurale, risultano degli evidenti errori di classificazione. La decisione sulla bontà della riclassificazione di questi simboli è rimandata alla fase di ricostruzione che li valuta all'interno del contesto informativo che essi devono rappresentare. L'utilizzo di una rete neurale, infine, consente di estendere le capacità di riconoscimento nel momento in cui si realizzi una nuova struttura addestrata su un insieme di simboli più esteso e con prestazioni migliori.

Ricostruzione

Il modello di ricostruttore implementato è in grado di recuperare gli errori introdotti dalle precedenti fasi di segmentazione e di produrre un risultato finale il più verosimile possibile al documento da ricostruire. Per il raggiungimento dei risultati ù stato molto importante la fase di studio iniziale dalla quale sono state tratte tutte le strategie che hanno portato alla creazione del modello finale del ricostruttore. L'analisi critica dei risultati ottenuti dalla fase di riconoscimento ha evidenziato un livello di incertezza nei risultati classificazione. Lo studio del contesto musicale ha fornito le basi sulle quali eliminare tale indeterminatezza. Tutto questo è stato tradotto in un meccanismo di decisione che accetta o meno le conclusioni tratte dal modulo di riconoscimento e, nel caso tali conclusioni non siano ritenute corrette, di intervenire per modificarle. Allo stesso tempo l'attenzione è stata rivolta al problema dell'estensibilità. Questa caratteristica è riscontrabile in tutti i moduli del sistema: sia la Tabella delle Relazioni, sia il modello probabilistico definito con la Grammatica posizionale e sia l'insieme di regole della Grammatica musicale possono essere modificati ed estesi senza provocare inefficienze da parte del ricostruttore stesso.

12.2 Valutazione delle prestazioni

Limitatamente alle immagini campione di musica monofonica utilizzate per i test, i risultati ottenuti dimostrano che: il sistema sviluppato, allo stato attuale, ha un comportamento medio non molto dissimile da SmartScore e rispetto ai due software è limitato nella gestione dei simboli di staccato, mordente, gruppetti, trillo, nel riconoscimento delle notine e dei gruppi irregolari, e presenta alcuni problemi nella ricostruzione delle legature. Relativamente ai due software utilizzati per la comparazione, i dati confermano le capacità di SharpEye 2, ritenuto da gran parte degli utenti il miglior software in commercio, e

una tendenza da parte di SmartScore ad introdurre simboli non corretti. L'analisi sul riconoscimento dei simboli più importanti come le note e le pause, ha mostrato che il sistema O^{3}MR presenta un'elevata capacità nella ricostruzione di tali simboli superiore ai due software utilizzati per la comparazione. In relazione ai criteri di valutazione utilizzati si nota una differenza nei valori degli indici di valutazione ed in particolare è stata confermata, come previsto, la diminuizione dei valori nel caso della valutazione basata sui simboli completi e sulle relazioni. In riferimento al successivo lavoro di correzione, il secondo criterio è forse quello che meglio descrive gli aspetti legati al comportamento dei sistemi di riconoscimento, coinvolgendo direttamente gli aspetti legati alla ricostruzione.

12.3 Considerazioni finali e sviluppi futuri

I metodi di valutazione utilizzati per misurare le prestazioni del sistema hanno mostrato un alto livello di ricostruzione degli spartiti monofonici campione anche alla luce delle comparazioni con i software presi a confronto. Il problema maggiore tuttavia è la mancanza di un criterio, di immagini e di metriche universalmente riconosciuti per valutare in modo oggettivo le prestazioni. In questo senso una delle attività future è migliorare i modelli proposti per la valutazione delle prestazione da sottoporre successivamente all'attenzione della comunità scientifica, cooperare nella definizione di un database di immagini di spartiti campione e di una terminologia standard. In questa ottica, è necessario non limitarsi al caso monofonico, ma considerare la musica più in generale possibile. Per questa ragione, uno degli sviluppi futuri per quanto concerne il sistema O^{3}MR sarà l'estensione della gestione dei simboli musicali attualmente non considerati e il riconoscimento di spartiti polifonici (in cui sono presenti accordi e gestione multi voce).

Infine, la definizione di un modello di valutazione delle prestazioni e la stima dei costi, legati alla scrittura di uno spartito con un editor di video scrittura musicale, consentirà di valutare il limite di convenienza nell'utilizzo di una procedura automatica di riconoscimento e le caratteristiche in termini di prestazioni che questa deve avere. A questo scopo, un'indagine statistica condotta su un campione significato di utenti di applicativi per la video scrittura e per il riconoscimento automatico di spartiti musicali potrà consentire la stima di tali costi, definire delle soluzioni per ridurre il tempo di valutazione del risultato di ricostruzione e stabilire se sia necessario considerare tutti i simboli musicali e le relazioni allo stesso livello oppure indirizzare le risorse nel riconoscimento dei simboli più importanti e più frequenti.

Appendice A

Archivio delle regole

```
// SIMBOLI BARLINE

VRULE
  ::S_SBARLINE(altezzaGreat>3.9)
  ==> BARLINE(isSingle);

VRULE
  S_DIGIT4(larghezzaLess<0.8)
  ::S_SBARLINE(altezzaGreat>2)
  ==> BARLINE(isSingle);

VRULE
  ::S_BEAM4(altezzaGreat>2,withoutNote,notBeamed)
  S_DIGIT1
  ==> BARLINE(isSingle);

VRULE
  ::S_DBARLINE(altezzaGreat>4)
  ==> BARLINE(isDouble);

VRULE
  ::S_CHORD3V(altezzaGreat>4)
  ==> BARLINE(isEnd);

VRULE
  ::S_BEAM4(altezzaGreat>4,notBeamed)
  ==> BARLINE(isEnd);

VRULE
  ::S_DIGIT1(altezzaGreat>4)
  ==>BARLINE(isEnd);

// CHIAVI (Le linee del pentagramma sono in ordine crescente a partire dall'alto)

VRULE
  ::S_TENOR(pos=OnStaffLine2)
  [S_STAFLINE]
  ==> CLEF(isTenore);
```

```
VRULE
  ::S_CC(pos=OnStaffLine3)
  ==> CLEF(isContralto);

VRULE
  [S_STAFLINE]
  ::S_CCLOW(pos=OnStaffLine4)
  ==> CLEF(isMezzosoprano);

VRULE
  [S_STAFLINE]
  ::S_CCLLOW(pos=OnStaffLine5)
  ==> CLEF(isSoprano);

VRULE
  ::S_BASS(pos=UprHalfStaff,altezzaGreat>2.3)
  [S_STAFLINE]
  ==> CLEF(isBasso);

VRULE
  [S_STAFLINE]
  ::S_BASS(pos=LwrHalfStaff,altezzaGreat>2.3)
  ==> CLEF(isBaritono);

VRULE
  ::S_TREBLE(altezzaGreat>3)
  ==> CLEF(isTreble);

(TDP che diventa chiave di FA)
VRULE
  ::S_TDP(pos=UprHalfStaff,altezzaGreat>1.5,insideStaff)
  [S_STAFLINE]
  ==> CLEF(isBasso);

VRULE
  [S_STAFLINE]
  ::S_TDP(pos=LwrHalfStaff,altezzaGreat>1.5,insideStaff)
  ==> CLEF(isBaritono);

(TDF che diventa chiave di DO)
VRULE
  ::S_TDF(pos=OnStaffLine4, altezzaGreat>3)
  ==> CLEF(isMezzosoprano);

VRULE
  ::S_TDF(pos=OnStaffLine5, altezzaGreat>3)
  ==> CLEF(isSoprano);

VRULE
  ::S_TDF(pos=OnStaffLine3, altezzaGreat>3)
  ==> CLEF(isContralto);
```

```
VRULE
  ::S_TDF(pos=OnStaffLine2, altezzaGreat>3)
  ==> CLEF(isTenore);

(S_BEAM3 che diventa chiave di DO)
VRULE
  ::S_BEAM3(pos=OnStaffLine4, altezzaGreat>3,larghezzaGreat>2)
  ==> CLEF(isMezzosoprano);

VRULE
  ::S_BEAM3(pos=OnStaffLine5, altezzaGreat>3,larghezzaGreat>2)
  ==> CLEF(isSoprano);

VRULE
  ::S_BEAM3(pos=OnStaffLine3, altezzaGreat>3,larghezzaGreat>2)
  ==> CLEF(isContralto);

VRULE
  ::S_BEAM3(pos=OnStaffLine2, altezzaGreat>3,larghezzaGreat>2)
  ==> CLEF(isTenore);

// INDICAZIONI TEMPORALI
VRULE
  S_STAFLINE
  ::S_C
  S_STAFLINE
  ==>TIME(durate:=c);

VRULE
  S_STAFLINE
  ::S_CSLASH
  S_STAFLINE
  ==>TIME(durate:=C);

VRULE
  ::S_DIGIT2(altezzaGreat>1.5)
  S_DIGIT2(altezzaGreat>1.5)
  ==>TIME(durate:=2/2);

VRULE
  S_DIGIT3(altezzaGreat>1.5)
  ::S_DIGIT2(altezzaGreat>1.5)
  ==>TIME(durate:=3/2);

VRULE
  S_DIGIT4(altezzaGreat>1.5)
  ::S_DIGIT2(altezzaGreat>1.5)
  ==>TIME(durate:=4/2);

VRULE
  ::S_DIGIT2(altezzaGreat>1.5)
  S_WHOLENOTE
  S_HALFREST8
  ==>TIME(durate:=2/2);
```

```
VRULE
  S_DIGIT2(altezzaGreat>1.5)
  ::NOTE(durate=1)
  S_HALFREST8
  ==>TIME(durate:=2/2);

VRULE
  ::S_DIGIT2(altezzaGreat>1.5)
  S_DIGIT4(altezzaGreat>1.5)
  ==>TIME(durate:=2/4);

VRULE
  S_WHOLENOTE
  S_HALFREST8
  ::S_DIGIT4(altezzaGreat>1.5)
  ==>TIME(durate:=2/4);

VRULE
  ::S_DIGIT3(altezzaGreat>1.5)
  S_DIGIT4(altezzaGreat>1.5)
  ==>TIME(durate:=3/4);

VRULE
  ::S_DIGIT4(altezzaGreat>1.5)
  S_DIGIT4(altezzaGreat>1.5)
  ==>TIME(durate:=4/4);

VRULE
  ::S_DIGIT3(altezzaGreat>1.5)
  S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=3/8);

VRULE
  ::S_DIGIT6(altezzaGreat>1.5)
  S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=6/8);

VRULE
  S_DIGIT6(altezzaLess<2.5)
  ::S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=6/8);

VRULE
  ::S_DIGIT5(altezzaGreat>1.5)
  S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=5/8);

VRULE
  S_BEAM1(altezzaGreat>0.7)
  S_FNOTEHEAD(altezzaGreat>1)
  ::S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=5/8);
```

```
VRULE
  S_FNOTEHEAD(altezzaGreat>0.7)
  S_BEAM1(altezzaGreat>0.7)
  ::S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=5/8);

VRULE
  S_REST16
  ::S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=3/8);

VRULE
  S_STAFLINE
  S_FNOTEHEAD(altezzaGreat>1)
  ::S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=5/8);

VRULE
  S_FNOTEHEAD(altezzaLess<1)
  S_DIGIT4(altezzaLess<1.2)
  ::S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=5/8);

VRULE
  S_DIGIT5(altezzaGreat>1.5)
  ::S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=5/8);

VRULE
  ::S_DIGIT9(altezzaGreat>1.5)
  S_DIGIT8(altezzaGreat>1.5)
  ==>TIME(durate:=9/8);

VRULE
  S_DIGIT1(altezzaGreat>1.5)
  ::S_REST8(altezzaGreat>1.5)
  ==>TIME(durate:=1/8);

// ALTERAZIONI
VRULE
  ::S_SHARP(altezzaGreat>1.7)
  ==> ALTERAZIONE(isDiesis);

VRULE
  ::S_CHORD2V(altezzaGreat>2.2)
  ==> ALTERAZIONE(isDiesis);

VRULE
  ::S_FLAT(altezzaGreat>1.7)
  ==> ALTERAZIONE(isBemolle);

VRULE
  ::S_HOOK1UP(altezzaGreat>1.7,larghezzaLess<1.7,isAlone)
  ==> ALTERAZIONE(isBemolle);
```

```
VRULE
  ::S_NATURAL(altezzaGreat>1.7)
  ==> ALTERAZIONE(isBequadro);

VRULE
  ::S_DSHARP(larghezzaGreat>0.7)
  ==> ALTERAZIONE(isDdiesis);

VRULE
  ::S_DFLAT
  ==> ALTERAZIONE(isDbemolle);

// NOTE
VRULE
  ::S_ENOTEHEAD(pos=UprHalfStaff,altezzaGreat>1,notBeamed)
  [S_STAFLINESTEM]
  [S_STAFLINESTEM]
  ==> NOTE(stem:=Dwn,head:=Empty,durate:=2);

VRULE
  [S_STAFLINESTEM]
  [S_STAFLINESTEM]
  ::S_ENOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1,notBeamed)
  ==> NOTE(stem:=Up,head:=Empty,durate:=2);

VRULE
  ::S_ENOTEHEAD(altezzaGreat>1,notBeamed)
  ==> NOTE(stem:=Dwn,head:=Empty,durate:=2);

VRULE
  ::S_WHOLENOTE(larghezzaGreat>1,altezzaGreat>1,notBeamed)
  ==> NOTE(stem:=None,head:=Empty,durate:=1);

VRULE
  ::S_FNOTEHEAD(pos=UprHalfStaff,altezzaGreat>1,larghezzaGreat>1)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=4);

VRULE
  S_ACCENTO(altezzaGreat>0.65)
  ::S_FNOTEHEAD(pos=UprHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=4,withAccento);

VRULE
   ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1,larghezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=4);

VRULE
  ::S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.52,altezzaLess<1.3)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=8,inBeam);
```

```
VRULE
  ::S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  S_BEAM1(conf>0.0012,altezzaGreat>0.52)
  S_BEAM1(conf>0.0012,altezzaGreat>0.52)
  S_BEAM1(conf>0.0012,altezzaGreat>0.52)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=32,inBeam);

VRULE
  S_BEAM1(altezzaGreat>0.52,altezzaLess<1.3)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1,pos=LwrHalfStaff,isBeamed)
  ==> NOTE(stem:=Up,head:=Fill,durate:=8,inBeam);

VRULE
  ::S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  S_BEAM2(conf>0.0012,altezzaGreat>1.1,altezzaLess<2)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=16,inBeam);

VRULE
  S_BEAM2(altezzaGreat>1.1,altezzaLess<2)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=16,inBeam);

VRULE
  ::S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  S_BEAM3(conf>0.0012,altezzaGreat>2)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=32,inBeam);

VRULE
  S_BEAM3(conf>0.0012,altezzaGreat>2)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=32,inBeam);

VRULE
  ::S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  S_BEAM4(conf>0.0012,altezzaGreat>2.5)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=64,inBeam);

VRULE
  ::S_FNOTEHEAD(altezzaGreat>1)
  S_BEAM4(conf>0.0012,altezzaGreat>2.5)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=64,inBeam);

VRULE
  S_BEAM4(conf>0.0012,altezzaGreat>2.5)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=64,inBeam);
```

```
VRULE
  ::S_FNOTEHEAD(altezzaGreat>1)
  [S_STAFLINESTEM]
  S_BEAM5(conf>0.0012)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=128,inBeam);

VRULE
  S_BEAM5(conf>0.0012)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=128,inBeam);

VRULE
  ::S_FNOTEHEAD(pos=UprHalfStaff,altezzaGreat>1)
  [S_STAFLINESTEM]
  S_HOOK1UP
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=8);

VRULE
  S_HOOK1DWN(altezzaGreat>0.4,altezzaLess<1.3)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=8);

VRULE
  ::S_FNOTEHEAD(pos=UprHalfStaff,altezzaGreat>1)
  [S_STAFLINESTEM]
  S_HOOK2UP
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=16);

VRULE
  ::S_FNOTEHEAD(pos=OnStaffLine3,altezzaGreat>1)
  [S_STAFLINESTEM]
  S_HOOK2UP
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=16);

VRULE
  S_HOOK2DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=16);

VRULE
  S_HOOK2DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=OnStaffLine3,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=16);

VRULE
  ::S_FNOTEHEAD(pos=UprHalfStaff,altezzaGreat>1)
  [S_STAFLINESTEM]
  S_HOOK3UP(altezzaGreat>2)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=32);
```

```
VRULE
  S_HOOK3DWN(altezzaGreat>2)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=32);

VRULE
  ::S_FNOTEHEAD(pos=UprHalfStaff,altezzaGreat>1)
  [S_STAFLINESTEM]
  S_HOOK4UP
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=64);

VRULE
  S_HOOK4DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=64);

VRULE
  ::S_FNOTEHEAD(pos=UprHalfStaff,altezzaGreat>1)
  [S_STAFLINESTEM]
  S_HOOK5UP
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=128);

VRULE
  S_HOOK5DWN
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(pos=LwrHalfStaff,altezzaGreat>1)
  ==> NOTE(stem:=Up,head:=Fill,durate:=128);

VRULE
  ::S_W2NOTE(altezzaGreat>2)
  ==> NOTE(stem:=None,head:=Empty,durate:=8/4);


// ACCORDI
VRULE
  S_FNOTEHEAD(altezzaGreat>1,pos=outStaff)
  [S_STAFLINESTEM]
  ::NOTE(head=Fill,pos=UprHalfStaff)
  ==> NOTE(head:=Fill,durate:=4,stem:=Dwn,inAccordo);

VRULE
  ::NOTE(head=Fill,pos=UprHalfStaff,pos=outStaff)
  [S_STAFLINESTEM]
  S_FNOTEHEAD(altezzaGreat>1)
  ==> NOTE(stem:=Dwn,inAccordo);

VRULE
  ::NOTE(head=Fill)
  [S_STAFLINESTEM]
  S_FNOTEHEAD(altezzaGreat>1,pos=LwrHalfStaff,pos=outStaff)
  ==> NOTE(stem:=Up,inAccordo);
```

```
VRULE
 ::NOTE(durate=2)
 [STAFLINESTEM]
 S_ENOTEHEAD(altezzaGreat>1,pos=UprHalfStaff)
 ==> NOTE(stem:=Dwn,inAccordo,durate:=2);

VRULE
 ::NOTE(head=Empty)
 [STAFLINESTEM]
 NOTE(head=Empty,pos=LwrHalfStaff)
 ==> NOTE(inAccordo,stem:=Up);

VRULE
 S_ENOTEHEAD(altezzaGreat>1,pos=UprHalfStaff)
 [STAFLINESTEM]
 ::NOTE(durate=2)
 ==> NOTE(inAccordo,stem:=Dwn,head:=Empty,durate:=2);

VRULE
 S_ENOTEHEAD(altezzaGreat>1,pos=LwrHalfStaff)
 [STAFLINESTEM]
 ::NOTE(durate=2)
 ==> NOTE(inAccordo,stem:=Up,head:=Empty,durate:=2);

VRULE
  NOTE(durate=4)
  [STAFLINESTEM]
  ::NOTE(durate=4,pos=UprHalfStaff)
  ==> NOTE(inAccordo,stem:=Dwn,durate:=4);

VRULE
  ::NOTE(inAccordo,durate=4,pos=UprHalfStaff)
  [STAFLINESTEM]
  NOTE(durate=4)
  ==> NOTE(inAccordo,stem:=Dwn,durate:=4);

VRULE
  ::NOTE(stem=Up,head=Fill)
  [S_STAFLINESTEM]
  S_CHORD2V(altezzaGreat>2)
  ==> NOTE(inAccordo);

VRULE
  ::NOTE(stem=Dwn,head=Fill)
  [S_STAFLINESTEM]
  S_CHORD2V(altezzaGreat>2)
  ==> NOTE(inAccordo);

VRULE
  S_CHORD2V(altezzaGreat>2)
  [S_STAFLINESTEM]
  ::NOTE(head=Fill)
  ==> NOTE(stem:=Dwn,inAccordo);
```

```
VRULE
  S_CHORD2V(altezzaGreat>2)
  [S_STAFLINESTEM]
  ::NOTE(stem=Dwn,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  ::NOTE(stem=Up,head=Fill)
  [S_STAFLINESTEM]
  S_CHORD3V(altezzaGreat>3)
  ==> NOTE(inAccordo);

VRULE
  ::NOTE(head=Empty)
  [S_STAFLINESTEM]
  S_CHORDE2V(altezzaGreat>2,larghezzaGreat>1)
  ==> NOTE(stem:=Dwn,inAccordo);

VRULE
  ::NOTE(stem=Up,head=Fill)
  [S_STAFLINESTEM]
  NOTE(stem=Up,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  S_FNOTEHEAD(altezzaGreat>1,conf>0.5)
  [S_STAFLINESTEM]
  ::NOTE(head=Fill,stem=Dwn,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  ::NOTE(stem=Dwn,head=Fill)
  [S_STAFLINESTEM]
  S_FNOTEHEAD(altezzaGreat>1,conf>0.5)
  ==> NOTE(inAccordo);

VRULE
  S_CHORD3V(altezzaGreat>3)
  [S_STAFLINESTEM]
  ::NOTE(stem=Dwn,head=Fill)
  ==> NOTE(inAccordo);

VRULE
  S_CHORDE2V(altezzaGreat>2,larghezzaGreat>1)
  [S_STAFLINESTEM]
  ::NOTE(stem=Dwn,head=Empty)
  ==> NOTE(inAccordo);

VRULE
  ::NOTE(stem=Dwn,head=Fill)
  [S_STAFLINESTEM]
  NOTE(stem=Dwn,head=Fill)
  ==> NOTE(inAccordo);
```

```
VRULE
 ::S_CHORDE2V(altezzaGreat>2,larghezzaGreat>1,isAlone,notBeamed,pos=UprHalStaff)
 [STAFLINESTEM]
 ==>NOTE(inAccordo,head:=empty,stem:=Dwn,durate:=2);

VRULE
 ::S_CHORDE2V(altezzaGreat>2,larghezzaGreat>1,withoutNote,notBeamed,pos=LwrHalStaff)
 [STAFLINESTEM]
 ==>NOTE(inAccordo,head:=empty,stem:=Up,durate:=2);

VRULE
  S_ENOTEHEAD(altezzaGreat>0.8,larghezzaGreat>0.8,pos=UprHalStaff)
  [S_STAFLINESTEM]
  ::S_CHORDE2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2)
  ==> NOTE(inAccordo,head:=Empty,stem:=Dwn,durate:=2);

VRULE
  ::S_CHORDE2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=UprHalStaff)
  [S_STAFLINESTEM]
  S_ENOTEHEAD(altezzaGreat>0.8,larghezzaGreat>0.8)
  ==> NOTE(inAccordo,head:=Empty,stem:=Dwn,durate:=2);

VRULE
 ::S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=UprHalStaff)
 [STAFLINESTEM]
 ==>NOTE(inAccordo,head:=Fill,stem:=Dwn,durate:=4);

VRULE
 ::S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=UprHalStaff)
 [STAFLINESTEM]
 S_BEAM1(altezzaGreat>0.4,altezzaLess<1.3)
 ==>NOTE(inAccordo,head:=Fill,stem:=Dwn,durate:=8);

VRULE
 S_BEAM1(altezzaGreat>0.4,altezzaLess<1.3)
 [STAFLINESTEM]
 ::S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=LwrHalStaff)
 ==>NOTE(inAccordo,head:=Fill,stem:=Up,durate:=8);

VRULE
 ::S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=UprHalStaff)
 [STAFLINESTEM]
 S_BEAM1(altezzaGreat>0.4,altezzaLess<1.3)
 S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
 ==>NOTE(inAccordo,head:=Fill,stem:=Dwn,durate:=32);

VRULE
 S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
 S_BEAM1(altezzaGreat>0.4,altezzaLess<1.3)
 [STAFLINESTEM]
 ::S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=LwrHalStaff)
 ==>NOTE(inAccordo,head:=Fill,stem:=Up,durate:=32);
```

```
VRULE
  ::S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=LwrHalStaff)
  [S_STAFLINESTEM]
  S_BEAM3(altezzaGreat>2,altezzaLess<3)
  ==> NOTE(inAccordo,inBeam,head:=Fill,stem:=Dwn,durate:=32);

VRULE
  ::S_FNOTEHEAD(altezzaGreat>0.8,larghezzaGreat>0.8)
  [S_STAFLINESTEM]
  S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=LwrHalStaff)
  [S_STAFLINESTEM]
  S_BEAM3(altezzaGreat>2,altezzaLess<3)
  ==> NOTE(inAccordo,inBeam,head:=Fill,stem:=Dwn,durate:=32);

VRULE
  S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=LwrHalStaff)
  [S_STAFLINESTEM]
  ::S_FNOTEHEAD(altezzaGreat>0.8,larghezzaGreat>0.8)
  [S_STAFLINESTEM]
  S_BEAM3(altezzaGreat>2,altezzaLess<3)
  ==> NOTE(inAccordo,inBeam,head:=Fill,stem:=Dwn,durate:=32);

VRULE
  ::S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,larghezzaLess<2,pos=LwrHalStaff)
  [S_STAFLINESTEM]
  S_FNOTEHEAD(altezzaGreat>0.8,larghezzaGreat>0.8)
  [S_STAFLINESTEM]
  S_BEAM3(altezzaGreat>2,altezzaLess<3)
  ==> NOTE(inAccordo,inBeam,head:=Fill,stem:=Dwn,durate:=32);

VRULE
  S_BEAM3(altezzaGreat>2,altezzaLess<3)
  [S_STAFLINESTEM]
  ::S_CHORD2V(altezzaGreat>2,larghezzaGreat>0.8,pos=LwrHalStaff)
  ==> NOTE(inAccordo,inBeam,head:=Fill,stem:=Up,durate:=32);

VRULE
 ::S_CHORD3V(altezzaGreat>2.7,larghezzaGreat>1,pos=UprHalStaff)
 ==>NOTE(inAccordo,head:=Fill,stem:=Dwn,durate:=4);

VRULE
 ::S_CHORD3V(altezzaGreat>2.7,larghezzaGreat>1,pos=LwrHalStaff)
 ==>NOTE(inAccordo,head:=Fill,stem:=Up,durate:=4);

VRULE
 ::S_CHORD3V(altezzaGreat>2.7,larghezzaGreat>1,pos=UprHalStaff)
 S_BEAM3(altezzaGreat>2,altezzaLess<3)
 ==>NOTE(inAccordo,head:=Fill,stem:=Dwn,durate:=32,inBeam);

VRULE
 ::S_CHORD3V(altezzaGreat>2.7,larghezzaGreat>1,pos=UprHalStaff)
 S_BEAM1(altezzaGreat>0.56,altezzaLess<1.3)
 S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
 ==>NOTE(inAccordo,head:=Fill,stem:=Dwn,durate:=32,inBeam);
```

```
VRULE
 S_BEAM3(altezzaGreat>2,altezzaLess<3)
 ::S_CHORD3V(altezzaGreat>2.7,larghezzaGreat>1,pos=LwrHalStaff)
 ==>NOTE(inAccordo,head:=Fill,stem:=Up,durate:=32,inBeam);

// BEAM
VRULE
  ::NOTE(stem=Dwn,durate=4)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.56,altezzaLess<1.3)
  ==> NOTE(inBeam, durate:=8);

VRULE
  S_BEAM1(altezzaGreat>0.56,altezzaLess<1.3)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,durate=4,isBeamed)
  ==> NOTE(inBeam, durate:=8);

VRULE
  ::NOTE(stem=Dwn,durate=4)
  [S_STAFLINESTEM]
  S_BEAM2(conf>0.0012,altezzaGreat>1.3,altezzaLess<2)
  ==> NOTE(inBeam, durate:=16);

VRULE
  S_BEAM2(conf>0.0012,altezzaGreat>1.3,altezzaLess<2)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up, durate=4)
  ==> NOTE(inBeam, durate:=16);

VRULE
  ::NOTE(stem=Dwn, durate=4)
  [S_STAFLINESTEM]
  S_BEAM3(altezzaGreat>2,altezzaLess<3)
  ==> NOTE(inBeam, durate:=32);

VRULE
  S_BEAM3(altezzaGreat>2,altezzaLess<3)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up, durate=4)
  ==> NOTE(inBeam, durate:=32);

VRULE
  ::NOTE(stem=Dwn, durate=8)
  [S_STAFLINESTEM]
  S_BEAM1(conf>0.0012,altezzaGreat>0.56)
  ==> NOTE(inBeam, durate:=16);

VRULE
  S_BEAM1(conf>0.0012,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up, durate=8)
  ==> NOTE(inBeam, durate:=16);
```

```
VRULE
  ::NOTE(stem=Dwn, durate=16)
  [S_STAFLINESTEM]
  S_BEAM1(conf>0.0012,altezzaGreat>0.56)
  ==> NOTE(inBeam, durate:=32);

VRULE
  S_BEAM1(conf>0.12,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up, durate=16)
  ==> NOTE(inBeam, durate:=32);

VRULE
  ::NOTE(stem=Dwn, durate=32)
  [S_STAFLINESTEM]
  S_BEAM1(conf>0.0012,altezzaGreat>0.56)
  ==> NOTE(inBeam, durate:=64);

VRULE
  S_BEAM1(conf>0.0012,altezzaGreat>0.56)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up, durate=32)
  ==> NOTE(inBeam, durate:=64);

VRULE
  ::NOTE(stem=Dwn, durate=8)
  [S_STAFLINESTEM]
  S_BEAM3(conf>0.0012,altezzaGreat>2)
  ==> NOTE(inBeam, durate:=64);

VRULE
  S_BEAM3(conf>0.0012,altezzaGreat>2)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up, durate=8)
  ==> NOTE(inBeam, durate:=64);

VRULE
  ::NOTE(stem=Dwn, durate=8)
  [S_STAFLINESTEM]
  S_BEAM2(conf>0.0012,altezzaGreat>1.3,altezzaLess<2)
  ==> NOTE(inBeam, durate:=32);

VRULE
  S_BEAM2(conf>0.0012,altezzaGreat>1.3,altezzaLess<2)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up, durate=8)
  ==> NOTE(inBeam, durate:=32);

VRULE
  ::NOTE(durate=4,stem=Up)
  [S_STAFLINESTEM]
  S_BEAM2(conf>0.0012,altezzaGreat>1.3,altezzaLess<2)
  ==> NOTE(stem:=Dwn, inBeam, durate:=16);
```

```
VRULE
  S_BEAM2(conf>0.0012,altezzaGreat>1.3,altezzaLess<2)
  [S_STAFLINESTEM]
  ::NOTE(durate=4,stem=Dwn)
  ==> NOTE(stem:=Up,inBeam, durate:=16);

VRULE
  ::NOTE(durate=4)
  [S_STAFLINESTEM]
  S_BEAM3(conf>0.0012,altezzaGreat>2)
  ==> NOTE(stem:=Dwn,inBeam, durate:=32);

VRULE
  S_BEAM3(conf>0.0012,altezzaGreat>2)
  [S_STAFLINESTEM]
  ::NOTE(durate=4)
  ==> NOTE(stem:=Up,inBeam, durate:=32);

VRULE
  ::NOTE(durate=4)
  [S_STAFLINESTEM]
  S_BEAM4(conf>0.0012,altezzaGreat>2.5)
  ==> NOTE(inBeam, durate:=64);

VRULE
  S_BEAM4(conf>0.0012,altezzaGreat>2.5)
  [S_STAFLINESTEM]
  ::NOTE(durate=4)
  ==> NOTE(inBeam, durate:=64);

VRULE
  ::NOTE(durate=4)
  [S_STAFLINESTEM]
  S_BEAM5(conf>0.0012,altezzaGreat>3)
  ==> NOTE(inBeam, durate:=128);

VRULE
  S_BEAM5(conf>0.0012,altezzaGreat>3)
  [S_STAFLINESTEM]
  ::NOTE
  ==> NOTE(inBeam, durate:=128);

VRULE
  ::NOTE(durate=4)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=16);

VRULE
  ::NOTE(durate=4)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.4,altezzaLess<1.3)
```

```
  S_BEAM1(altezzaGreat>0.4,altezzaLess<1.3)
  S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=64);

VRULE
  ::NOTE(durate=8,stem=Dwn)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=32);

VRULE
  ::NOTE(durate=8,stem=Up)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.4,conf>0.7)
  S_BEAM2(altezzaGreat>1.3,altezzaLess<2,conf>0.7)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=32);

VRULE
  ::NOTE(durate=4,stem=Dwn)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=32);

VRULE
  S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
  S_BEAM1(altezzaGreat>0.4)
  ::NOTE(durate=4)
  ==> NOTE(stem:=Up,head:=Fill,durate:=32);

VRULE
  S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
  S_HOOK1UP(altezzaGreat>0.4)
  ::NOTE(durate=4)
  ==> NOTE(stem:=Up,head:=Fill,durate:=32);


VRULE
  ::NOTE(durate=16,stem=Dwn)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.4)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=32);

VRULE
  ::NOTE(durate=4)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=32);
```

```
VRULE
  ::NOTE(durate=4)
  [S_STAFLINESTEM]
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=64);

VRULE
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  ::NOTE(durate=4)
  ==> NOTE(stem:=Up,head:=Fill,durate:=32);

VRULE
  S_BEAM1(altezzaGreat>0.4)
  S_STAFLINE(altezzaGreat>0.4)
  S_BEAM1(altezzaGreat>0.4)
  ::NOTE(durate=4)
  ==> NOTE(stem:=Up,head:=Fill,durate:=32);

VRULE
  ::NOTE(durate=4)
  S_BEAM1(altezzaGreat>0.52)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=8);

VRULE
  ::NOTE(durate=16,stem=Dwn)
  S_BEAM1(altezzaGreat>0.4,altezzaLess<1.3)
  S_BEAM1(altezzaGreat>0.4,altezzaLess<1.3)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=64);

VRULE
  S_BEAM1(conf>0.80,altezzaGreat>0.56,altezzaLess<1.3)
  ::NOTE(durate=4,stem=Dwn)
  ==> NOTE(stem:=Up,head:=Fill,durate:=8);

VRULE
  S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
  ::NOTE(durate=4)
  ==> NOTE(stem:=Up,head:=Fill,durate:=16);

VRULE
  ::NOTE(durate=4)
  S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=16);

VRULE
  ::NOTE(durate=16)
  S_BEAM2(altezzaGreat>1.3,altezzaLess<2)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=64);
```

```
VRULE
  ::NOTE(durate=4)
  S_BEAM1(altezzaLess<1)
  S_BEAM3(altezzaGreat>2)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=64);

VRULE
  S_BEAM3(altezzaGreat>2)
  S_BEAM1(altezzaLess<1)
  ::NOTE(durate=4)
  ==> NOTE(stem:=Up,head:=Fill,durate:=64);


// HOOK
VRULE
  ::NOTE(stem=Dwn,head=Fill,durate=4)
  [S_STAFLINESTEM]
  S_HOOK1UP
  ==> NOTE(durate:=8);

VRULE
  ::NOTE(stem=Dwn,head=Fill,durate=4)
  [S_STAFLINESTEM]
  S_TDF(altezzaLess<0.6)
  ==> NOTE(durate:=8);

VRULE
  S_HOOK1DWN(altezzaLess<1)
  ::NOTE(stem=Up,head=Fill,durate=4)
  ==> NOTE(durate:=8);

VRULE
  ::NOTE(stem=Dwn,head=Fill,durate=4)
  [S_STAFLINESTEM]
  S_HOOK2UP
  ==> NOTE(durate:=16);

VRULE
  S_HOOK2DWN
  ::NOTE(stem=Up,head=Fill,durate=4)
  ==> NOTE(durate:=16);

VRULE
  ::NOTE(stem=Dwn,head=Fill,durate=4)
  [S_STAFLINESTEM]
  S_HOOK3UP(altezzaGreat>2)
  ==> NOTE(durate:=32);

VRULE
  S_HOOK3DWN(altezzaGreat>2,durate=4)
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(durate:=32);
```

```
VRULE
  ::NOTE(stem=Dwn,head=Fill,durate=4)
  [S_STAFLINESTEM]
  S_HOOK4UP
  ==> NOTE(durate:=64);

VRULE
  S_HOOK4DWN
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill,durate=4)
  ==> NOTE(durate:=64);

VRULE
  ::NOTE(stem=Dwn,head=Fill)
  [S_STAFLINESTEM]
  S_HOOK5UP
  ==> NOTE(durate:=128);

VRULE
  S_HOOK5DWN
  [S_STAFLINESTEM]
  ::NOTE(stem=Up,head=Fill)
  ==> NOTE(durate:=128);

// SIMBOLI
VRULE
  S_ACCENTO(altezzaGreat>0.7)
  ::NOTE(stem=Dwn)
  ==> NOTE(withAccento);

VRULE
  ::NOTE(stem=Up)
  S_ACCENTO(altezzaGreat>0.7)
  ==> NOTE(withAccento);

VRULE
  S_TENUTO
  ::NOTE(stem=Dwn)
  ==> NOTE(withTenuto);

VRULE
  S_STACCATO(altezzaLess<0.6)
  ::NOTE(stem=Dwn)
  ==> NOTE(withStaccato);

VRULE
  ::NOTE(stem=Up)
  S_TENUTO
  ==> NOTE(withTenuto);

VRULE
  ::NOTE(stem=Up)
  S_STACCATO(altezzaLess<0.6)
  ==> NOTE(withStaccato);
```

```
VRULE
  ::NOTE(stem=Up)
  [S_STAFLINE]
  SIMBOLO
  ==> NOTE;

VRULE
  SIMBOLO(isAccento)
  ::NOTE(stem=Dwn)
  ==> NOTE(withAccento);

VRULE
  SIMBOLO(isAccento)
  ::NOTE(stem=Up)
  ==> NOTE(withAccento);

VRULE
  S_ACCENTO(altezzaGreat>0.9)
  ::NOTE(stem=Up)
  ==> NOTE(withAccento);

VRULE
  ::NOTE(stem=Up)
  SIMBOLO(isAccento)
  ==> NOTE(withAccento);

VRULE
  NOTE(stem=Up)
  ::SIMBOLO(isAccento)
  ==> NOTE(withAccento,correctPos);

VRULE
  ::NOTE
  S_TDP(pos=outStaff)
  ==> NOTE(withPiano);

VRULE
  S_FNOTEHEAD(altezzaGreat>1)
  ::SIMBOLO(isAccento)
  ==> NOTE(durate:=4,stem:=Up,head:=Fill,withAccento,correctPos);



// PAUSE
VRULE
  ::S_W2REST(altezzaGreat>0.6)
  ==> PAUSE(durate:=8/4);

VRULE
  ::S_WREST(notBeamed,withoutNote,isAlone,altezzaLess<0.9,
    altezzaGreat>0.4,larghezzaGreat>0.5,larghezzaLess<1.6,pos=Rest2pos)
  ==> PAUSE(durate:=2);
```

```
VRULE
  ::S_BEAM1(notBeamed,withoutNote,isAlone,altezzaLess<0.9,
    altezzaGreat>0.4,larghezzaGreat>0.5,larghezzaLess<1.6,pos=Rest2pos)
  ==> PAUSE(durate:=2);

VRULE
  ::S_WREST(notBeamed,withoutNote,isAlone,altezzaLess<0.9,
    altezzaGreat>0.4,larghezzaGreat>0.5,larghezzaLess<1.6,pos=Rest1pos)
  ==> PAUSE(durate:=1);

VRULE
  S_STAFLINE
  S_STAFLINE
  ::S_REST2
  S_STAFLINE
  S_STAFLINE
  ==> PAUSE(durate:=2);

VRULE
  ::S_REST4(larghezzaGreat>0.7,altezzaLess<4,notBeamed)
  ==> PAUSE(durate:=4);

VRULE
  ::S_REST4(larghezzaGreat>0.7,altezzaGreat>3,altezzaLess<4,isBeamed,conf>0.8)
  ==> PAUSE(durate:=4);

VRULE
  ::S_REST8(altezzaGreat>0.4, isAlone, notBeamed)
  ==> PAUSE(durate:=8);

VRULE
  ::S_REST8(altezzaGreat>1.5, larghezzaGreat>1,notAlone)
  ==> PAUSE(durate:=8);

VRULE
  ::S_HALFREST8(altezzaGreat>0.5,conf>0.1,larghezzaGreat>1,notAlone)
  ==> PAUSE(durate:=8);

VRULE
  ::S_HALFREST8(altezzaGreat>0.4, isAlone, notBeamed)
  ==> PAUSE(durate:=8);

VRULE
  ::S_HALFREST8(altezzaGreat>0.45)
  S_HOOK1UP(altezzaGreat>0.45)
  ==> PAUSE(durate:=16);

VRULE
  ::PAUSE(durate=8)
  S_HOOK1UP(altezzaGreat>0.4)
  ==> PAUSE(durate:=16);
```

```
VRULE
  ::S_REST8(altezzaGreat>0.45)
  S_HALFREST8(altezzaGreat>0.45)
  ==> PAUSE(durate:=16);

VRULE
  S_HALFREST8(altezzaGreat>0.45)
  ::S_REST8(altezzaGreat>0.45)
  ==> PAUSE(durate:=16);

VRULE
  ::S_REST16(altezzaGreat>1.7)
  ==> PAUSE(durate:=16);

VRULE
  ::S_REST32(altezzaGreat>2,larghezzaGreat>1)
  ==> PAUSE(durate:=32);

VRULE
  ::S_REST64(altezzaGreat>2.5)
  ==> PAUSE(durate:=64);

VRULE
  S_REST8(altezzaGreat>0.4)
  ::PAUSE(durate=8)
  ==> PAUSE(durate:=16);

VRULE
  S_HALFREST8(altezzaGreat>0.45)
  ::PAUSE(durate=8)
  ==> PAUSE(durate:=16);

VRULE
  ::PAUSE(durate=8)
  S_REST8(altezzaGreat>0.5)
  ==> PAUSE(durate:=16);

VRULE
  ::PAUSE(durate=8)
  S_REST16(altezzaGreat>1.5)
  ==> PAUSE(durate:=32);

// SIMBOLI
VRULE
  ::S_TDP(isAlone,pos=outStaff,pos=LwrHalfStaff)
  ==> SIMBOLO(isPiano);

VRULE
  ::S_TDF(isAlone,pos=outStaff,pos=LwrHalfStaff)
  ==> SIMBOLO(isForte);

VRULE
  ::S_ACCENTO(altezzaGreat>0.7)
  ==> SIMBOLO(isAccento);
```

```
VRULE
  ::S_CORONA
  ==> SIMBOLO(isCorona);

VRULE
 ::S_PUNTVAL(altezzaLess<0.6)
 ==> SIMBOLO(isPunto);

VRULE
  ::S_FNOTEHEAD(altezzaLess<0.6)
  ==> SIMBOLO(isPunto);

VRULE
  ::S_FNOTEHEAD(altezzaLess<0.8, larghezzaLess<0.8)
  ==> SIMBOLO(isPunto);

//REGOLE PER STRIP SUDICE
VRULE
  S_BEAM1(altezzaGreat>0.56)
  [S_STAFLINE]
  ::ALTERAZIONE
  ==> ALTERAZIONE;

VRULE
  S_BEAM2
  [S_STAFLINE]
  ::ALTERAZIONE
  ==> ALTERAZIONE;

VRULE
  S_BEAM3
  [S_STAFLINE]
  ::ALTERAZIONE
  ==> ALTERAZIONE;

VRULE
  S_BEAM4
  [S_STAFLINE]
  ::ALTERAZIONE
  ==> ALTERAZIONE;

VRULE
  S_BEAM5
  [S_STAFLINE]
  ::ALTERAZIONE
  ==> ALTERAZIONE;

VRULE
  ::ALTERAZIONE
  [S_STAFLINE]
  S_BEAM1(altezzaGreat>0.56)
  ==> ALTERAZIONE;
```

```
VRULE
  ::ALTERAZIONE
  [S_STAFLINE]
  S_BEAM2
  ==> ALTERAZIONE;

VRULE
  ALTERAZIONE
  [S_STAFLINE]
  ::S_BEAM2
  ==> ALTERAZIONE;

VRULE
  ::ALTERAZIONE
  [S_STAFLINE]
  S_BEAM3
  ==> ALTERAZIONE;

VRULE
  ::ALTERAZIONE
  [S_STAFLINE]
  S_BEAM4
  ==> ALTERAZIONE;

VRULE
  ::ALTERAZIONE
  [S_STAFLINE]
  S_BEAM5
  ==> ALTERAZIONE;

VRULE
  S_BEAM1(altezzaGreat>0.56)
  [S_STAFLINE]
  ::SIMBOLO
  ==> SIMBOLO;

VRULE
  S_BEAM2
  [S_STAFLINE]
  ::SIMBOLO
  ==> SIMBOLO;

VRULE
  S_BEAM3
  [S_STAFLINE]
  ::SIMBOLO
  ==> SIMBOLO;

VRULE
  S_BEAM4
  [S_STAFLINE]
  ::SIMBOLO
  ==> SIMBOLO;
```

```
VRULE
  S_BEAM5
  [S_STAFLINE]
  ::SIMBOLO
  ==> SIMBOLO;

VRULE
  ::SIMBOLO
  [S_STAFLINE]
  S_BEAM1(altezzaGreat>0.56)
  ==> SIMBOLO;

VRULE
  ::SIMBOLO
  [S_STAFLINE]
  S_BEAM2
  ==> SIMBOLO;

VRULE
  ::SIMBOLO
  [S_STAFLINE]
  S_BEAM3
  ==> SIMBOLO;

VRULE
  ::SIMBOLO
  [S_STAFLINE]
  S_BEAM4
  ==> SIMBOLO;

VRULE
  ::SIMBOLO
  [S_STAFLINE]
  S_BEAM5
  ==> SIMBOLO;

VRULE
  ::SIMBOLO
  [S_STAFLINE]
  S_SLURSTAFLINE(altezzaLess<0.7)
  ==> SIMBOLO;

VRULE
  S_SLURSTAFLINE(altezzaLess<0.7)
  [S_STAFLINE]
  ::SIMBOLO
  ==> SIMBOLO;

\\REGOLE PER CONFIGURAZIONI PARTICOLARI

VRULE
 ::S_ACCENTO (conf<0.3,altezzaLess<0.7)
 ==> LINE;
```

```
VRULE
 ::S_STAFLINESLUR (larghezzaGreat>0.56)
 S_BEAM1(withoutNote,isBeamed)
 S_BEAM1
 ==> LINE;

VRULE
 ::S_STAFLINESLUR (larghezzaGreat>0.56)
 S_BEAM1(withoutNote,isBeamed)
 S_BEAM2(altezzaGreat>1.3)
 ==> LINE;

VRULE
 ::S_STAFLINESLUR (larghezzaGreat>0.9)
 S_BEAM1(withoutNote,isBeamed)
 ==> LINE;

VRULE
 S_STAFLINESLUR(larghezzaGreat>0.9)
 ::S_BEAM1(withoutNote,isBeamed)
 ==> LINE;

VRULE
  ::S_FNOTEHEAD(altezzaGreat>1)
  S_TDP(altezzaGreat>1,pos=insideStaff)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=16);

VRULE
  ::NOTE(durate=4)
  S_TDP(altezzaGreat>1,pos=insideStaff)
  ==> NOTE(stem:=Dwn,head:=Fill,durate:=16);


VRULE
  ::S_STAFLINESLUR(larghezzaGreat>1.3,withoutNote,isAlone,notBeamed)
  ==> LINE(isAlone);

VRULE
  ::S_STAFLINESLUR(larghezzaGreat>0.6,larghezzaLess<1.3,altezzaGreat>0.4,
   withoutNote,isAlone,pos=insideStaff,notBeamed)
  ==> LINE(isAlone);

VRULE
   ::S_PUNTVAL(pos=OnSpace2)
   S_PUNTVAL(pos=OnSpace3,withoutNote)
   ==> SIMBOLO(isDoublePunto);


****************** REGOLE ORIZZONTALI **********************

// SIMBOLI COLLEGATI A ALTERAZIONI
HRULE
  ALTERAZIONE(dxMax=2,dyMax=2) ::NOTE ==> NOTE(inAlterazione);
```

```
HRULE
  ALTERAZIONE(dxMax=2) ::NOTE(inAccordo) ==> NOTE(inAlterazione);

HRULE
  ::ALTERAZIONE(dxMax=2,dyMax=0.7) ALTERAZIONE ==> ALTERAZIONE;

HRULE
  ::ALTERAZIONE(dxMax=0.7) ALTERAZIONE ==> ALTERAZIONE;

HRULE
  ::CLEF(dxMax=1.4,dyMax=2) ALTERAZIONE ==> INTESTAZIONE;

HRULE
  ::INTESTAZIONE(dxMax=0.7) ALTERAZIONE ==> INTESTAZIONE;

// NOTE  E PAUSE COLLEGATI A PUNTI DI VALORE
HRULE
 ::NOTE(dxMax=1.2,dyMax=1) S_PUNTVAL ==> NOTE(isPunto);

HRULE
 ::NOTE(dxMax=1.2,dyMax=1) SIMBOLO(isPunto) ==> NOTE;

HRULE
 ::NOTE(dxMax=1.2,inAccordo) SIMBOLO(isPunto) ==> NOTE;

HRULE
  NOTE(dxMax=1.2,dyMax=1) ::SIMBOLO(isPunto) ==> NOTE;

HRULE
 ::NOTE(isPunto,dxMax=1,dyMax=1) SIMBOLO(isPunto) ==> NOTE;

HRULE
  ::PAUSE(dxMax=3,dyMax=2.5) SIMBOLO(isPunto) ==> PAUSE;

HRULE
  PAUSE(dxMax=3,dyMax=2.5) ::SIMBOLO(isPunto) ==> PAUSE;

HRULE
  ::PAUSE(isPunto,dyMax=2) SIMBOLO(isPunto) ==> PAUSE;

HRULE
 ::PAUSE(dxMax=1.5,dyMax=2) S_PUNTVAL(isAlone) ==> PAUSE(isPunto);

HRULE
 ::PAUSE(dxMax=2,dyMax=2.5) S_PUNTVAL ==> PAUSE(isPunto);


\\REGOLE DI RICOSTRUZIONE UNCINI DISCENDENTI
HRULE
  ::NOTE(dxMax=1,dyMax=5,durate=4,notBeamed) S_HOOK1DWN(altezzaGreat>1) ==> NOTE(durate:=8);

HRULE
  ::NOTE(dxMax=1,dyMax=5,durate=4,notBeamed)
S_HOOK3DWN(altezzaGreat>1,altezzaLess<3) ==> NOTE(durate:=8);
```

```
HRULE
  ::NOTE(dxMax=1,dyMax=5,durate=4,notBeamed) S_REST8(altezzaGreat>1) ==> NOTE(durate:=8);

HRULE
  ::NOTE(dxMax=1,dyMax=5,durate=4,notBeamed) PAUSE(durate=8) ==> NOTE(durate:=8);

HRULE
  ::NOTE(dxMax=1,dyMax=5,durate=4,notBeamed) S_REST4 ==> NOTE(durate:=8);

HRULE
  ::NOTE(dxMax=1,dyMax=5,durate=4,notBeamed) PAUSE(durate=4) ==> NOTE(durate:=8);

HRULE
  ::NOTE(dxMax=1,dyMax=5,durate=4,notBeamed) S_REST16(altezzaGreat>1.5) ==> NOTE(durate:=16);


// SIMBOLI CHE FORMANO LINEE
HRULE
  ::S_CRESCBEGIN(dyMax=1) S_CRESCLINE ==> LINE;

HRULE
  ::S_CRESCLINE(dyMax=1) S_CRESCLINE ==> LINE;

HRULE
  ::LINE(dyMax=1) S_CRESCLINE ==> LINE;

HRULE
  S_DECRESCLINE(dyMax=1) ::S_DECREEND ==> LINE;

HRULE
  ::LINE(dyMax=1) S_DECREEND ==> LINE;

HRULE
  ::S_DECRESCLINE(dyMax=1) S_DECRESCLINE ==> LINE;

HRULE
  ::LINE(dyMax=1) S_DECRESCLINE ==> LINE;

HRULE
  ::LINE(dyMax=1) ::LINE ==> LINE;

HRULE
  ::S_SLUR(dyMax=1) S_SLUR ==> LINE;

HRULE
  S_SLUR(dyMax=1) ::S_SLUR ==> LINE;

HRULE
  ::S_SLUR(dyMax=1) ::LINE ==> LINE;

HRULE
  ::LINE(dyMax=1) ::S_SLUR ==> LINE;
```

```
HRULE
  ::S_SLURSTAFLINE(dyMax=1) S_STAFLINESLUR ==> LINE;

HRULE
  ::S_STAFLINESLUR(dyMax=1) S_SLURSTAFLINE ==> LINE;

HRULE
  ::LINE(dyMax=1) ::S_STAFLINESLUR ==> LINE;

HRULE
  ::LINE(dyMax=1) ::S_SLURSTAFLINE ==> LINE;

HRULE
  ::S_SLUR(dyMax=1) ::S_STAFLINESLUR ==> LINE;

HRULE
  ::S_SLUR(dyMax=1) ::S_SLURSTAFLINE ==> LINE;

HRULE
  ::S_SLURSTAFLINE(dyMax=1) ::S_SLUR ==> LINE;

HRULE
  ::S_STAFLINESLUR(dyMax=1) ::S_SLUR ==> LINE;

HRULE
  ::S_SLURSTAFLINE(dyMax=1) ::LINE ==> LINE;

HRULE
  ::S_STAFLINESLUR(dyMax=1) ::LINE ==> LINE;

HRULE
  S_PUNTVAL(dyMax=0.5) ::LINE ==> LINE;

HRULE
  ::LINE(dyMax=0.5) S_PUNTVAL ==> LINE;

HRULE
  SIMBOLO(isPunto,dyMax=0.5) ::LINE ==> LINE;

HRULE
  ::LINE(dyMax=0.5) SIMBOLO(isPunto) ==> LINE;



// REGOLE PER LE BARRE DI RITORNELLO
HRULE
 S_PUNTVAL(pos=OnSpace2,dxMax=1) ::BARLINE(isEnd) ==> BARLINE(isRepeat);

HRULE
 S_PUNTVAL(pos=OnSpace3,dxMax=1) ::BARLINE(isEnd) ==> BARLINE(isRepeat);

HRULE
  ::BARLINE(isRepeat) S_PUNTVAL(pos=OnSpace2,dxMax=1) ==> BARLINE(isStart);
```

```
HRULE
  ::BARLINE(isRepeat) S_PUNTVAL(pos=OnSpace3,dxMax=1) ==> BARLINE(isStart);

HRULE
 S_PUNTVAL(pos=OnSpace2,dxMax=1) ::BARLINE(isDouble) ==> BARLINE(isEnd,isRepeat);

HRULE
 S_PUNTVAL(pos=OnSpace3,dxMax=1) ::BARLINE(isDouble) ==> BARLINE(isEnd,isRepeat);

HRULE
 SIMBOLO(isDoublePunto,dxMax=1) ::BARLINE(isEnd) ==> BARLINE(isRepeat);

HRULE
 SIMBOLO(isDoublePunto,dxMax=1) ::BARLINE(isDouble) ==> BARLINE(isEnd,isRepeat);

HRULE
 ::BARLINE(isEnd) ::SIMBOLO(isDoublePunto,dxMax=1)  ==> BARLINE(isStart);

HRULE
 BARLINE(isSingle,dxMax=1) ::BARLINE(isEnd) ==> BARLINE(isEnd);

HRULE
 BARLINE(isEnd,dxMax=1) ::BARLINE(isSingle) ==> BARLINE(isStart);

HRULE
 ::BARLINE(isSingle) ::S_HOOK3DWN(altezzaGreat>3,dxMax=1) ==> BARLINE(isEnd);

HRULE
 BARLINE(isSingle,dxMax=1) ::BARLINE(isSingle) ==> BARLINE(isDouble);

HRULE
 BARLINE(isSingle,dxMax=1) ::BARLINE(isDouble) ==> BARLINE(isDouble);


// REGOLE PER RICOSTRUIRE CHIAVI DI DO

HRULE
 S_BEAM3(dxMax=1) ::S_CHORD2V(pos=OnSpace2,pos=UprHalfStaff) ==> CLEF(isTenore);

HRULE
 S_BEAM3(dxMax=1,pos=LwrHalfStaff) ::S_CHORD2V(pos=OnSpace4) ==> CLEF(isMezzosoprano);

HRULE
 S_BEAM3(dxMax=1,pos=LwrHalfStaff) ::S_CHORD2V(pos=OnSpace5) ==> CLEF(isSoprano);

HRULE
 S_CCLLOW(dxMax=1.5,pos=LwrHalfStaff) ::S_CHORD2V(pos=outStaff) ==> CLEF(isSoprano);

HRULE
 S_CCLLOW(dxMax=1.5,pos=LwrHalfStaff) ::S_DIGIT1(pos=LwrHalfStaff) ==> CLEF(isSoprano);

HRULE
 ::S_CCLLOW(dxMax=1.5,pos=LwrHalfStaff) S_DIGIT1(pos=LwrHalfStaff) ==> CLEF(isSoprano);
```

```
HRULE
 S_DBARLINE(dxMax=1) ::PAUSE(pos=OnStaffLine2,durate=4) ==> CLEF(isMezzosoprano);

HRULE
 ::S_DBARLINE(dxMax=0.5) NOTE(pos=OnStaffLine2,durate=4,inAccordo) ==> CLEF(isTenore);

HRULE
 S_DBARLINE(dxMax=1) ::PAUSE(pos=OnSpace4,durate=4) ==> CLEF(isMezzosoprano);

HRULE
 S_DBARLINE(dxMax=1) ::PAUSE(pos=OnStaffLine3,durate=4) ==> CLEF(isContralto);



// REGOLE PER RICOSTRUZIONE NOTE SEMIBREVI
HRULE
  ::S_HOOK1UP(dxMax=0.5,dyMax=0.5,notBeamed,withoutNote)
S_HOOK1UP ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  ::S_PUNTVAL(dxMax=0.5,dyMax=0.5,altezzaGreat>0.9,withoutNote)
S_CHORD2V(altezzaLess<1.5,withoutNote) ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  ::S_PUNTVAL(dxMax=0.5,dyMax=0.5,altezzaGreat>0.9,withoutNote)
S_DIGIT4 ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  ::S_DIGIT4(dxMax=1.5,dyMax=0.5,altezzaGreat>0.6,withoutNote)
S_DIGIT4 ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  ::S_DIGIT6(dxMax=1.5,dyMax=0.5,altezzaGreat>0.6,withoutNote)
S_DIGIT4 ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  S_DIGIT4(dxMax=1.5,dyMax=0.5,altezzaGreat>0.6,withoutNote)
::S_DIGIT4 ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  ::S_DIGIT4(dxMax=1.1,dyMax=0.5,altezzaGreat>0.6,withoutNote)
S_PUNTVAL(altezzaGreat>0.9) ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  ::S_DIGIT4(dxMax=1.1,dyMax=0.5,altezzaGreat>0.6,withoutNote)
S_BEAM4 ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  S_REST16(altezzaLess<1.7,withoutNote)
::S_DIGIT4(dxMax=1,dyMax=0.5,altezzaGreat>0.6) ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  ::S_REST16(altezzaLess<1.7,withoutNote)
S_DIGIT4(dxMax=1,dyMax=0.5,altezzaGreat>0.6) ==> NOTE(stem:=None,head:=Empty,durate:=1);
```

```
HRULE
  ::S_DIGIT4(dxMax=1,dyMax=0.5,altezzaGreat>0.6,withoutNote)
S_BASS(altezzaLess<2) ==> NOTE(stem:=None,head:=Empty,durate:=1);

HRULE
  ::S_PUNTVAL(dxMax=0.5,dyMax=0.5,altezzaGreat>0.9,larghezzaGreat>0.5)
S_PUNTVAL(altezzaGreat>0.9,larghezzaGreat>0.5,withoutNote)
==> NOTE(stem:=None,head:=Empty,durate:=1);


// REGOLE PER RICOSTRUZIONE TEMPO C
HRULE
 S_C_SX(dxMax=1,dyMax=0.2,notBeamed,pos=OnStaffLine3)
::S_C_DX(pos=OnStaffLine3,larghezzaLess<1.5) ==>TIME(durate:=c);


// REGOLE PER RICOSTRUZIONE BEMOLLI
HRULE
  ::S_HOOK3DWN(dxMax=1,dyMax=2) S_CHORD2V ==> ALTERAZIONE(isBemolle);

HRULE
  ::S_HOOK1DWN(dxMax=1,dyMax=2) S_CHORD2V ==> ALTERAZIONE(isBemolle);

HRULE
  S_HOOK3DWN ::S_CHORD2V(dxMax=1) ==> ALTERAZIONE(isBemolle);

HRULE
  S_HOOK1DWN ::S_CHORD2V(dxMax=1) ==> ALTERAZIONE(isBemolle);

// REGOLE RICOSTRUZIONE DINAMICHE PIANO e FORTI
HRULE
  SIMBOLO(dxMax=2,isPiano) ::NOTE ==> NOTE(withPiano);

HRULE
  SIMBOLO(dxMax=2,isForte) ::NOTE ==> NOTE(withForte);
```

Bibliografia

[1] A. Andronico and A. Ciampa. On automatic pattern recognition and acquistion of printed music. In *Proceedings of the International Computer Music Conference*, pages 245–278, Venice, Italy, 1982.

[2] J. V. Mahoney. Automatic analysis of musical score images. BSc thesis, Department of Computer Science and Engineering, MIT, Cambridge, May 1982.

[3] Hirokazu Kato and Seiji Inokuchi. The recogntion system for printed piano music using musical knowledge and constraints. In *Proceedings of the International Association for Pattern Recognition Worskshop on Syntactic and Structural Pattern Recognition*, pages 231–248, Murray Hill, NJ, June 1990.

[4] David Bainbridge. *A Complete Optical Music Recognition System: Looking to the Future*. Dpt of Computer Science, Univ. of Canterbury, GB, November 1994.

[5] David Bainbridge, Tim Bell. *An Extensible Optical Music Recognition System*, in the Australasian Computer Science Conference (Melbourne 1996), pp 308-317.

[6] David Bainbridge, Tim Bell. *Dealing with Superimposed Objects in Optcal Music Recognition*, in Image Processing and its Applications (Dublin 1997), pp 756-760.

[7] N. P. Carter, R. A. Bacon, and T. Messenger. The acquisition, representation and reconstruction of printed music by computer: A survey. *Computers and the Humanities*, 22:117 ff., 1988.

[8] N. P. Carter. *Automatic Recognition of Printed Music in the Context of Electronic Publishing*. PhD thesis, University of Surrey, February 1989.

[9] N. P. Carter and R. A. Bacon. Automatic recognition of music notation. In *Proceedings of the International Association for Pattern Recognition Worskshop on Syntactic and Structural Pattern Recognition*, page 482 ff., Murray Hill, NJ, June 1990.

[10] Dorothea Blostein and Nicholas P. Carter. Recognition of Music Notation. In *SSPR '90 Workink Group Report*, 1990.

[11] Nicholas P. Carter and Richard A. Bacon. Automatic recognition of printed music. Dept. of Physics, University of Surrey, GB. Preprint of an article, 1991.

[12] Nicholas P. Carter. A new edition of walton's façade using automatic score recognition. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition (Proceedings of International Workshop on Structural and Syntactic Pattern Recognition, Bern, CH)*, volume 5 of *Series in Machine Perception and Artificial Intelligence*, pages 352–362. World Scientific, 1992.

[13] Nicholas P. Carter. Segmentation and preliminary recognition of madrigals notated in white mensural notation. *Machine Vision and Applications*, 5(3):223–230, 1992.

[14] Nicholas P. Carter. Music score recognition: Problems and prospects. In *Computing in Musicology*, volume 9, pages 152–158. Center for Computer Assisted Research in the Humanities (CCARH), Stanford, Menlo Park, CA (USA), 1994.

[15] Vladimit Y. Bushel. *Music Score Recognition*, http://kurort.komkon.org/~bushel.

[16] I. Leplumey and J. Camillerapp. Comparison of region labelling for musical scores. In *Proceedings of First International Conference on Document Analysis*, volume 2, pages 674–682, Saint-Malo, France, 1991.

[17] B. Couasnon and J. Camillerapp. *A Way to Separate Knowledge From Program in Structured Document Analisys: Application to Optical Music Recognition.* IRISA / INSA - Département Informatique, F-35043 Rennes Cedex, France, 1995 IEEE.

[18] Martin Roth *An Approch To Recognition Of Printed Music.* Swiss Federal Institute of Technology, Institute for theoretucal computer science, ETH Zurigo, Switzerland, 1994.

[19] D. S. Prerau. *Computer Pattern Recognition of Standard Engraved Music Notation.* PhD thesis, MIT, September 1970.

[20] D. S. Prerau. Computer pattern recognition of printed music. In *Proceedings of the Fall Joint Computer Conference*, Montvale, NJ 39, November 1971. AFIPS Press.

[21] D. S. Prerau. Do-re-mi: A program that recognizes music notation. *Computer and the Humanites*, 9(1):25–29, January 1975.

[22] Tojo, A., Aoyama, H. *Automatic Recognition of Music Score*, Proceedings, 6th International Conference on Pattern Recognition, Munich, Germany, p.1123 (1982)

[23] Martin, P. and Bellisant, 1991, Proc. Of First International Conf. on Document Analysis, 1 (Saint Malo, France) 417-425.

[24] T. Kobayakawa. *Auto Music Score Recognizing System*, In Donald P. D'Amato, editor, Proceedings SPIE: Character Recognition Technologies, volume 1906, May 1993.

[25] Kia C. Ng, Roger D. Boyle, David Cooper. *Automated Optical Music Score Recognition and its Enhancement using High-level Musical Knowledge*, Proceedings of the XI Colloquium on Musical Information, Università degli studi di Bologna, Dipartimento di Musica e Spettacolo (1995).

[26] Eleanor Selfridge-Field. *How Practical is Optical Music Recognition as an Input Method*, Computing in Musicology 9 (1993-94), pp 159-166.

[27] Eleanor Selfridge-Field. *Optical Recognition of Music Notation: A Survey of Current Work*, Computing in Musicology 9 (1993-94), pp 109-145.

[28] Eleanor Selfridge-Field. *Beyond MIDI - The Handbook of Musical Codes.*, Eleanor Selfridge-Field (Ed.). London, UK: The MIT Press, 1997.

[29] Ichiro Fujinaga. *Adaptive Optical Music Recognition*, Ph.D Dissertion. McGill University, Montreal, CA, 1997.

[30] Ichiro Fujinaga. Optical music recognition using projections. Master's thesis, McGill University, Montreal, CA, 1988.

[31] Ichiro Fujinaga, Bo Alphonce, and Bruce Pennycook. Issues in the design of an optical music recognition system. In *Proceedings of the International Computer Music Conference*, pages 113–116, Ohio State University, November 1989.

[32] A. K. O. Choi, K. P. Chan. *Automatic Optical Music Recognition.* Final Year Project Report, Dept. of Computer Science, Hong Kong University.

[33] Takebumi Itagaki, Shuji Hashimoto, Masayuki Isogai, and Sadamu Ohteru. Automatic recognition on some different types of musical notation. In *Proceedings of the International Association for Pattern Recognition Worskshop on Syntactic and Structural Pattern Recognition*, page 488 ff., Murray Hill, NJ, June 1990.

[34] W. F. McGee and P. Merkley. *The optical sanning of medieval music*, Computers and the Humanities, 25(1): 47-53, 1991.

[35] William McGee. *MusicReader: An Interactive Optical Music Recognition System*, Computing in Musicology 9 (1993-94), pp 146-151.

[36] Notescan. http://musicwareinc.com/prod14.htm.

[37] Yoichi Fujimoto et al. The keyboard playing robot WABOT-2. *Bulletin of Science and Engineering Research Laboratory*, 112, 1985.

[38] T. Matsushima, T. Harada, I. Sonomoto, K.Kanamori, A. Uesugi, Y. Nimura, S. Hashimoto, and S. Ohteru. Automated recognition system for musical score - the vision system of WABOT-2. *Bulletin of Science and Engineering Research Laboratory, Waseda University*, 112:25–52, September 1985.

[39] Toshiaki Matsushima. Automated high speed recognition of printed music (WABOT-2 vision system). In *Proceedings of the 1985 International Conference on Advances Robotics*, page 477 ff., Japan Industrial Robot Association (JIRA), Shiba Koen Minato-ku, Tokyo, 1985.

[40] J. Anstice, T. Bell, A. Cockburn and M. Setchell. The Design of a Pen-Based Musical Input System. OzCHI'96: The Sixth Australian Conference on Computer-Human Interaction. Hamilton, New Zealand. 24-27 November, 1996. pages 260-267. IEEE Press.

[41] E. Ng , T.Bell , A. Cockburn. Improvements to a Pen-Based Musical Input System. OzCHI'98: The Australian Conference on Computer-Human Interaction. Adelaide, Australia. 29 November to 4 December, 1998. pages 239–252. IEEE Press.

[42] J. W. Roach and J. E. Tatem. Using domain knowledge in low-level visual processing to interpret handwritten music: an experiment. *Pattern Recognition*, 21(1):33–44, 1988.

[43] A. T. Clarke, B. M. Brown, and M. P. Thorne. Inexpensive optical character recogniton of music notation: A new alternative for publishers. In *Proceedings of the Computers in Music Research Conference*, page 84 ff., Bailrigg, Lancaster, April 1988.

[44] Bharath R. Modayur, Visvanathan Ramesh, Robert M. Haralick, and Linda G. Shapiro. Muser - a prototype musical score recognition system using mathematical morphology. Intelligent Systems Laboratory, EE Dept, FT-10, University of Washington, Seattle WA 98195, June 1992.

[45] H.Miyao, T.Ejima, M.Miyahara, and K.Kotani. *Symbol Recognition for Printed Piano Scores Based on the Musical Knowledge*, Trans. IEICE(D-II), Vol.J75-D-II, No.11, pp.1848-1855 (1992).

[46] H. Fahmy, B. Blostein. *The Graph Rewriting Paradigm for Discrete Relaxation: Applications to Sheet Music Recognition*, International Journal of Pattern Recognition and Artificial Intelligence Vol 12 n.6, pp. 763-799, Sept. 1998

[47] Bharath R. Modayur. Music Score Recognition - a selective attention approach using mathematical morphology. Intelligent Systems Laboratory, EE Dept, FT-10, University of Washington, Seattle WA 98195, March 1996.

[48] Martin Roth. *Review of Midiscan*. http://www.cs.waikato.ac.nz/ davidb/omr/roth_ms_review.html.

[49] "NIFF 6a: Notation Interchange File Format". tech. rep., NIFF Consortium, July 1995.

[50] "Standard Music Description Language (SMDL)". ISO/IEC DIS 10743.

[51] P. Nesi, N. Baldini, P. Bellini, F. Bennati, F. De Meo, A. Giotti, S. Macchi, and L. Mengoni. *Analisi ad Oggetti della Musica*. tech. rep., Dipartimento di Sistemi e Informatica, Facoltà di Ingegneria, Università di Firenze, RT 26/95, Florence, Italy, 1995.

[52] P. Nesi, N. Baldini, P. Bellini, F. Bennati, F. De Meo, A. Giotti, S. Macchi, and L. Mengoni. *LIOO: Leggio Interativo Object-Oriented, Manuale Utente*. tech. rep., Dipartimento di Sistemi e Informatica, Facoltà di Ingegneria, Università di Firenze, RT 27/95, Florence, Italy, 1995.

[53] Rumelhart, D. E., and J. L. McClelland. Parallel Distributed Processing: Exploration in the Microstrure of Cognition, Vol. 1 Cambridge University Press, 1986.

[54] L. Smith, SCORE. Beyond MIDI - The Handbook of Musical Codes, (E. Selfridge-Field, ed.), The MIT Press, London, pages 252-282.

[55] D. Taupin, R. Mitchell, A. Egler. Using TEX to Write Polyphonic or Instrumental Music ver T.77, http:\\hprib.lps.u-psud.fr, 1997.

[56] I. Icking. MuTEX, MusicTEX, and MusiXTEX Beyond MIDI - The Handbook of Musical Codes, (E. Selfridge-Field, ed.), The MIT Press, London, pages 222-231,1997.

[57] P. Bellini, P. Nesi. WEDELMUSIC FORMAT: An XML Music Notation Format for Emerging Applications. Proceedings of the 1st International Conference of Web Delivering of Music. 23-24 November, Florence, Italy, pages.79-86, IEEE press (2001).

[58] P. Bellini, F. Fioravanti and P. Nesi. Managining Music in Orchestras. IEEE COmputer, pages.26-34, September, 1999.

[59] M. Good. MusicXML for Notation and Analysis. In W. B. Hewlett and E. Selfridge-Field (Eds.), The Virtual Score Representation, Retrieval, Restoration (pp.113-124). Cambridge, MT: The MIT Press, 2001.

[60] SMDL ISO/IEC. Standard Music Description Language. ISO/IEC DIS 10743, 1995.

[61] N. Luth. *Automatic Identification of Music Notation*, Proceedings of the 2nd International Conference of Web Delivering of Music. Darmstadt (Germany): IEEE press, 2002.