3D Computer Vision

Course for PhD program in Information Engineering

28/02, 02/03, 07/03 2023

Marco Fanfani, PhD

Course Outline

- Full course 12 hours
  - 28/02/2023
  - 02/03/2023
  - 07/03/2023
- 3 CFUs
- Topics:
  - Introduction to geometrical computer vision
  - 3D Reconstruction:
    - Stereo, Structure from Motion, Multi-view Stereo, Structured Light, DEM Modelling, Shape from Shading, Photometric Stereo
  - Visual odometry, SLAM, and Localization

References

- Books
  - Hartley, R., Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*
  - Szeliski, R. (2011). *Computer vision algorithms and applications*
  - Fusiello, A. (2018). *Visione Computazionale. Tecniche di Ricostruzione Tridimensionale*

- Other courses
  - Course by Prof. C. Colombo at the Artificial Intelligence Master Degree at the University of Florence (link)
  - Course by Prof. A. Geiger at the Tuebingen University (link)
  - Several other courses on Computer Vision can be found online (e.g., Coursera, YouTube, etc.)

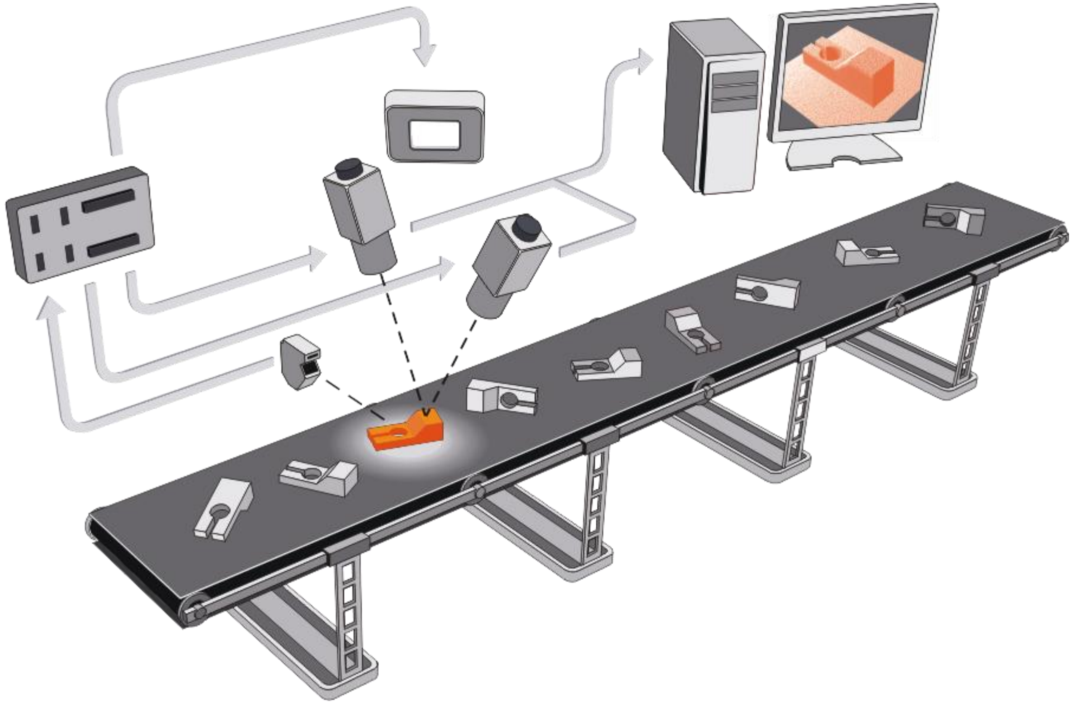
- Coding
  - **Python**: OpenCV, Numpy, Pillow, Scikit-Image, SciPy, Open3D, …
  - **Matlab**: CV Toolbox, Kovesy functions (https://www.peterkovesi.com/matlabfns/), Zisserman functions (https://www.robots.ox.ac.uk/~vgg/hzbook/code/)
  - **Deep-learning**: Tensorflow, Keras, PyTorch, …

Applications

- Cultural Heritage

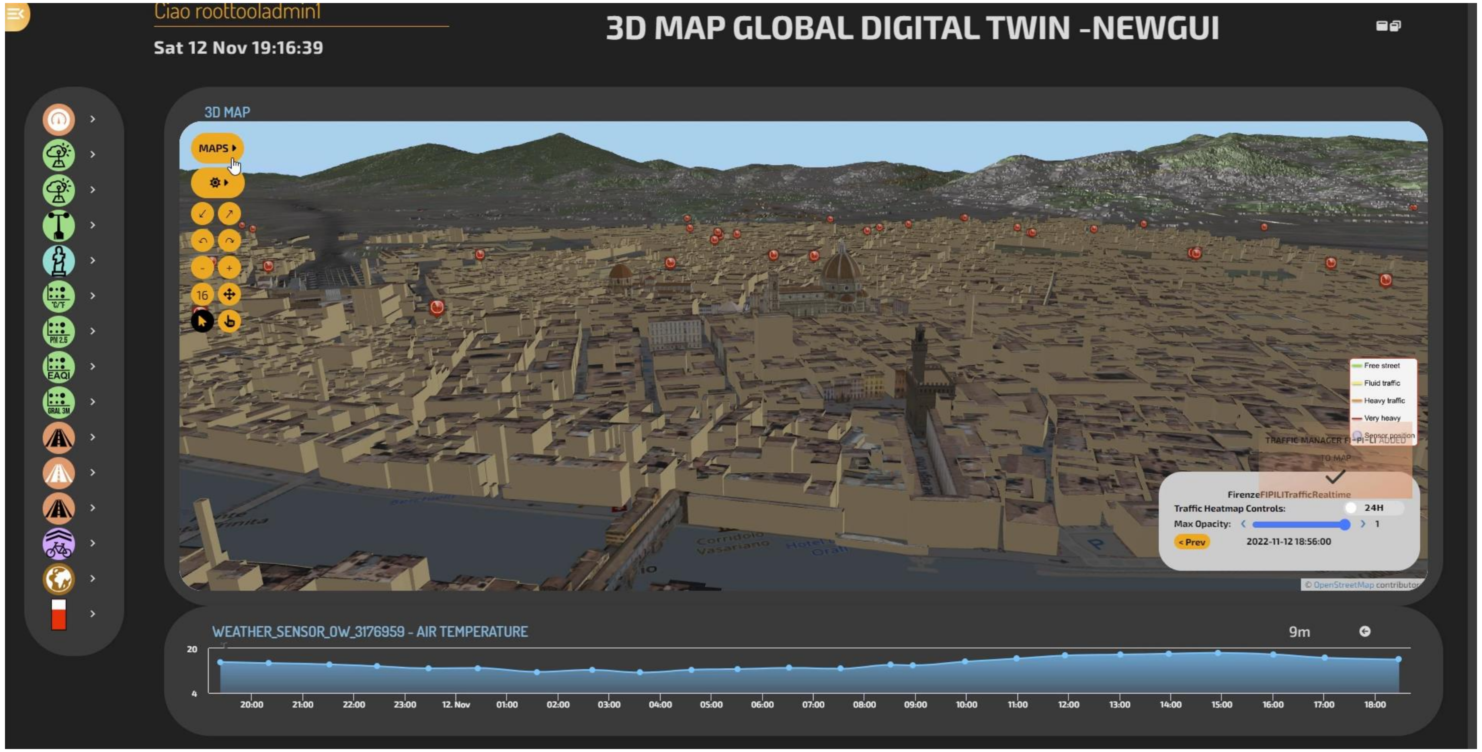
Applications

- Industrial application



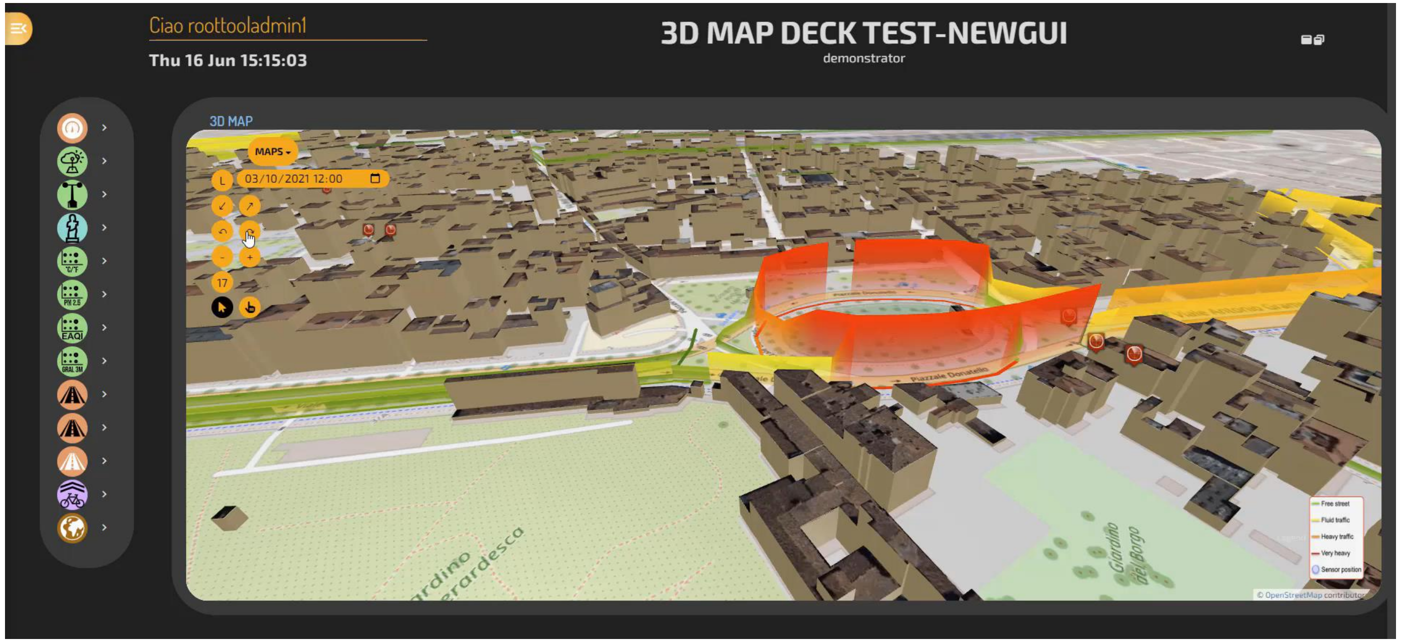
Applications

- City digital twin modelling



Applications

- City Digital Twin modelling
  - Traffic congestion visualization
  - Dispersion of pollutant
  - Urban planning
  - Areas for solar panel installation
  - …

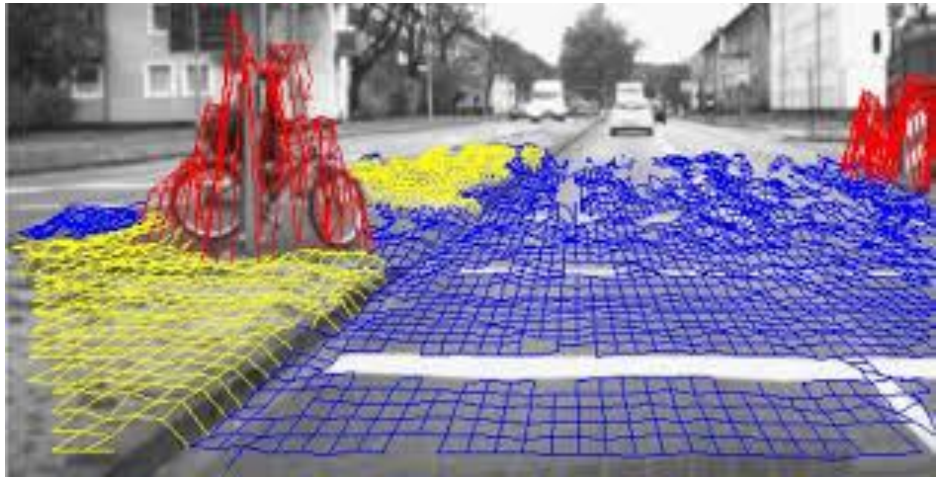


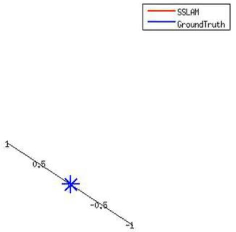
Applications

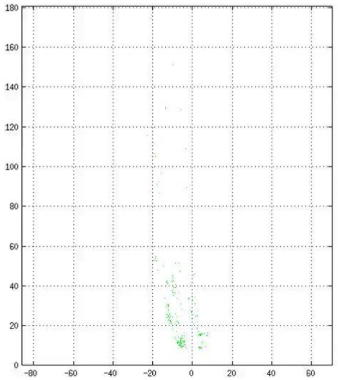
- Virtual/Augmented Reality
  - Structure inspection
  - 3D model visualization
  - Urban planning
  - Realistic environment for simulations

Applications

- Autonomous driving
  - Vehicle odometry
  - Environment perception
  - Obstacle detection

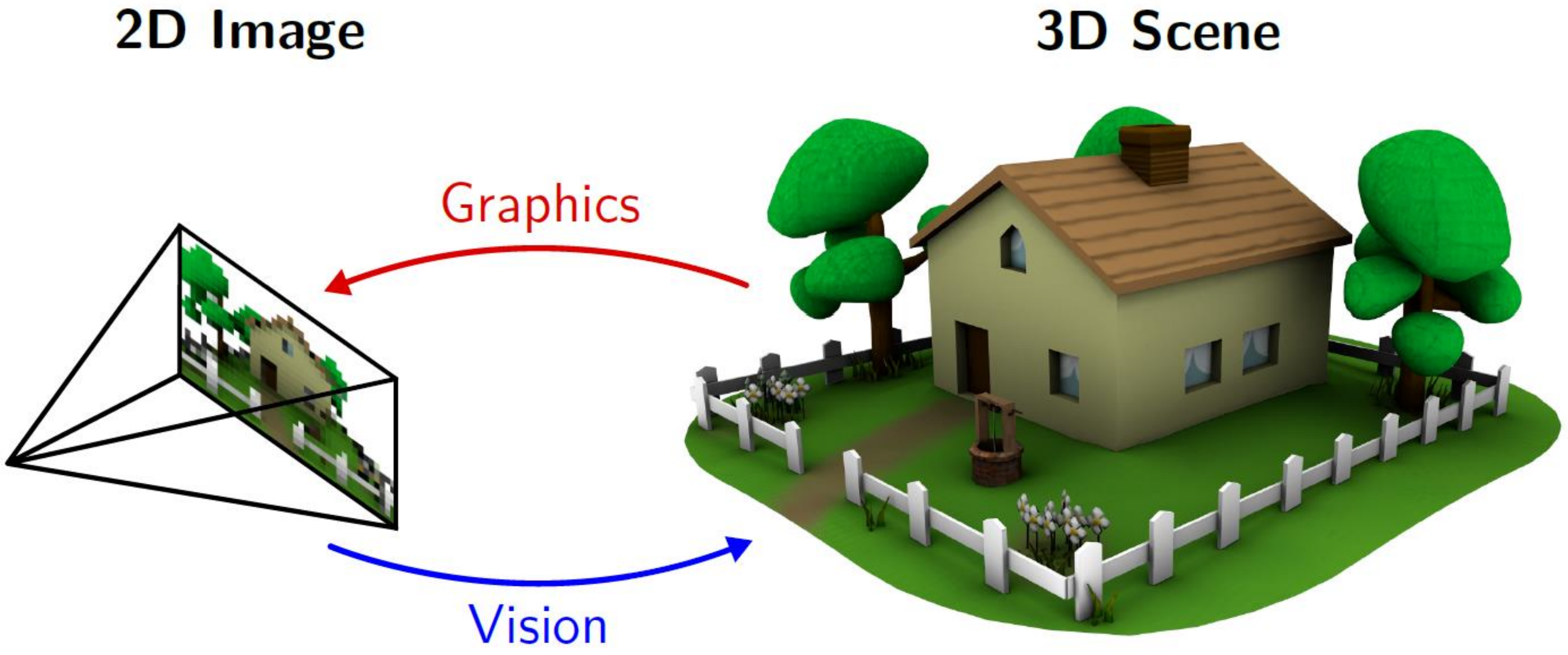






Introduction to Geometrical Computer Vision

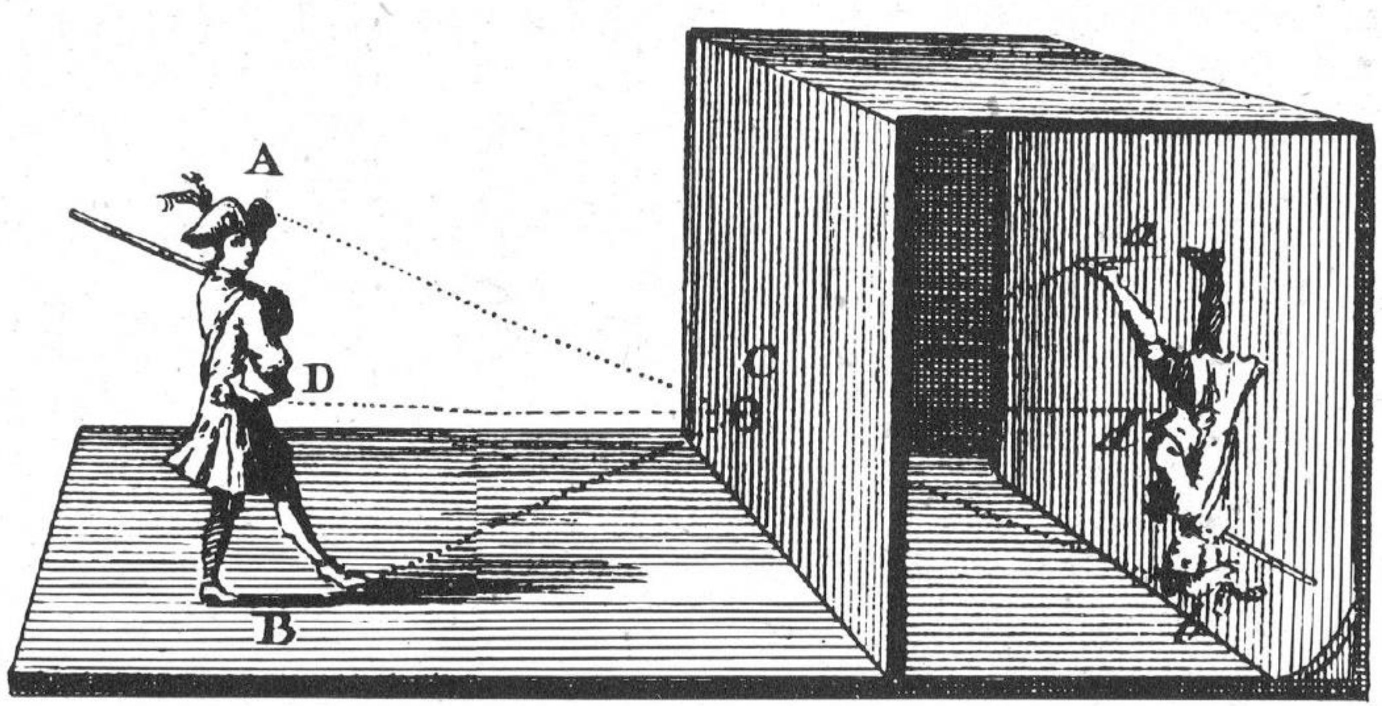
Image projection



Slide from A. Geiger

Camera Obscura

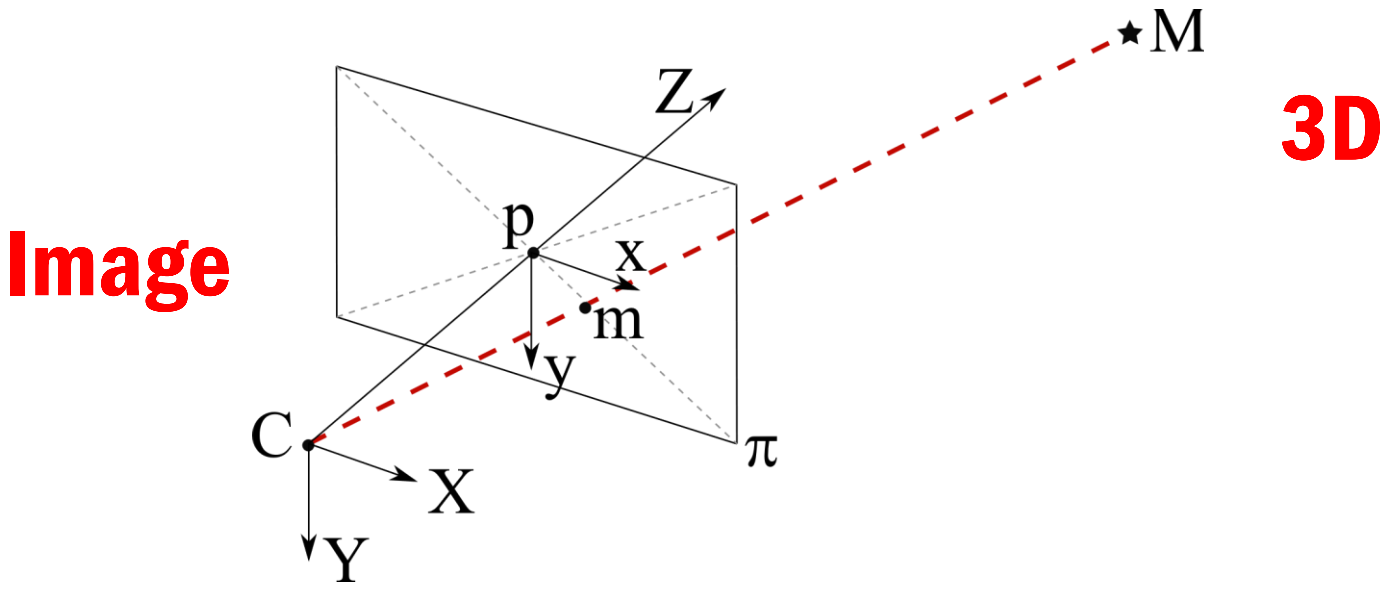
3D



Image

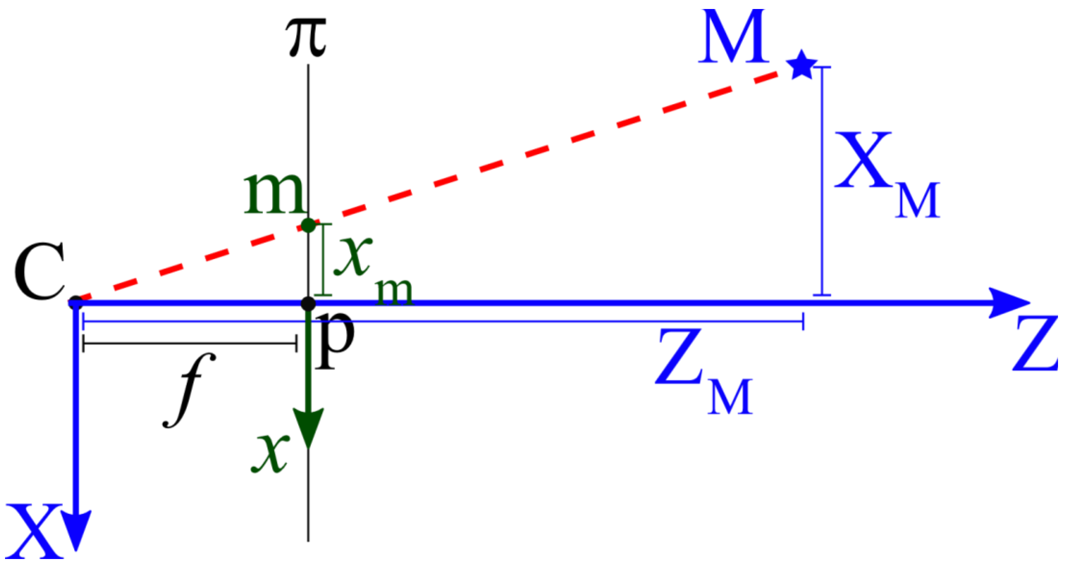
- A darkened room with a **small hole or lens** at one side through which an **image is projected** onto a wall or table opposite the hole.
- The concept was developed further into the **photographic camera** in the first half of the 19th century, when camera obscura boxes were used to expose **light-sensitive materials** to the projected image.

Pinhole camera model

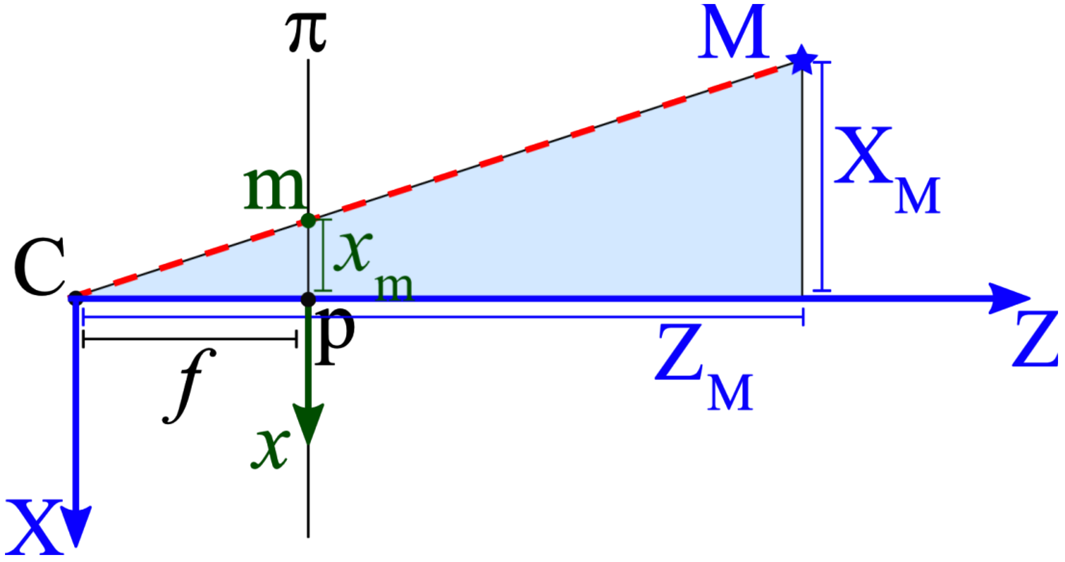


- It models the projection of a **3D point** (M) to an **image point** (m) constrained by the respective position of the **camera center** (C) and the **image plane** π
- The projection is a function $f: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ (loss of information)
- p is the **image principal point**, the projection of C onto π

Camera projection

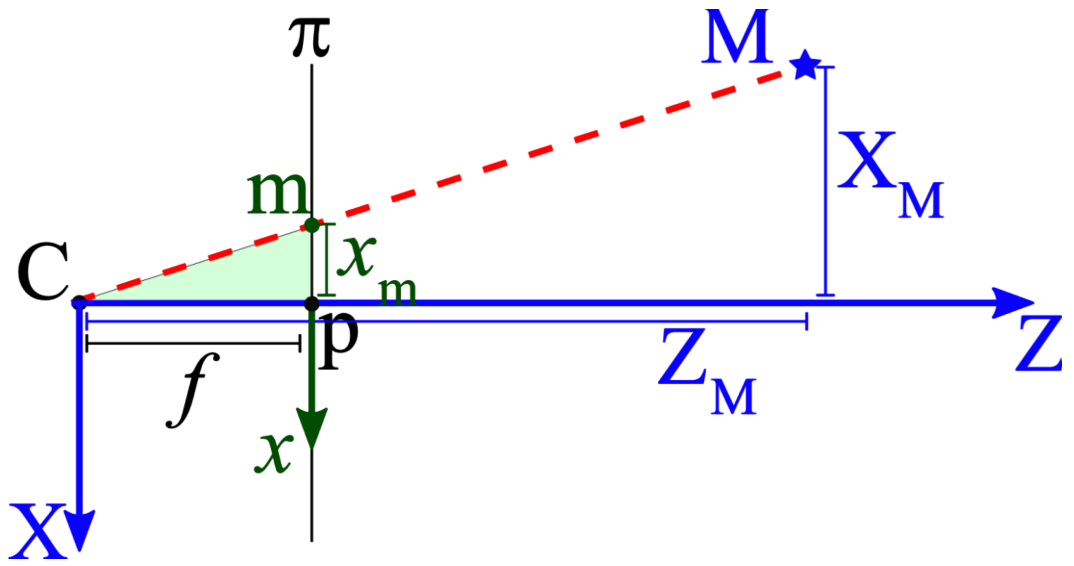


Camera projection



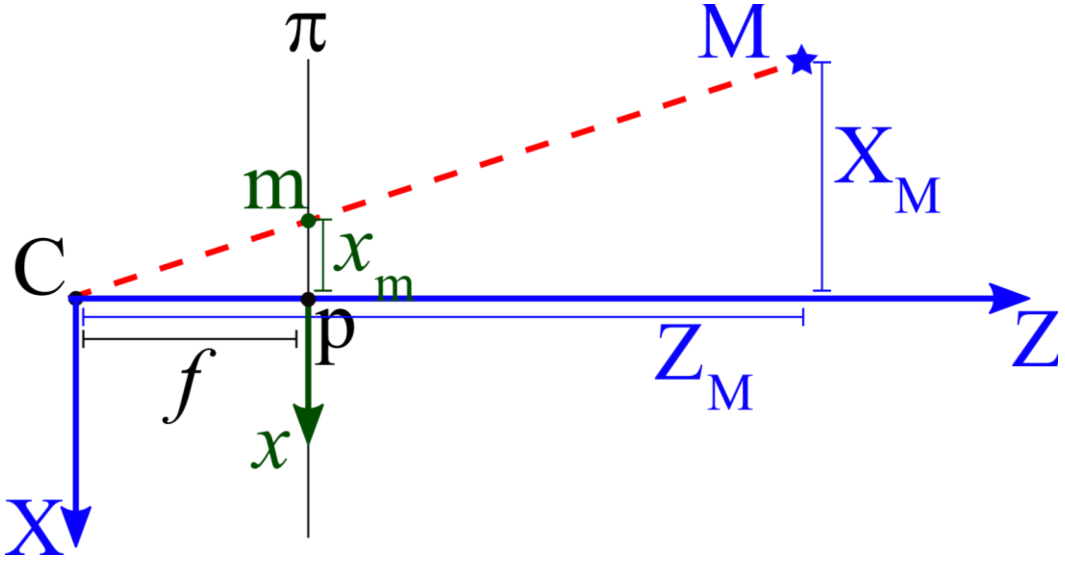
- To obtain the projection m of the 3D point M we can exploit the relation between the similar **blue** and **green** triangle

Camera projection



- To obtain the projection m of the 3D point M we can exploit the relation between the similar **blue** and **green** triangle

Camera projection



- To obtain the projection m of the 3D point M we can exploit the relation between the similar **blue** and **green** triangle

$$\frac{x_m}{f} = \frac{X_M}{Z_M} \quad \Rightarrow \quad x_m = f\,\frac{X_M}{Z_M}$$

Homogeneous coordinate

- A 2D or 3D point can be expressed in **inhomogeous coordinates**

$$\mathbf{m} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

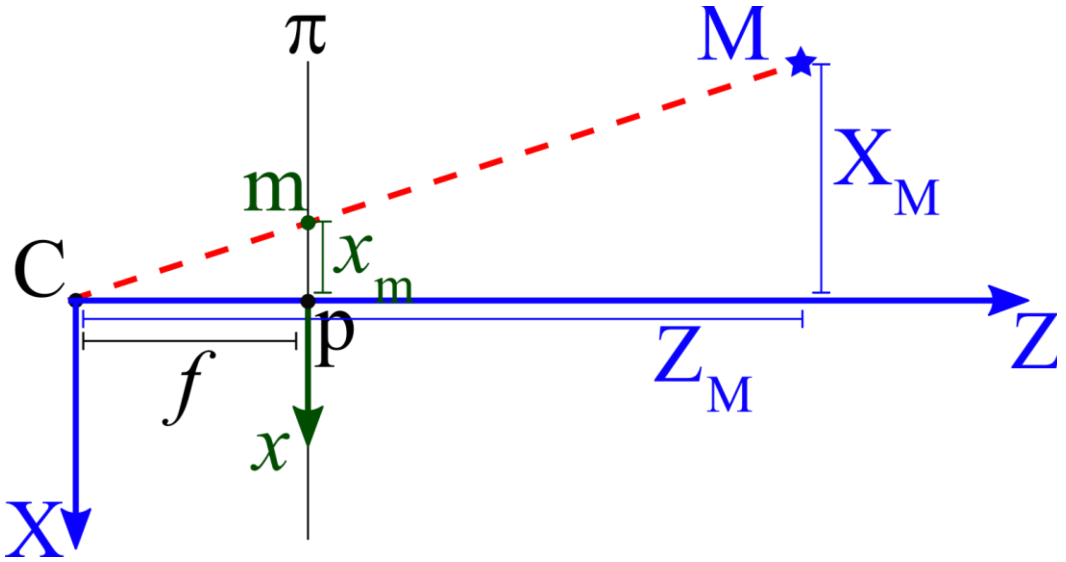
or in **homogeneous coordinates**

$$\bar{\mathbf{m}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \bar{\mathbf{M}} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Note that, in homogeneous coordinate, two points are the same if the are equal **up to a scale factor**

$$\bar{\mathbf{m}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{\tilde{w}} \tilde{\mathbf{m}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} = \begin{bmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \\ 1 \end{bmatrix}$$

Camera projection



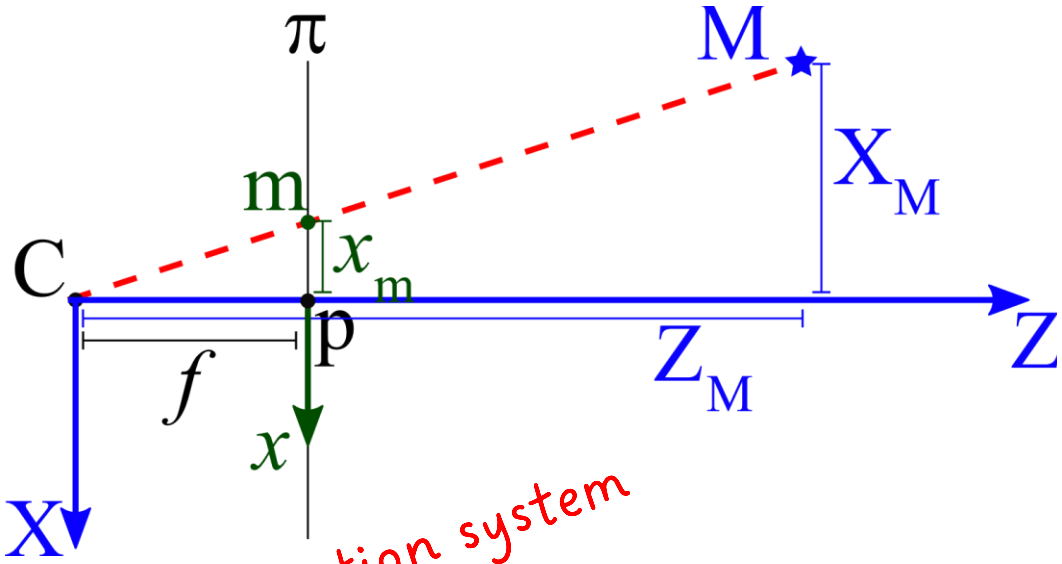
$$\frac{x_m}{f} = \frac{X_M}{Z_M} \quad \Rightarrow \quad x_m = f\,\frac{X_M}{Z_M}$$

$$\frac{y_m}{f} = \frac{Y_M}{Z_M} \quad \Rightarrow \quad y_m = f\,\frac{Y_M}{Z_M}$$

$$\mathbf{m} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix} = \begin{bmatrix} fX_M \\ fX_M \\ Z_M \end{bmatrix} = \begin{bmatrix} fX_M/Z_M \\ fX_M/Z_M \\ 1 \end{bmatrix} = \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix}$$

- f is the **focal length** and is expressed in **pixels** (px)

Projection function



$$\frac{x_m}{f} = \frac{X_M}{Z_M} \Rightarrow x_m = f\,\frac{X_M}{Z_M}$$

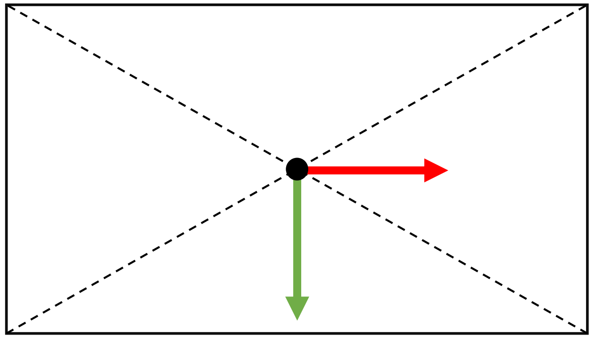
$$\frac{y_m}{f} = \frac{Y_M}{Z_M} \Rightarrow y_m = f\,\frac{Y_M}{Z_M}$$

Linear equation system

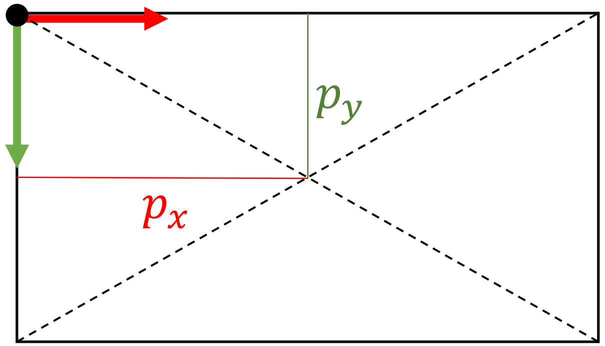
$$\mathbf{m} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix} = \begin{bmatrix} fX_M \\ fX_M \\ Z_M \end{bmatrix} = \begin{bmatrix} fX_M/Z_M \\ fX_M/Z_M \\ 1 \end{bmatrix} = \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix}$$

- f is the **focal length** and is expressed in **pixels** (px)

Camera projection



$$\mathbf{m} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{X_M} \\ \mathrm{Y_M} \\ \mathrm{Z_M} \\ 1 \end{bmatrix} = \begin{bmatrix} f\mathrm{X_M}/\mathrm{Z_M} \\ f\mathrm{Y_M}/\mathrm{Z_M} \\ 1 \end{bmatrix} = \begin{bmatrix} x_\mathrm{m} \\ y_\mathrm{m} \\ 1 \end{bmatrix}$$



$$\mathbf{m} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{X_M} \\ \mathrm{Y_M} \\ \mathrm{Z_M} \\ 1 \end{bmatrix} = \begin{bmatrix} (f\mathrm{X_M} - \mathrm{Z_M}p_x)/\mathrm{Z_M} \\ (f\mathrm{Y_M} - \mathrm{Z_M}p_y)/\mathrm{Z_M} \\ 1 \end{bmatrix} = \begin{bmatrix} x_\mathrm{m} \\ y_\mathrm{m} \\ 1 \end{bmatrix}$$

- $(p_x, p_y)^\top$ are the coordinates of the **image principal point**

Camera matrix

$$\mathbf{m} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix} \Rightarrow \mathbf{m} = \begin{bmatrix} f & \sigma & p_x & 0 \\ 0 & \delta f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix}$$

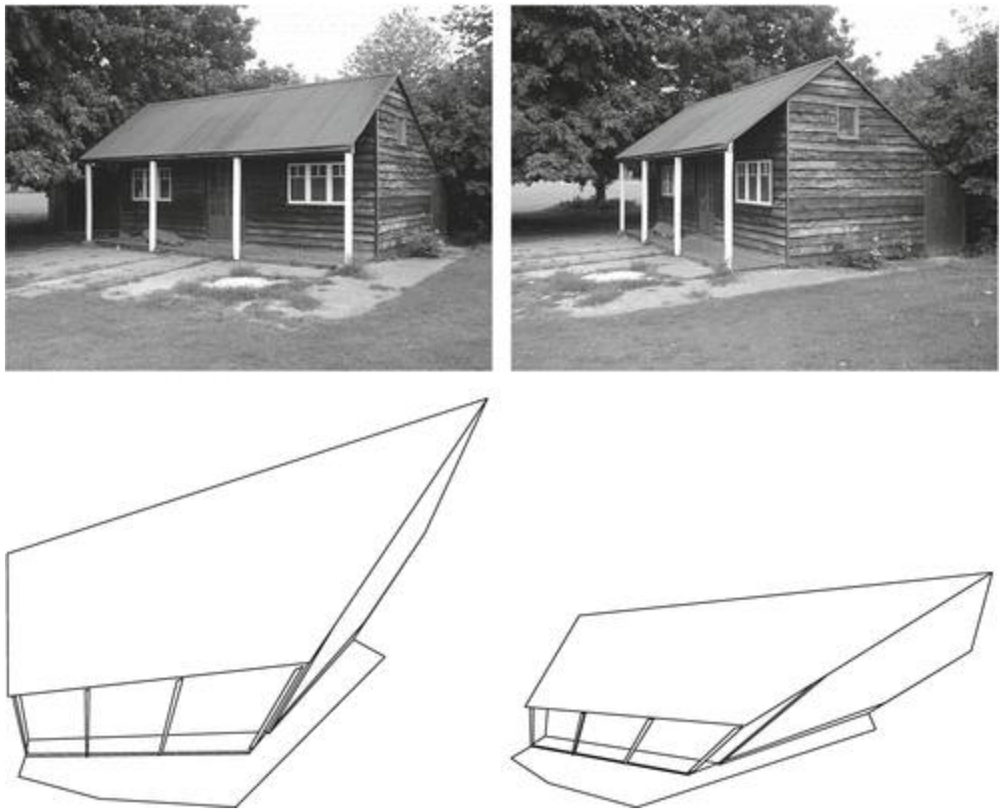
The first 3x3 submatrix is the **calibration** or intrinsic (K) matrix

- $(p_x, p_y)^T$ are the coordinates of the **image principal point**
- f is the **focal length**
- δ is the **aspect ratio** between the x and y axis (non-square pixels)
- σ is the **skew**, $\sigma \neq 0$ if the x and y axis are not perpendicular

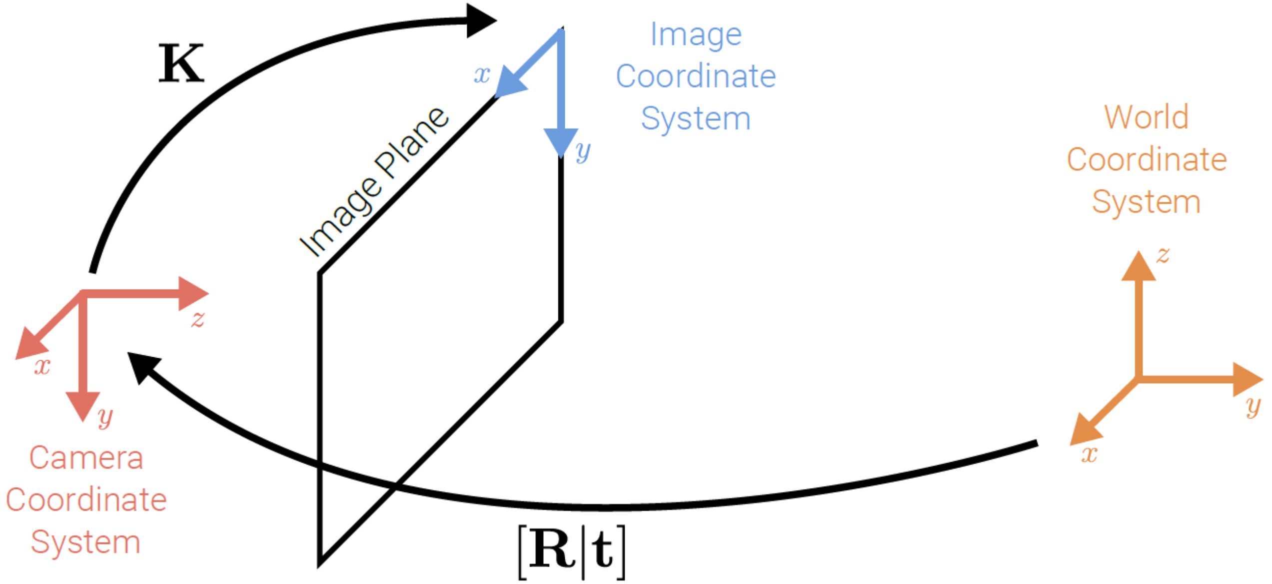
⇒ In modern cameras, we can safely assume $\delta = 1$ and $\sigma = 0$

Projective reconstruction

- Without K we can obtain a 3D reconstruction affected by a **projective transformation**



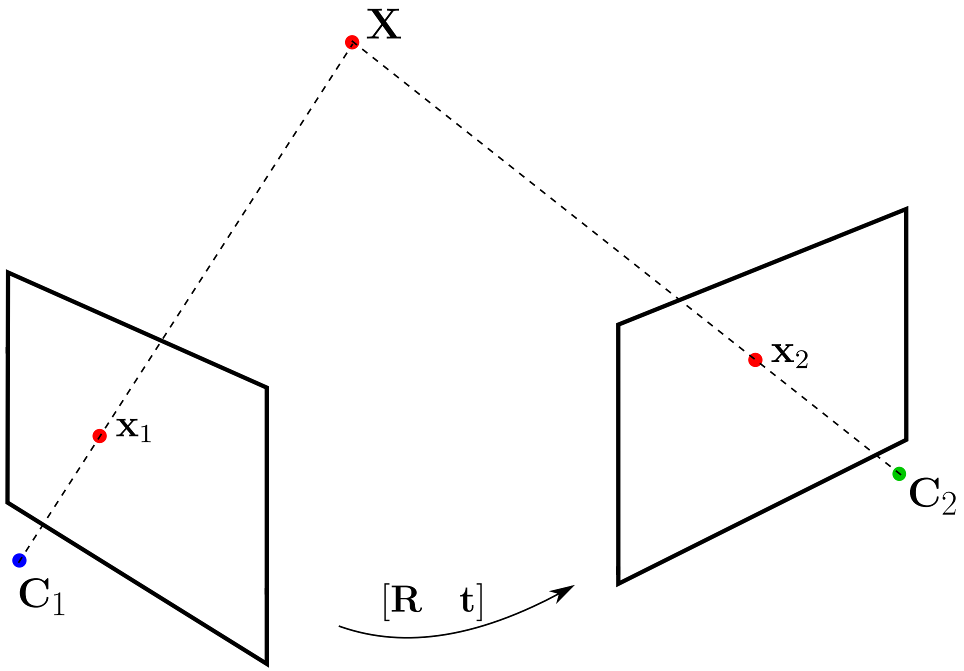
Full camera matrix



- If $W_{cs} \not\equiv C_{cs}$ we have to take into account a **3D rigid transformation**: Rotation R + Translation **t**

$$\mathbf{m} = [\mathbf{K} \quad \mathbf{0}]\mathbf{M}_C = [\mathbf{K} \quad \mathbf{0}]\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}\mathbf{M}_W = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{M}_W = \mathbf{P}\mathbf{M}_W$$

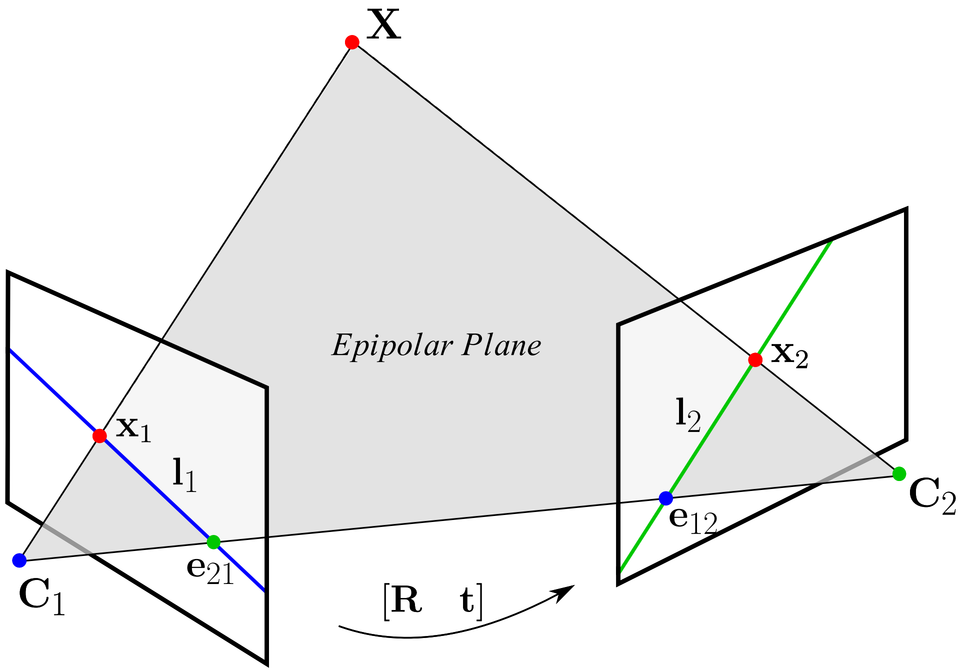
Epipolar geometry



- Two cameras:
  - $P_1 = K_1[I \quad \mathbf{0}]$ such that $\mathbf{x}_1 = P_1\mathbf{X}$
  - $P_2 = K_2[R \quad \mathbf{t}]$ such that $\mathbf{x}_2 = P_2\mathbf{X}$

- Note: $W_{cs} \equiv C_{cs}^1$

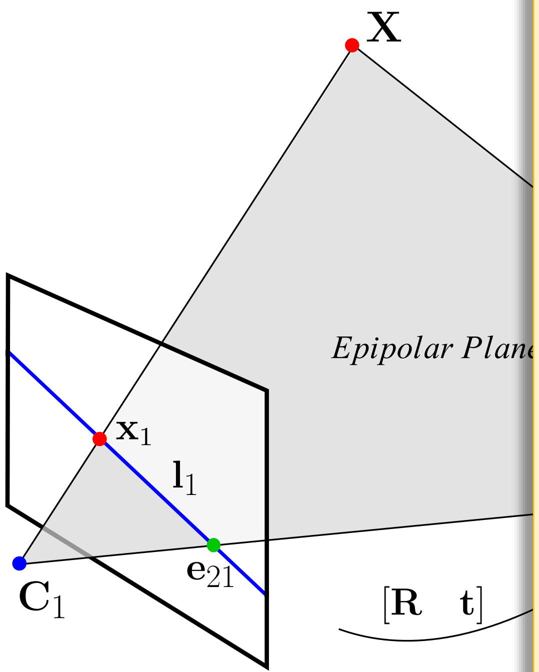
Epipolar geometry



where

- $\mathbf{e}_{21}$ is the projection of $\mathbf{C}_2$ onto P_1
- $\mathbf{e}_{12}$ is the projection of $\mathbf{C}_1$ onto P_2
- $\mathbf{e}_{21}$ and $\mathbf{e}_{12}$ are the **epipoles**
- $\mathbf{l}_1$ and $\mathbf{l}_2$ are the **epipolar lines**

Epipolar geometry



$$\mathbf{x}_1 = \mathrm{K}_1[\mathrm{I} \quad \mathbf{0}]\mathbf{X} \Leftrightarrow \mathbf{x}_1 = \mathrm{K}_1\breve{\mathbf{X}} \Leftrightarrow \breve{\mathbf{X}} = \mathrm{K}_1^{-1}\mathbf{x}_1$$

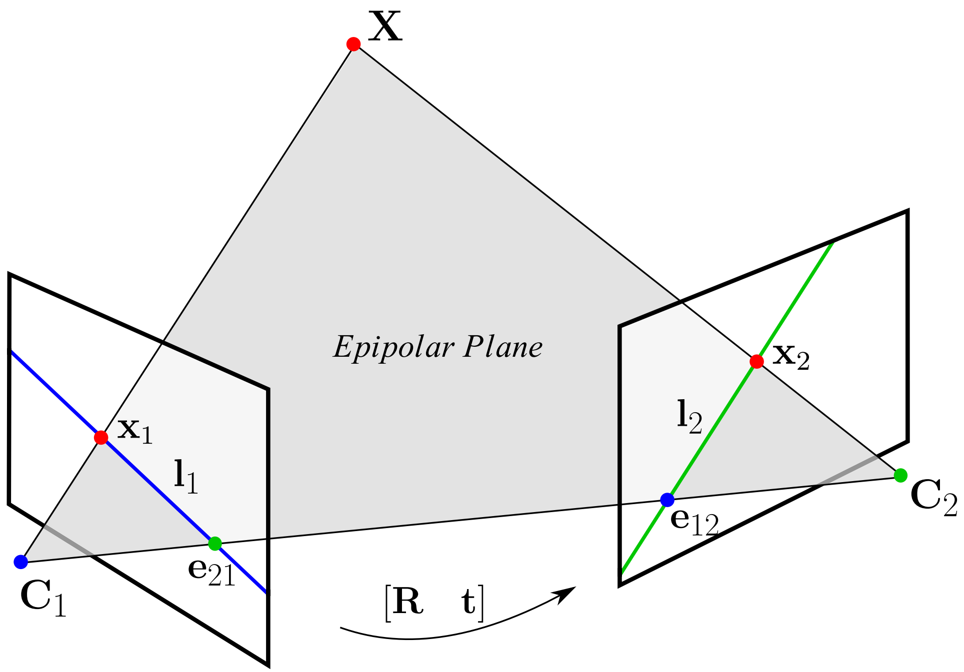
and

$$\mathbf{x}_2 = \mathrm{K}_2[\mathrm{R} \quad \mathbf{t}]\mathbf{X} \Leftrightarrow \mathbf{x}_2 = \mathrm{K}_2(\mathrm{R}\breve{\mathbf{X}} + \mathbf{t})$$

where

- $\mathbf{e}_{21}$ is the projection of $\mathbf{C}_2$ onto P_1
- $\mathbf{e}_{12}$ is the projection of $\mathbf{C}_1$ onto P_2
- $\mathbf{e}_{21}$ and $\mathbf{e}_{12}$ are the **epipoles**
- $\mathbf{l}_1$ and $\mathbf{l}_2$ are the **epipolar lines**

Epipolar geometry



where

- $\mathbf{e}_{21}$ is the projection of $\mathbf{C}_2$ onto P_1
- $\mathbf{e}_{12}$ is the projection of $\mathbf{C}_1$ onto P_2
- $\mathbf{e}_{21}$ and $\mathbf{e}_{12}$ are the **epipoles**
- $\mathbf{l}_1$ and $\mathbf{l}_2$ are the **epipolar lines**

$$\mathbf{x}_1 = K_1[I \quad \mathbf{0}]\mathbf{X} \Leftrightarrow \mathbf{x}_1 = K_1\breve{\mathbf{X}} \Leftrightarrow \breve{\mathbf{X}} = K_1^{-1}\mathbf{x}_1$$

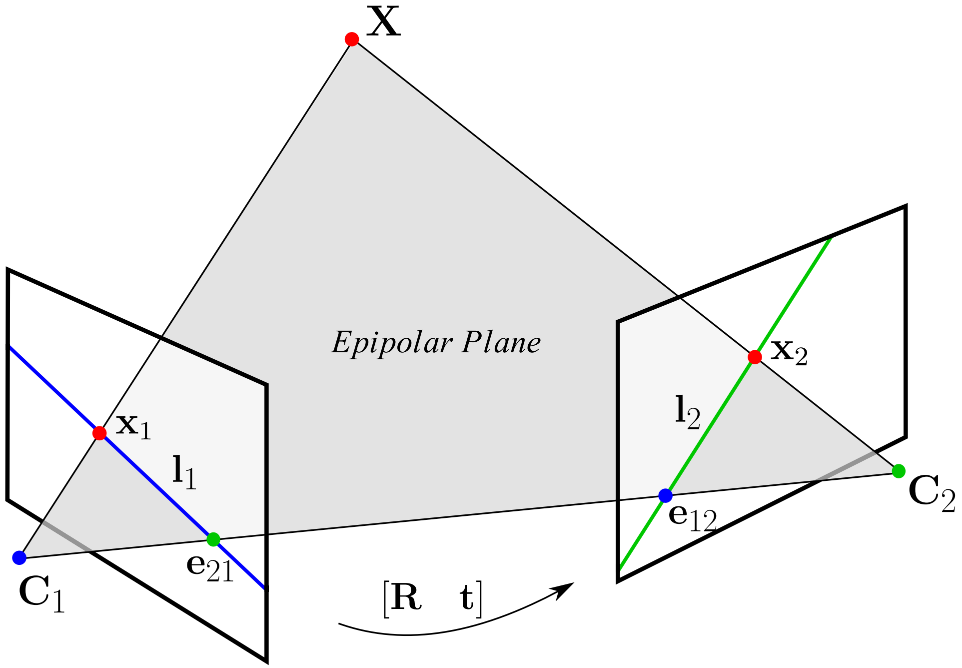
and

$$\mathbf{x}_2 = K_2[R \quad \mathbf{t}]\mathbf{X} \Leftrightarrow \mathbf{x}_2 = K_2(R\breve{\mathbf{X}} + \mathbf{t})$$

by substituting $\breve{\mathbf{X}}$ we get

$$\mathbf{x}_2 = K_2(RK_1^{-1}\mathbf{x}_1 + \mathbf{t}) \Leftrightarrow K_2^{-1}\mathbf{x}_2 = RK_1^{-1}\mathbf{x}_1 + \mathbf{t}$$

Epipolar geometry



where

- $\mathbf{e}_{21}$ is the projection of $\mathbf{C}_2$ onto P_1
- $\mathbf{e}_{12}$ is the projection of $\mathbf{C}_1$ onto P_2
- $\mathbf{e}_{21}$ and $\mathbf{e}_{12}$ are the **epipoles**
- $\mathbf{l}_1$ and $\mathbf{l}_2$ are the **epipolar lines**

$$\mathbf{x}_1 = \mathrm{K}_1[\mathrm{I} \quad \mathbf{0}]\mathbf{X} \Leftrightarrow \mathbf{x}_1 = \mathrm{K}_1\breve{\mathbf{X}} \Leftrightarrow \breve{\mathbf{X}} = \mathrm{K}_1^{-1}\mathbf{x}_1$$

and

$$\mathbf{x}_2 = \mathrm{K}_2[\mathrm{R} \quad \mathbf{t}]\mathbf{X} \Leftrightarrow \mathbf{x}_2 = \mathrm{K}_2\left(\mathrm{R}\breve{\mathbf{X}} + \mathbf{t}\right)$$

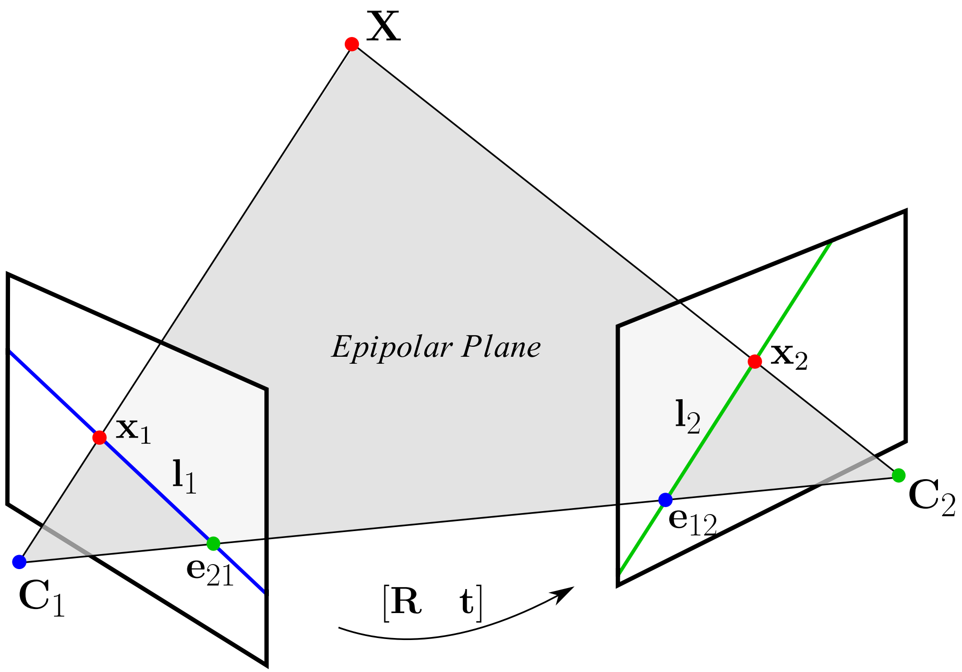
by substituting $\breve{\mathbf{X}}$ we get

$$\mathbf{x}_2 = \mathrm{K}_2(\mathrm{R}\mathrm{K}_1^{-1}\mathbf{x}_1 + \mathbf{t}) \Leftrightarrow \mathrm{K}_2^{-1}\mathbf{x}_2 = \mathrm{R}\mathrm{K}_1^{-1}\mathbf{x}_1 + \mathbf{t} \Leftrightarrow$$

$$\mathbf{t} \times (\mathrm{K}_2^{-1}\mathbf{x}_2) = \mathbf{t} \times \mathrm{R}\mathrm{K}_1^{-1}\mathbf{x}_1 \Leftrightarrow$$

$$(\mathrm{K}_2^{-1}\mathbf{x}_2)^{\top}\,[\mathbf{t} \times (\mathrm{K}_2^{-1}\mathbf{x}_2)] = (\mathrm{K}_2^{-1}\mathbf{x}_2)^{\top}\,[\mathbf{t} \times \mathrm{R}]\,(\mathrm{K}_1^{-1}\mathbf{x}_1) = 0$$

Epipolar geometry



where

- $\mathbf{e}_{21}$ is the projection of $\mathbf{C}_2$ onto P_1
- $\mathbf{e}_{12}$ is the projection of $\mathbf{C}_1$ onto P_2
- $\mathbf{e}_{21}$ and $\mathbf{e}_{12}$ are the **epipoles**
- $\mathbf{l}_1$ and $\mathbf{l}_2$ are the **epipolar lines**

$$\mathbf{x}_1 = K_1[I \quad \mathbf{0}]\mathbf{X} \Leftrightarrow \mathbf{x}_1 = K_1\breve{\mathbf{X}} \Leftrightarrow \breve{\mathbf{X}} = K_1^{-1}\mathbf{x}_1$$

and

$$\mathbf{x}_2 = K_2[R \quad \mathbf{t}]\mathbf{X} \Leftrightarrow \mathbf{x}_2 = K_2\left(R\breve{\mathbf{X}} + \mathbf{t}\right)$$

by substituting $\breve{\mathbf{X}}$ we get

$$\mathbf{x}_2 = K_2(RK_1^{-1}\mathbf{x}_1 + \mathbf{t}) \Leftrightarrow K_2^{-1}\mathbf{x}_2 = RK_1^{-1}\mathbf{x}_1 + \mathbf{t} \Leftrightarrow$$

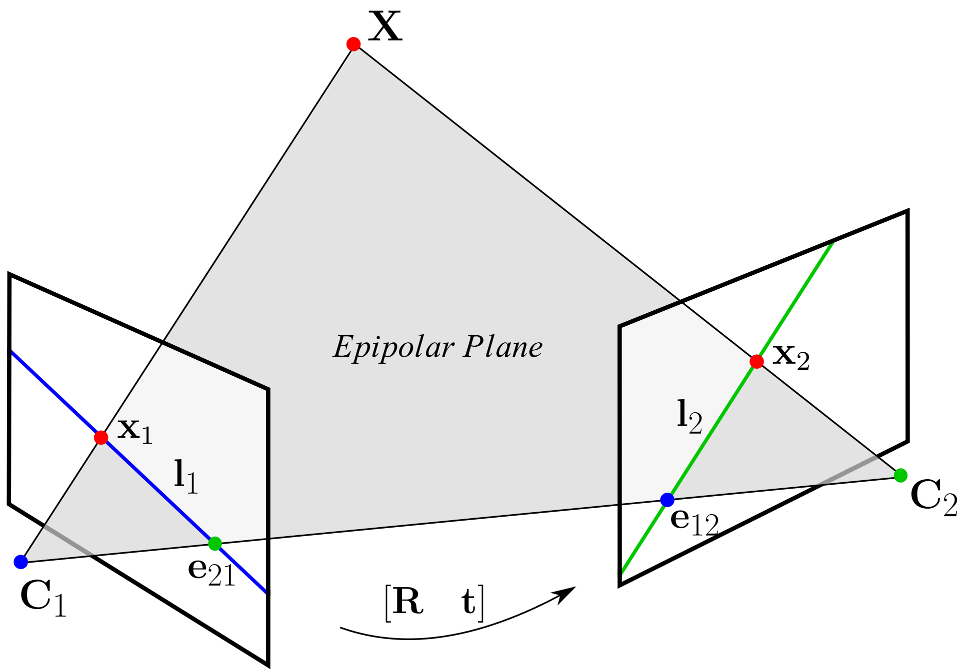
$$\mathbf{t} \times (K_2^{-1}\mathbf{x}_2) = \mathbf{t} \times RK_1^{-1}\mathbf{x}_1 \Leftrightarrow$$

$$(K_2^{-1}\mathbf{x}_2)^\top [\mathbf{t} \times (K_2^{-1}\mathbf{x}_2)] = (K_2^{-1}\mathbf{x}_2)^\top [\mathbf{t} \times R] (K_1^{-1}\mathbf{x}_1) = 0$$

With the notation $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$

$$(K_2^{-1}\mathbf{x}_2)^\top [\mathbf{t}]_\times R(K_1^{-1}\mathbf{x}_1) = 0$$

Essential and Fundamental Matrices



where

- $\mathbf{e}_{21}$ is the projection of $\mathbf{C}_2$ onto P_1
- $\mathbf{e}_{12}$ is the projection of $\mathbf{C}_1$ onto P_2
- $\mathbf{e}_{21}$ and $\mathbf{e}_{12}$ are the **epipoles**
- $\mathbf{l}_1$ and $\mathbf{l}_2$ are the **epipolar lines**

$$E = [\mathbf{t}]_\times R$$

is the **essential matrix** and encode the roto-translation between two cameras. E can be **decomposed** to obtain R and $\mathbf{t}/||\mathbf{t}||$ (i.e., translation can be recovered up to a scale factor ambiguity)

If $\mathbf{K}_1$ and $\mathbf{K}_2$ are unknown, we have the **fundameltal matrix** F

$$(K_2^{-1}\mathbf{x}_2)^\top E(K_1^{-1}\mathbf{x}_1) = (K_2^{-1}\mathbf{x}_2)^\top[\mathbf{t}]_\times R(K_1^{-1}\mathbf{x}_1) =$$

$$\mathbf{x}_2^\top K_2^{-\top}[\mathbf{t}]_\times RK_1^{-1}\mathbf{x}_1 = \mathbf{x}_2^\top F\mathbf{x}_1 = 0$$

and

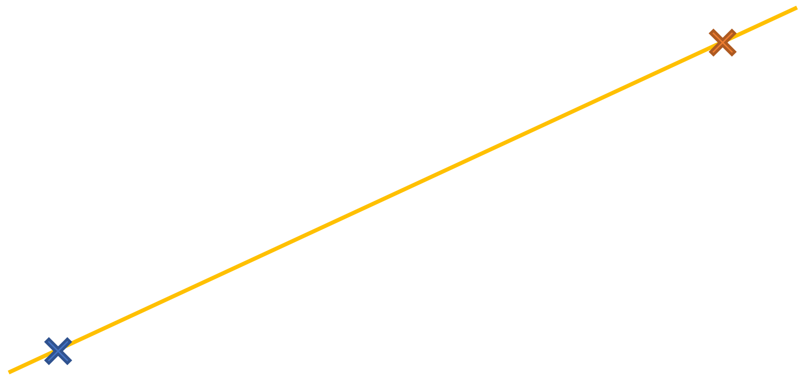
$$E = K_2^\top F K_1$$

Lines in homogeneous coordinates

With homogeneous coordinates

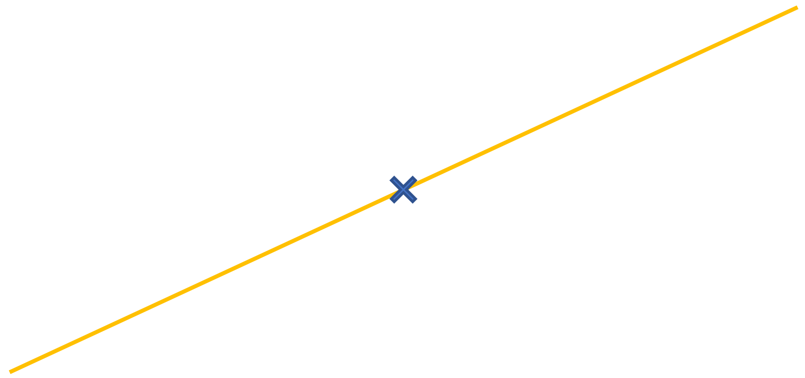
- a **line** passing though **two points** is given by

$$\mathbf{l}_{12} = \mathbf{x}_1 \times \mathbf{x}_2$$

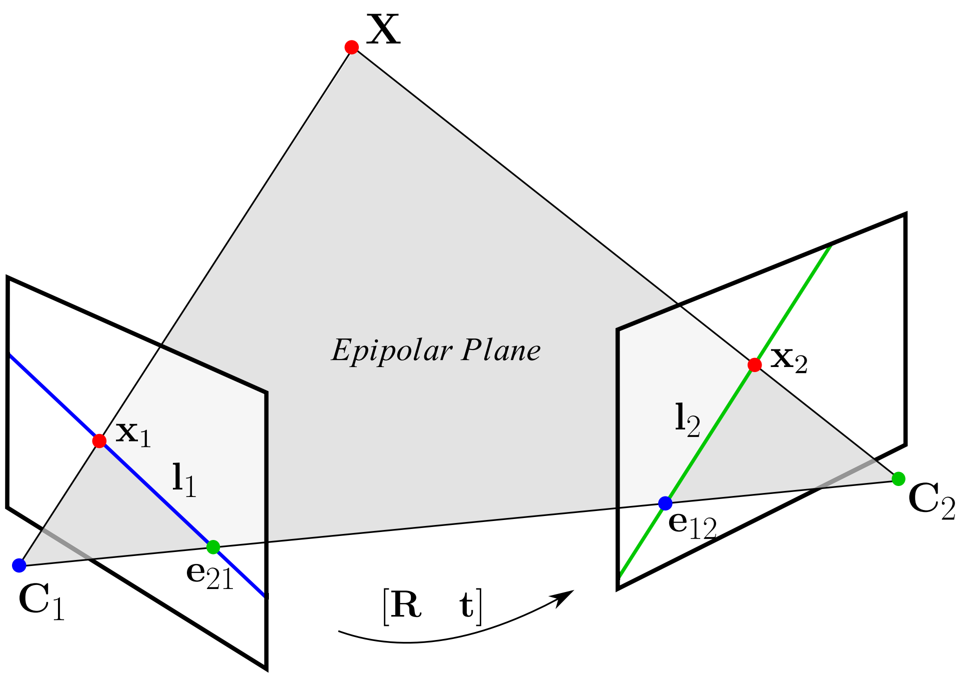


- and a **point lies on a line** iif

$$\mathbf{l}_{12}^{\top}\mathbf{x}_1 = (\mathbf{x}_1 \times \mathbf{x}_2)^{\top}\mathbf{x}_1 = 0$$



Fundamental Matrix



where

- $\mathbf{e}_{21}$ is the projection of $\mathbf{C}_2$ onto P_1
- $\mathbf{e}_{12}$ is the projection of $\mathbf{C}_1$ onto P_2
- $\mathbf{e}_{21}$ and $\mathbf{e}_{12}$ are the **epipoles**
- $\mathbf{l}_1$ and $\mathbf{l}_2$ are the **epipolar lines**

We see that

$$\mathbf{x}_2^\mathsf{T} F \mathbf{x}_1 = 0$$

The **fundamental (essetial) matrix maps points into lines.** Indeed,

$$F\mathbf{x}_1 = \mathbf{l}_2$$

and

$$\mathbf{x}_2^\mathsf{T} \mathbf{l}_2 = 0$$

Also

$$\mathbf{e}_{12}^\mathsf{T} \mathbf{l}_2 = 0 = \mathbf{e}_{12}^\mathsf{T} F \mathbf{x}\,, \forall\, \mathbf{x}$$

so $\mathbf{e}_{12}^\mathsf{T} F = 0$, i.e., $\mathbf{e}_{12}^\mathsf{T}$ is the **left null-space** of F (and similarly, $\mathbf{e}_{21}$ is the **right null-space**)

Fundamental Matrix

Properties:

- F **maps points into lines**
- If F is the fundamental matrix for (P_1, P_2), then $F^\intercal$ is for (P_2, P_1)
- $\mathbf{e}_{12}^\intercal$ is the **left null-space** of F (and similarly, $\mathbf{e}_{21}$ is the **right null-space**)
- F has 7 degree of freedom
- $\det(F) = 0$

Fundamental Matrix

Properties:

- F **maps points into lines**
- If F is the fundamental matrix for $(\mathrm{P}_1, \mathrm{P}_2)$, then $\mathrm{F}^\intercal$ is for $(\mathrm{P}_2, \mathrm{P}_1)$
- $\mathbf{e}_{12}^\intercal$ is the **left null-space** of F (and similarly, $\mathbf{e}_{21}$ is the **right null-space**)
- F has 7 degree of freedom
- $\det(\mathrm{F}) = 0$

- Since $\mathbf{x}_2^\intercal \mathrm{F} \mathbf{x}_1 = 0$, F can be **computed using image correspondences**
  - 7 correspondences + $\det(\mathrm{F}) = 0$
  - 8 correspondences (8-point algorithm)

Funda

Pro

- F
- If
- e … ce)
- F
- d
- Si … s

8-point algorithm

Let $\{\mathbf{x}_i\}$ and $\{\mathbf{x}_j\}$ be the sets of corresponding points between images I and J

1. Since $\mathbf{x}_j^T \mathrm{F} \mathbf{x}_i = 0$, each pair gives rise to

$$\begin{bmatrix} x_j & y_j & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

$$\Leftrightarrow \begin{bmatrix} x_j\, x_i & x_j\, y_i & x_j & y_j\, x_i & y_j\, y_i & y_j & x_i & y_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

2. F can be determined up to a scale factor, so we can put $f_{33} = 1$
3. Stacking at least 8 of such constraints, F can be computed solving a linear system
4. SVD decomposition can be used to solve the homogeneous system

Fundamental Matrix

Properties:

- F **maps points into lines**
- If F is the fundamental matrix for (P_1, P_2), then $F^\top$ is for (P_2, P_1)
- $\mathbf{e}_{12}^\top$ is the **left null-space** of F (and similarly, $\mathbf{e}_{21}$ is the **right null-space**)
- F has 7 degree of freedom
- $\det(F) = 0$

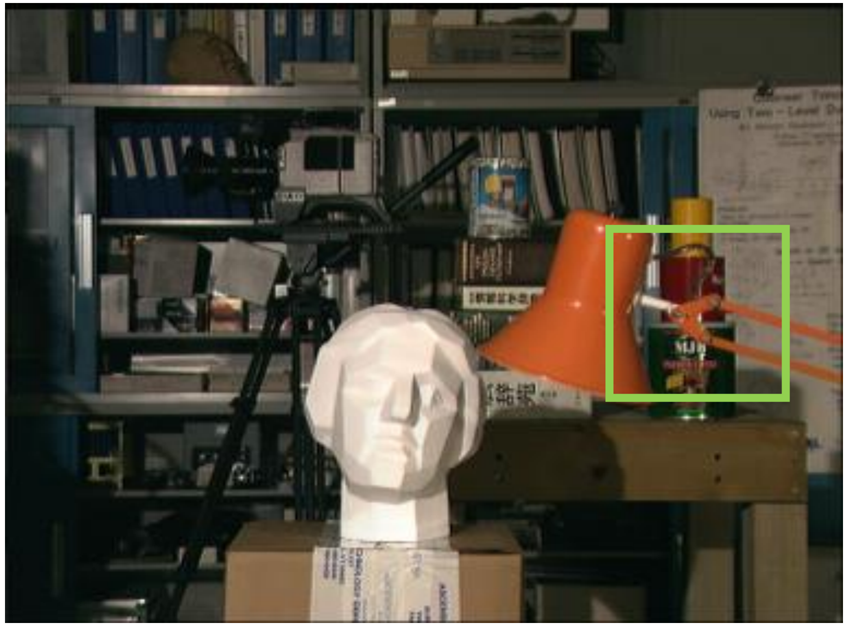
- Since $\mathbf{x}_2^\top F \mathbf{x}_1 = 0$, F can be **computed using image correspondences**
  - 7 correspondences + $\det(F) = 0$
  - 8 correspondences (8-point algorithm)
- Estimation of F is typically available in CV libraries
- If we know the calibration K, F can be upgraded to E

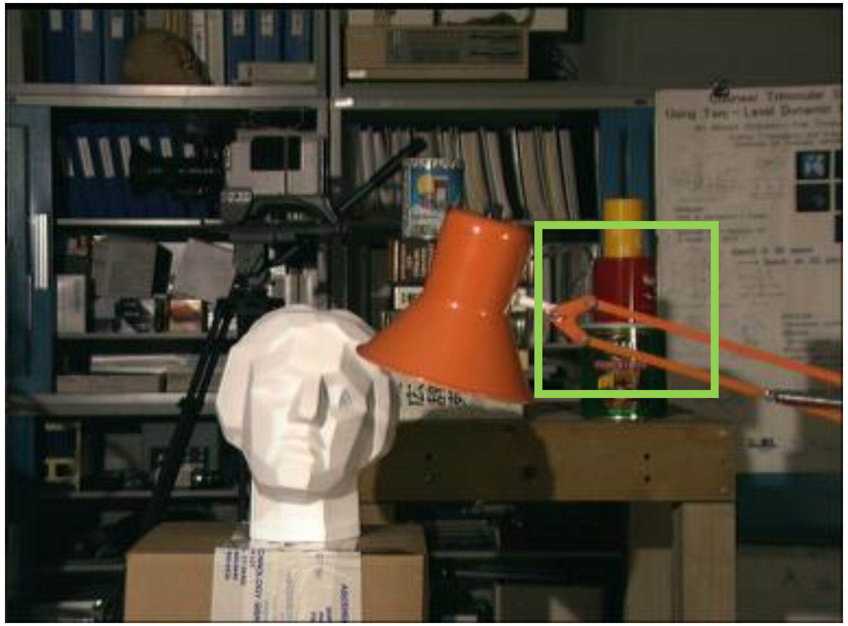
Parallax

- **Parallax** is the **apparent shift** of an object's position against a background due to a change in the observer's **point of view**.

Parallax

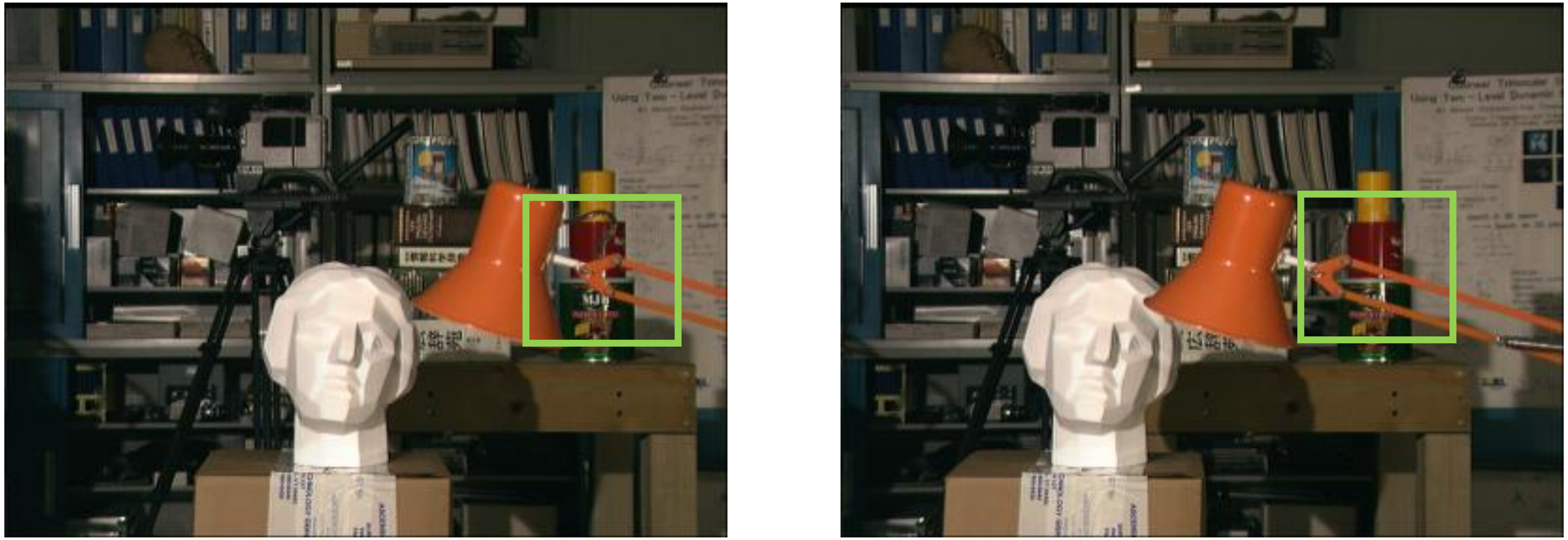
- **Parallax** is the **apparent shift** of an object's position against a background due to a change in the observer's **point of view**.





Parallax

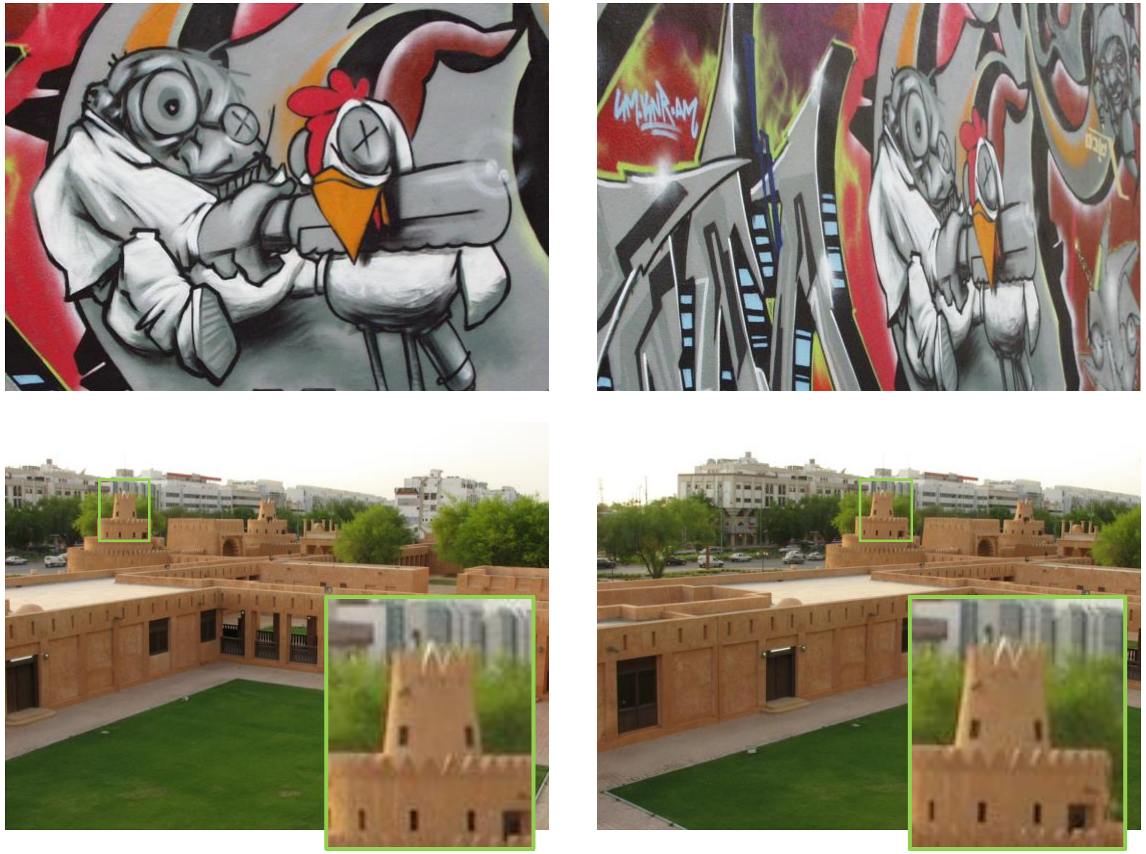
- **Parallax** is the **apparent shift** of an object's position against a background due to a change in the observer's **point of view**.



- We have parallax if
  - The scene is **tridimensional**
  - There is a **translations** between the point of views

Parallax

- Parallax is absent if
  - We observe a **planar** scene
  - We observe a scene at **infinity**
  - We have a **pure rotational** motion
- In these case we **cannot estimate** the fundamental matrix (neither obtain a 3D reconstruction)
- Planar **homography** can model such cases



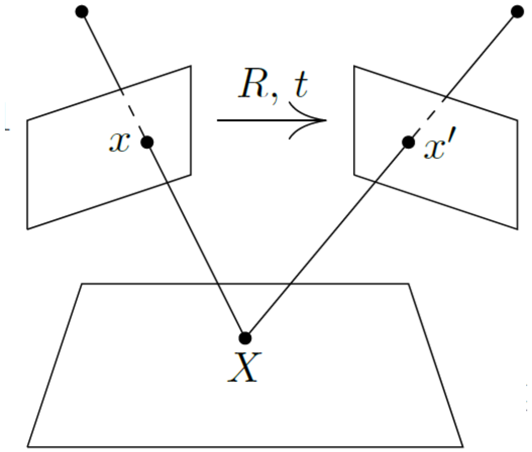
Homographies

- Homographies are projective transformations used to obtain mapping between planes

Homographies

- Homographies are projective transformations used to obtain mapping between planes

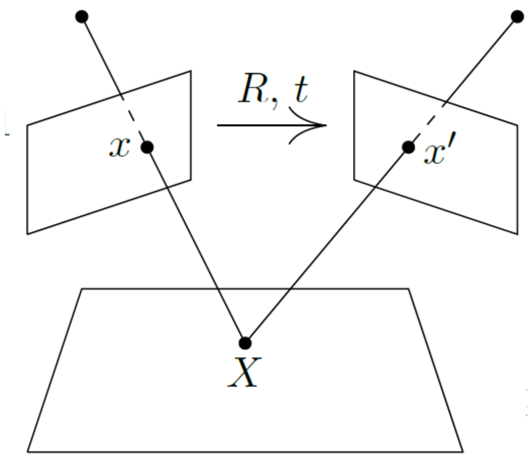
- It can be used to find corresponding image points related to
  - 3D points belonging to a plane

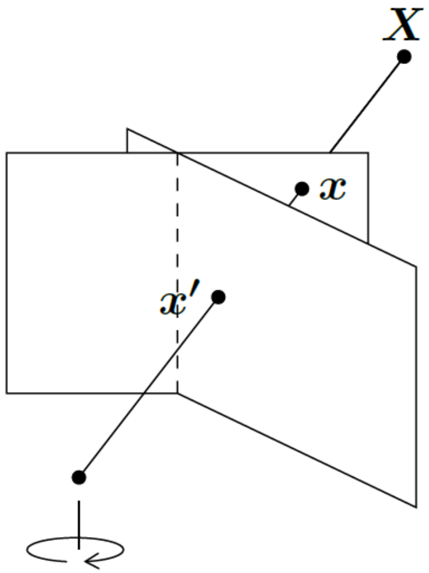


Homographies

- Homographies are projective transformations used to obtain mapping between planes

- It can be used to find corresponding image points related to
  - 3D points belonging to a plane
  - General 3D points acquired by cameras subjected to pure rotation





Homographies

- Given a 3D point $\mathbf{X}$ and its 2D projection $\mathbf{x}$ on the image plane, then exist the following relation

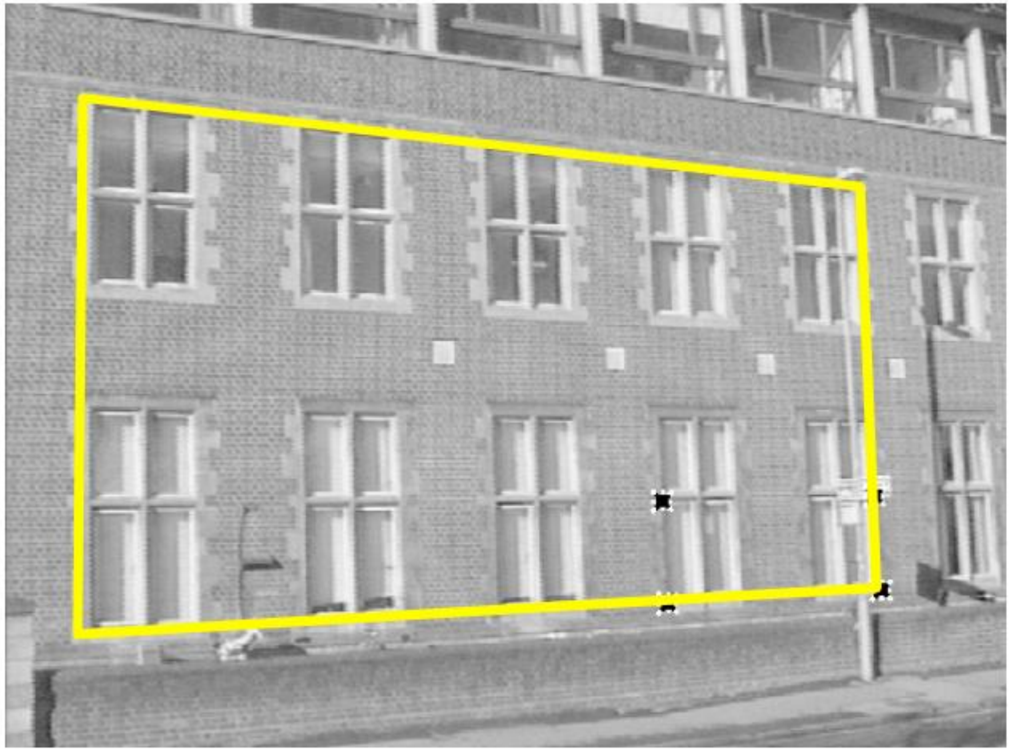
$$\mathbf{x} = \mathrm{H}\mathbf{X}$$

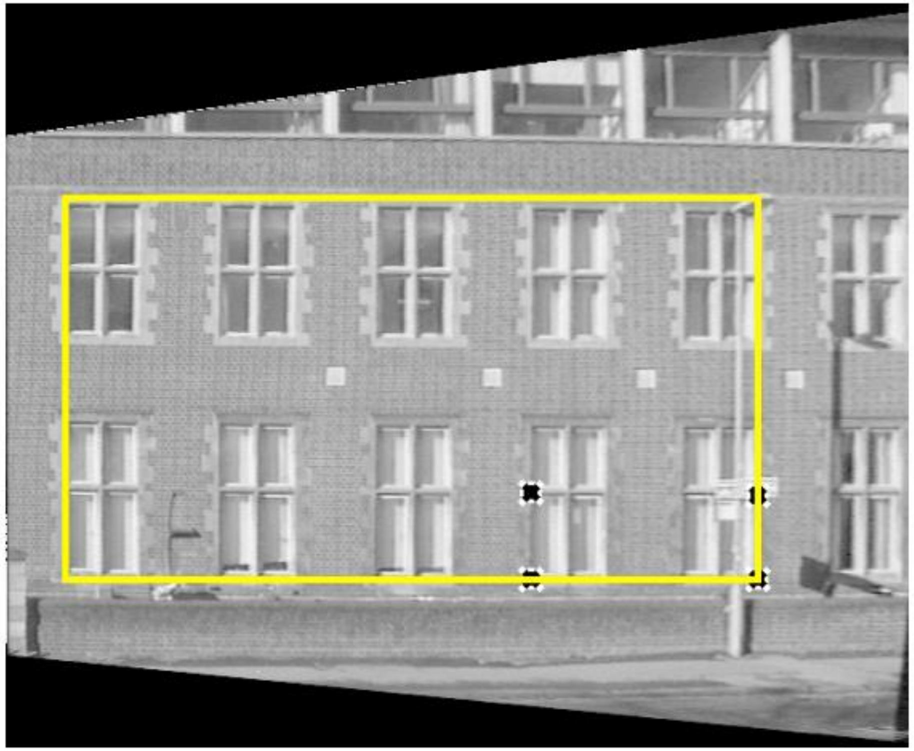
where H is a 3x3 non singular matrix.

- H has 8 degrees of freedom

Homographies

- Homographies can be used to **remove projective distortions**

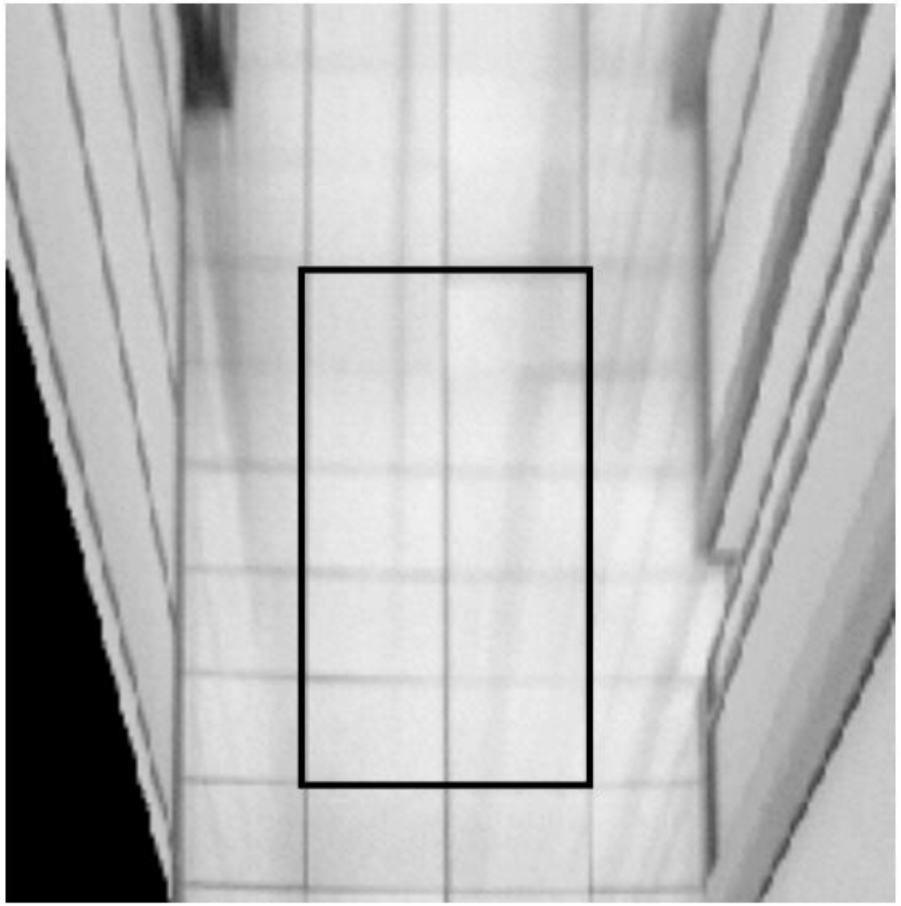




from Hartley & Zisserman

Homographies

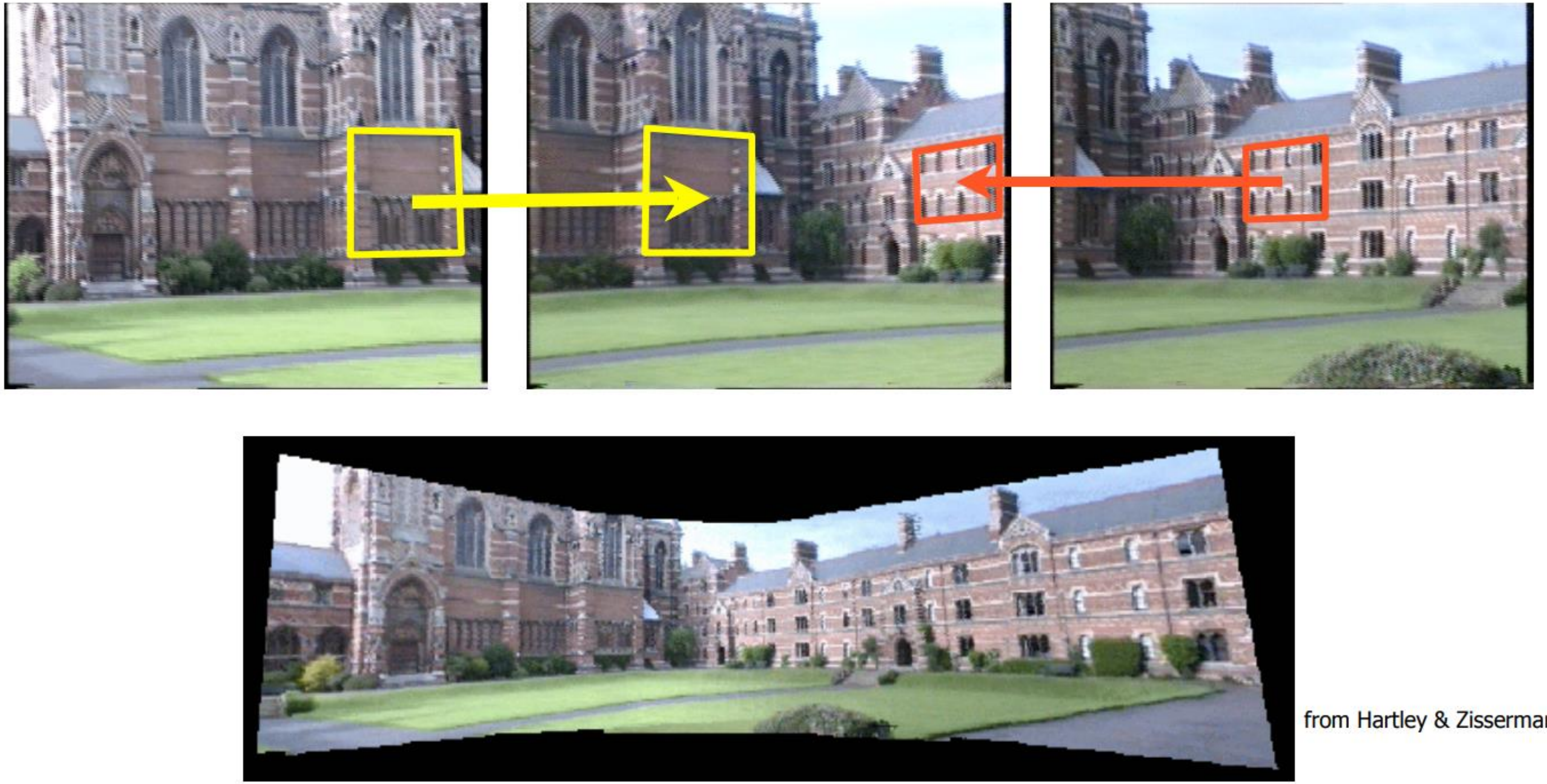
- Homographies can be used to **compute a bird-eye view**



from Hartley & Zisserman

Homographies

- Homographies can be used to **obtain photo mosaics**



from Hartley & Zisserman

Homographies

- Planar homography: map a 3D plane to the image plane

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\mathbf{X}$$

Homographies

- Planar homography: map a 3D plane to the image plane

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\mathbf{X}$$

- If $\mathbf{X} \in \pi$ and suppose $\pi: Z = 0$

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\mathbf{X} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

Homographies

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\mathbf{X} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{r}_1 \mathrm{r}_2 \mathrm{r}_3 \mathbf{t}]\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

Homographies

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\mathbf{X} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{r}_1 \mathrm{r}_2 \mathrm{r}_3 \mathbf{t}]\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{r}_1 \mathrm{r}_2 \mathbf{t}]\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \mathbf{1} \end{bmatrix}$$

Homographies

- $\mathrm{K}[\mathrm{r}_1 \mathrm{r}_2 \mathbf{t}]$ is a 3x3 matrix that we can call H, such that

$$\mathbf{x} = \mathrm{P}\mathbf{X} = \mathrm{K}[\mathrm{r}_1 \mathrm{r}_2 \mathbf{t}] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathrm{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Homographies

- We know that $\mathbf{x}' = \mathrm{H}\mathbf{x}$,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homographies

- We know that $\mathbf{x}' = \mathrm{H}\mathbf{x}$,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

and we can rewrite $\mathrm{H}\mathbf{x}$ as

j-th row of H

$$\mathrm{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1\top}\mathbf{x}_i \\ \mathbf{h}^{2\top}\mathbf{x}_i \\ \mathbf{h}^{3\top}\mathbf{x}_i \end{pmatrix}$$

Homographies

- Since $\mathbf{x}' = \mathrm{H}\mathbf{x}$, by computing the cross-product of both side by $\mathbf{x}'$ we can obtain

$$\mathbf{x}' \times \mathbf{x}' = \mathbf{x}' \times \mathrm{H}\mathbf{x} = \mathbf{0}$$

or by expanding the formula

$$\mathbf{x}'_i \times \mathrm{H}\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}^{3\top}\mathbf{x}_i - w'_i\mathbf{h}^{2\top}\mathbf{x}_i \\ w'_i\mathbf{h}^{1\top}\mathbf{x}_i - x'_i\mathbf{h}^{3\top}\mathbf{x}_i \\ x'_i\mathbf{h}^{2\top}\mathbf{x}_i - y'_i\mathbf{h}^{1\top}\mathbf{x}_i \end{pmatrix}$$

Homographies

- Such equation give rise to three costraints

$$\begin{bmatrix} \mathbf{0}^\top & -w'_i\mathbf{x}_i^\top & y'_i\mathbf{x}_i^\top \\ w'_i\mathbf{x}_i^\top & \mathbf{0}^\top & -x'_i\mathbf{x}_i^\top \\ -y'_i\mathbf{x}_i^\top & x'_i\mathbf{x}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

of which only two are linearly independet

Homographies

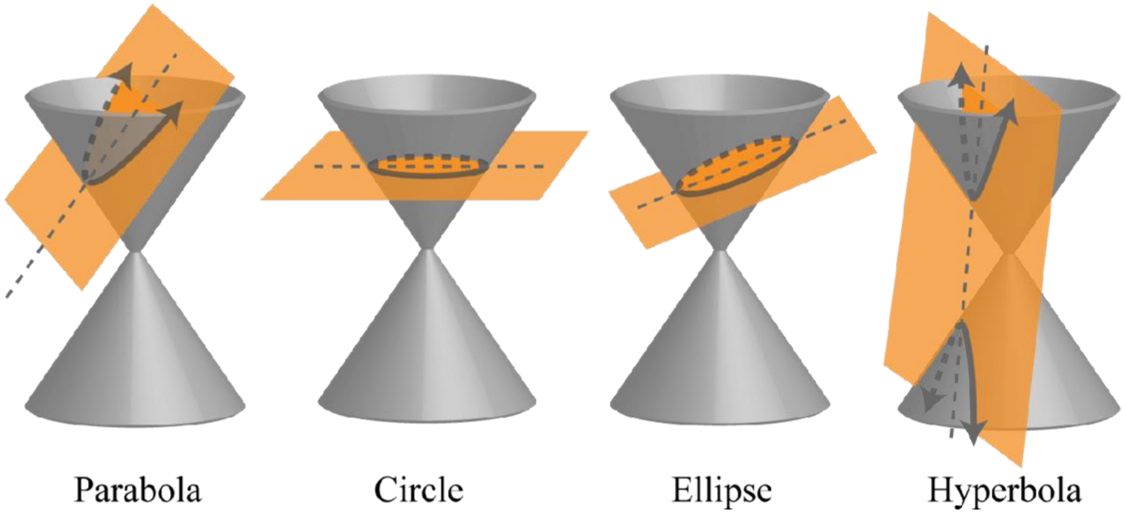
- Such equation give rise to three costraints

$$\begin{bmatrix} \mathbf{0}^\top & -w_i'\mathbf{x}_i^\top & y_i'\mathbf{x}_i^\top \\ w_i'\mathbf{x}_i^\top & \mathbf{0}^\top & -x_i'\mathbf{x}_i^\top \\ -y_i'\mathbf{x}_i^\top & x_i'\mathbf{x}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

  of which only two are linearly independet

- Since H has 8 DoF, we need at least 4 corresponding points to estimate the homography

Conic



- A conic (e.g., parabola, circle, ellipse, and hyperbola) in inhomogeneous coordinates has equation

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

- While in homogenous coordinates, where $x = x_1/x_3$ and $y = x_2/x_3$, we get

$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0$$

Conic

- The equation $ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0$ can be put in matrix form as

$$\mathbf{x}^\top \mathrm{C}\mathbf{x} = 0$$

where

$$\mathrm{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$$

Conic

- The equation $ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0$ can be put in matrix form as

$$\mathbf{x}^\top \mathrm{C}\mathbf{x} = 0$$

where

$$\mathrm{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$$

This is a **symmetric matrix**, defined by six parameters, but since we are in homogeneous coordinates there is a scale ambiguity and **a conic has only 5 DoF**

Conic

- Under a projective transformation H (i.e., an homography) where

$$\mathbf{x}' = \mathrm{H}\mathbf{x}$$

we get

$$\mathbf{x}^\top \mathrm{C}\mathbf{x} = [\mathbf{x}'\mathrm{H}^{-1}]^\top \mathrm{C}[\mathrm{H}^{-1}\mathbf{x}'] =$$

$$= \mathbf{x}'^\top \mathrm{H}^{-\top}\mathrm{C}\mathrm{H}^{-1}\mathbf{x}' =$$

$$\mathbf{x}'^\top [\mathrm{H}^{-\top}\mathrm{C}\mathrm{H}^{-1}]\mathbf{x}' = \mathbf{x}'^\top \mathrm{C}'\mathbf{x}' = 0$$

- So C under projection H maps to $\mathrm{C}' = \mathrm{H}^{-\top}\mathrm{C}\mathrm{H}^{-1}$

Absolute Conic and Plane at Infinity

- The **absolute conic** Ω_∞ is a conic that lies on the **plane at infinity** π_∞

$$\pi_\infty = [0\ 0\ 0\ 1]^\top$$

- A point $\mathbf{X}_\infty \in \pi_\infty$ iif $\mathbf{X}_\infty = [X_1\ X_2\ X_3\ 0]^\top$, indeed

$$\pi_\infty^\top \mathbf{X}_\infty = [0\ 0\ 0\ 1] \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 0 \end{bmatrix} = 0$$

Absolute Conic and Plane at Infinity

- A point $\mathbf{X}_\infty \in \Omega_\infty$ iif

$$\begin{cases} X_1^2 + X_2^2 + X_3^2 = 0 \\ X_4 = 0 \end{cases}$$

- This relation can then be expressed as

$$[X_1\ X_2\ X_3]\mathrm{I}\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = 0$$

so

$$\Omega_\infty = \mathrm{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Absolute Conic and Plane at Infinity

- Any 3D plane intersect the plane at infinity π_∞ in a line that is called **line at infinity $\mathbf{L}_\infty$**

- Any **circle** intersect the line at infinity $\mathbf{L}_\infty$ in two points known as the circular points. Indeed

$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0$$

in the case of a circle, $a = c$ and $b = 0$. So setting $a = c = 1$ we get

$$x_1^2 + x_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0$$

and since it intersect $\mathbf{L}_\infty \in \pi_\infty$, $x_3 = 0$, then

$$x_1^2 + x_2^2 = 0$$

- Such equation admits two solutions $\mathbf{I}$ and $\mathbf{J}$

Circular Points

- The $\mathbf{I}$ and $\mathbf{J}$ solution of $x_1^2 + x_2^2 = 0$ are called **circular points**, where

$$\mathbf{I} = \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} \text{ and } \mathbf{J} = \begin{bmatrix} 1 \\ -i \\ 0 \end{bmatrix}$$

where i is the imaginary unit, such as $i^2 = -1$

- Also, since $\mathbf{I}, \mathbf{J} \in \mathbf{L}_\infty$

$$\mathbf{L}_\infty = \mathbf{I} \times \mathbf{J}$$

- The projections of $\mathbf{I}$ and $\mathbf{J}$ are called **imaged circular points**

- Since any 3D plane intersect π_∞ in $\mathbf{L}_\infty$, the circular points can also be related to a plane

IAC – Image of the Absolute Conic

- To project the absolute conic on the image plane, since $\Omega_\infty \in \pi_\infty$ we can use the homography that exist between π_∞ and the image plane, i.e. H_∞

- H_∞ maps any point on π_∞ onto the image plane. Since $\mathbf{X}_\infty \in \pi_\infty$ iif $\mathbf{X}_\infty = [X_1\ X_2\ X_3\ 0]^\mathrm{T}$ we can write

$$\mathbf{x} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\mathbf{X}_\infty = \mathrm{K}[\mathrm{R}\ \mathbf{t}]\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 0 \end{bmatrix} = \mathrm{KR}\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \mathrm{H}_\infty\, \mathbf{X}_\infty$$

- So, $\mathrm{H}_\infty = \mathrm{KR}$

IAC – Image of the Absolute Conic

- Using the conic projection we so earlier we can map $\Omega_\infty = I$ to the image plane using $\mathrm{H}_\infty = \mathrm{KR}$ as

$$\omega = \mathrm{H}_\infty^{-\top} \Omega \mathrm{H}_\infty^{-1} = [\mathrm{KR}]^{-\top} \mathrm{I} [\mathrm{KR}]^{-1} = \mathrm{K}^{-\top} \mathrm{R}^{-\top} \mathrm{R}^{-1} \mathrm{K}^{-1}$$

- Since R is an orthonormal matrix, $\mathrm{R}^{-1} = \mathrm{R}^\top$ and $\mathrm{R}\mathrm{R}^{-1} = \mathrm{I}$, we can obtain

$$\omega = \mathrm{K}^{-\top} \mathrm{R}^{-\top} \mathrm{R}^{-1} \mathrm{K}^{-1} = \mathrm{K}^{-\top} (\mathrm{R}^{-1})^\top \mathrm{R}^{-1} \mathrm{K}^{-1} =$$
$$= \mathrm{K}^{-\top} \mathrm{R}\mathrm{R}^{-1} \mathrm{K}^{-1} = \mathrm{K}^{-\top} \mathrm{K}^{-1} = (\mathrm{K}\mathrm{K}^\top)^{-1}$$

- So, the IAC $\omega = (\mathrm{K}\mathrm{K}^\top)^{-1}$ depends on the calibration matrix K

IAC – Image of the Absolute Conic

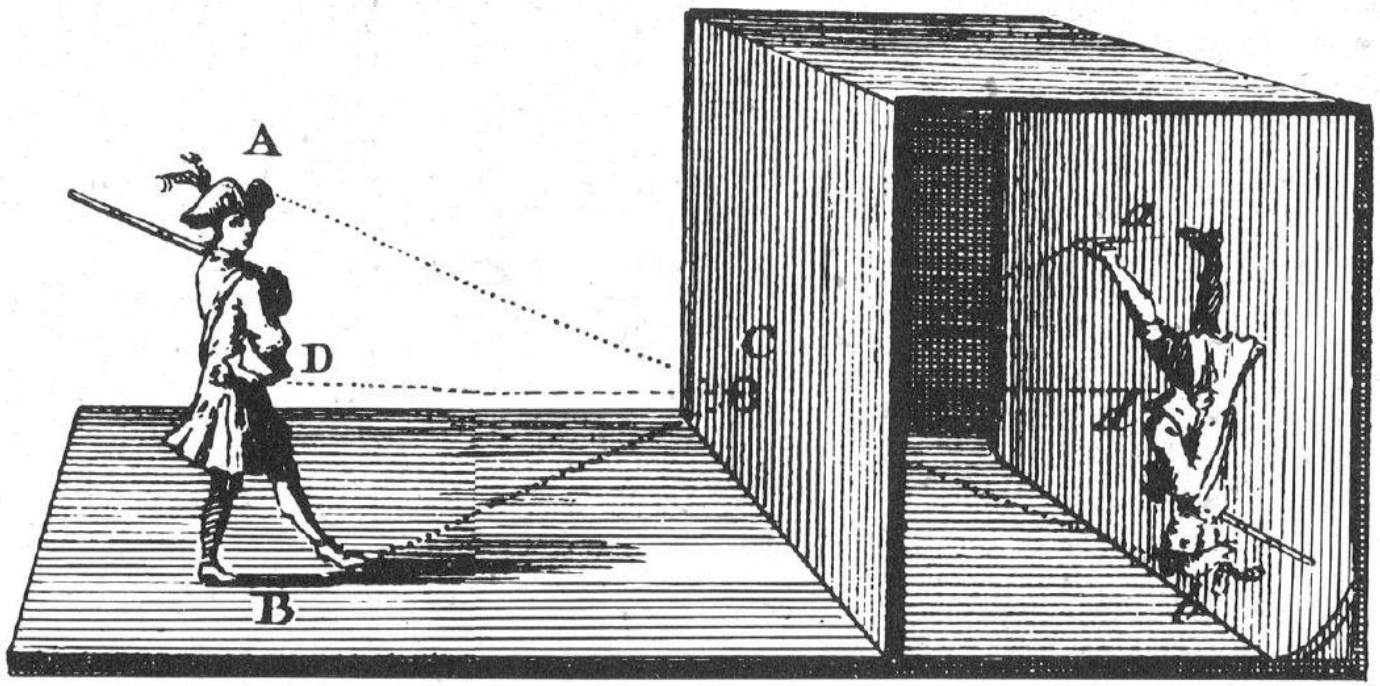
- The IAC $\omega = (\mathrm{K}\mathrm{K}^{\top})^{-1}$ depends on the calibration matrix K

- ω can be decomposed to obtain the calibration K using the Cholesky factorisation since K is an upper-triangular matrix

- So, being able to estimate ω will led us to obtain the camera calibration matrix K

Conclusions

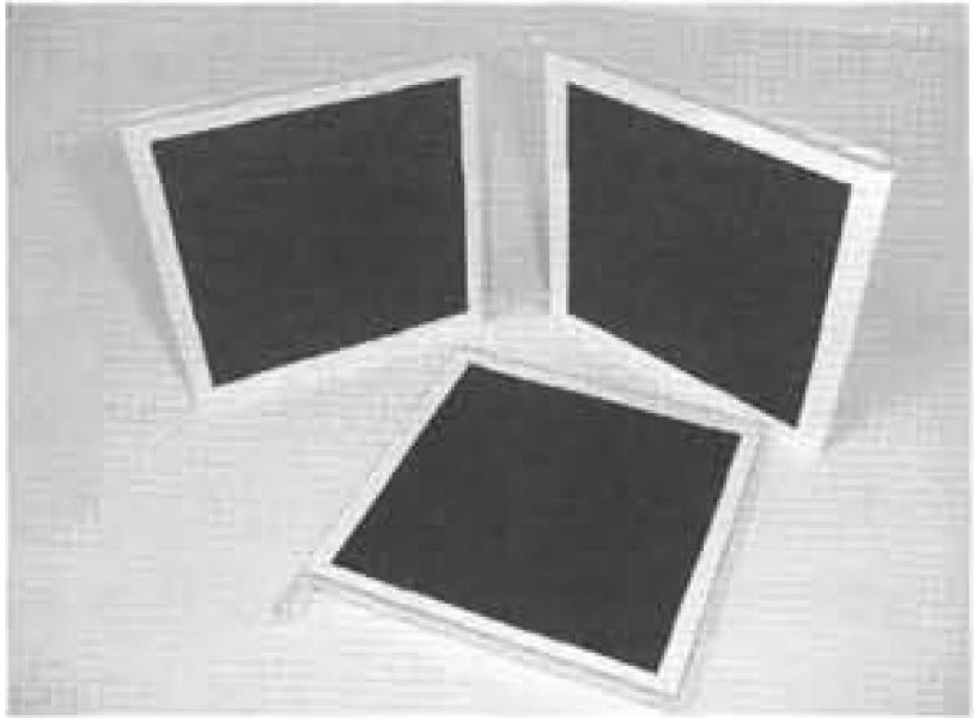
- Camera projection
- Homogeneous coordinates
- Full camera matrix (P)
- Epipolar geometry
  - Essential matrix (E)
  - Fundamental matrix (F)
- Parallax effect
- Homographies
- Conics
- Absolute conic
- Plane at infinity
- Circular points
- Image of the Absolute Conic (IAC)



Camera Calibration

Camera calibration

- A view of three non parallel planes can be used to compute the intrinsic matrix (i.e., the calibration matrix K)



Camera calibration

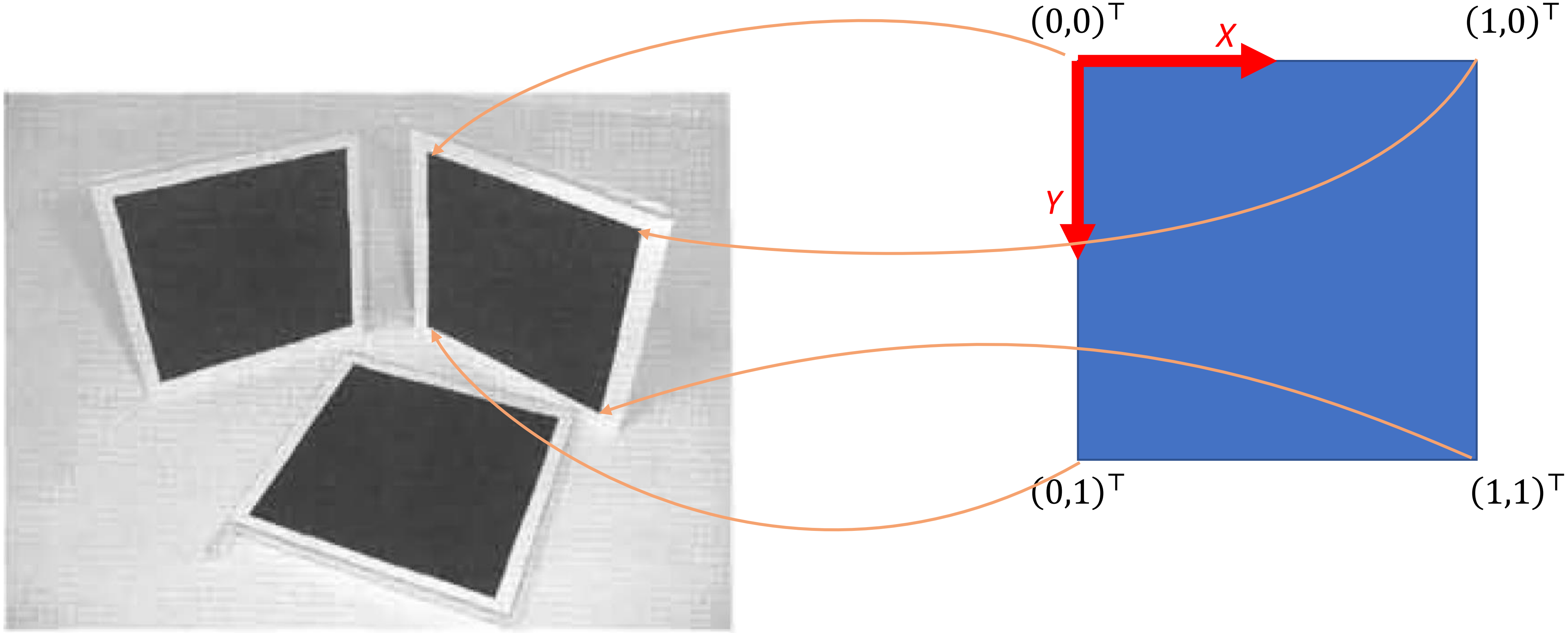
- A view of three non parallel planes can be used to compute the intrinsic matrix (i.e., the calibration matrix K)

- For convenience we can use three planar squared pattern

- Each of the planes give rise to an homography H_k with $k = 1,2,3$

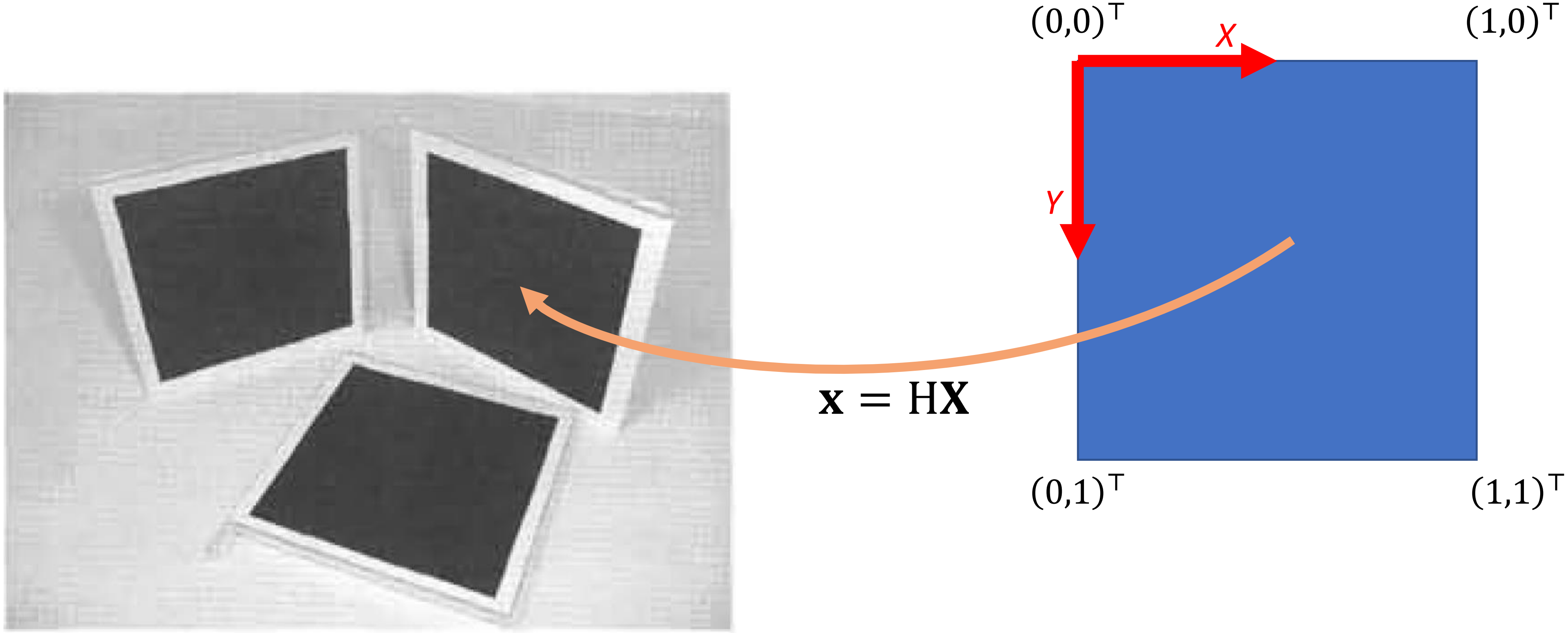
Camera calibration

- Such homographies can be estimated by mapping the image corners to four points as $(0,0)^{\top}, (0,1)^{\top}, (1,0)^{\top}, (1,1)^{\top}$



Camera calibration

- Such homographies can be estimated by mapping the image corners to four points as $(0,0)^\top, (0,1)^\top, (1,0)^\top, (1,1)^\top$



Camera calibration

- The obtained homography H_k can then be used to map the circular points $\mathbf{I}$, $\mathbf{J}$

Camera calibration

- The obtained homography H_k can then be used to map the circular points $\mathbf{I}$, $\mathbf{J}$

- Remember that $\mathbf{I}, \mathbf{J} \in \mathbf{L}_\infty$ and $\mathbf{L}_\infty$ is the intersection of any plane π with π_∞. So $\mathbf{I}, \mathbf{J}$ also belong to π

Camera calibration

- The obtained homography H_k can then be used to map the circular points $\mathbf{I}$, $\mathbf{J}$

- Remember that $\mathbf{I}, \mathbf{J} \in \mathbf{L}_\infty$ and $\mathbf{L}_\infty$ is the intersection of any plane π with π_∞. So $\mathbf{I}$, $\mathbf{J}$ also belong to π

$$\mathbf{i} = \mathrm{H}\mathbf{I} = [\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3]\begin{bmatrix}1\\ i\\ 0\end{bmatrix} = [\mathbf{h}_1 + i\mathbf{h}_2]$$

Camera calibration

- The obtained homography H_k can then be used to map the circular points $\mathbf{I}$, $\mathbf{J}$

- Remember that $\mathbf{I}$, $\mathbf{J} \in \mathbf{L}_\infty$ and $\mathbf{L}_\infty$ is the intersection of any plane π with π_∞. So $\mathbf{I}$, $\mathbf{J}$ also belong to π

$$\mathbf{i} = \mathrm{H}\mathbf{I} = [\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3]\begin{bmatrix}1\\ i\\ 0\end{bmatrix} = [\mathbf{h}_1 + i\mathbf{h}_2]$$

$$\mathbf{j} = \mathrm{H}\mathbf{J} = [\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3]\begin{bmatrix}1\\ -i\\ 0\end{bmatrix} = [\mathbf{h}_1 - i\mathbf{h}_2]$$

Camera calibration

- Since $\mathbf{I}, \mathbf{J} \in \Omega$, then $\mathbf{i}, \mathbf{j} \in \omega$

Camera calibration

- Since $\mathbf{I}, \mathbf{J} \in \Omega$, then $\mathbf{i}, \mathbf{j} \in \omega$
- We can use the imaged circular points obtained from H to estimate ω

$$[\mathbf{h}_1 \pm i\mathbf{h}_2]^\top \omega [\mathbf{h}_1 \pm i\mathbf{h}_2] = 0$$

Camera calibration

- Since $\mathbf{I}, \mathbf{J} \in \Omega$, then $\mathbf{i}, \mathbf{j} \in \omega$
- We can use the imaged circular points obtained from H to estimate ω

$$[\mathbf{h}_1 \pm i\mathbf{h}_2]^\top \omega [\mathbf{h}_1 \pm i\mathbf{h}_2] = 0$$

- Each of such equations give rise to two constraints by separating the real and imaginary parts

$$\mathbf{h}_1^\top \omega \mathbf{h}_2 = 0$$

$$\mathbf{h}_1^\top \omega \mathbf{h}_1 = \mathbf{h}_2^\top \omega \mathbf{h}_2$$

Camera calibration

- Since $\mathbf{I}, \mathbf{J} \in \Omega$, then $\mathbf{i}, \mathbf{j} \in \omega$
- We can use the imaged circular points obtained from H to estimate ω

$$[\mathbf{h}_1 \pm i\mathbf{h}_2]^\top \omega [\mathbf{h}_1 \pm i\mathbf{h}_2] = 0$$

- Each of such equations give rise to two constraints by separating the real and imaginary parts

$$\mathbf{h}_1^\top \omega \mathbf{h}_2 = 0$$

$$\mathbf{h}_1^\top \omega \mathbf{h}_1 = \mathbf{h}_2^\top \omega \mathbf{h}_2$$

- Those constraints are linear equations in ω

Camera calibration

- Since $\mathbf{I}, \mathbf{J} \in \Omega$, then $\mathbf{i}, \mathbf{j} \in \omega$
- We can use the imaged circular points obtained from H to estimate ω

$$[\mathbf{h}_1 \pm i\mathbf{h}_2]^\top \omega [\mathbf{h}_1 \pm i\mathbf{h}_2] = 0$$

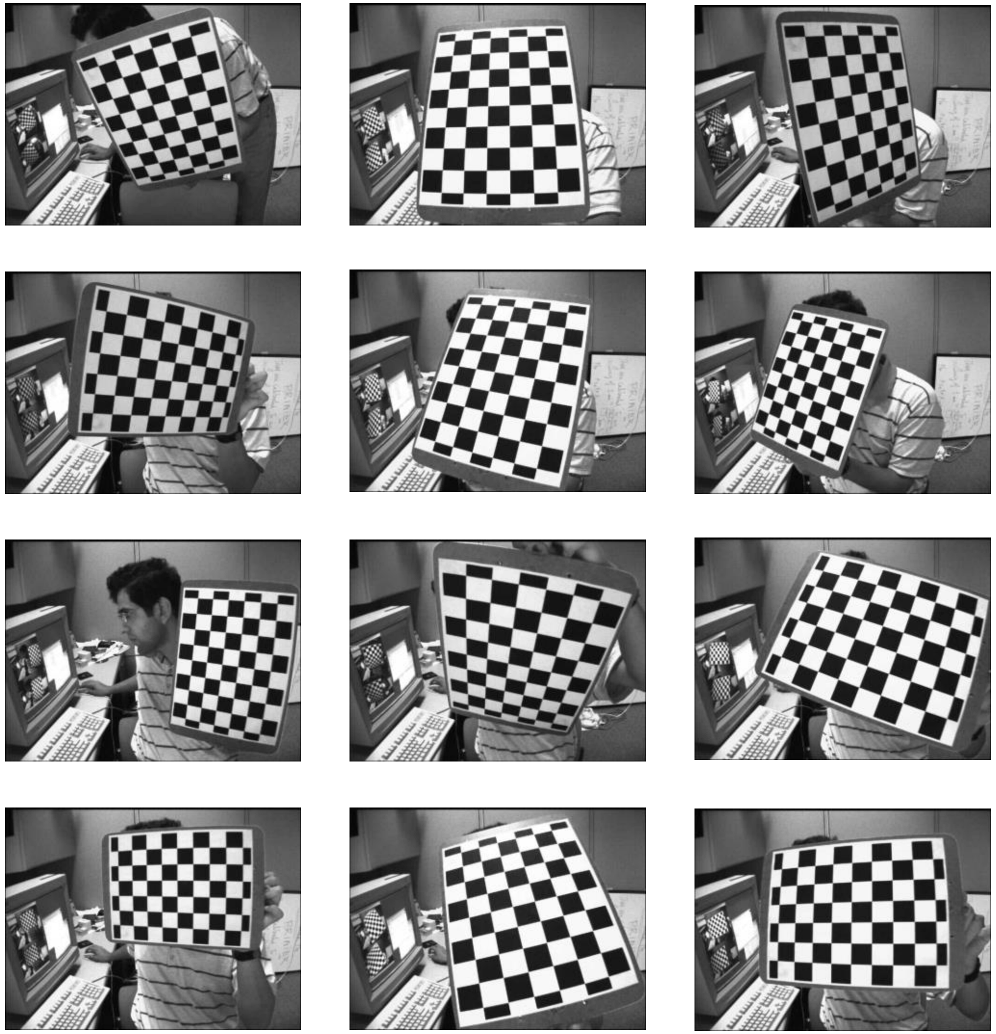
- Each of such equations give rise to two constraints by separating the real and imaginary parts

$$\mathbf{h}_1^\top \omega \mathbf{h}_2 = 0$$

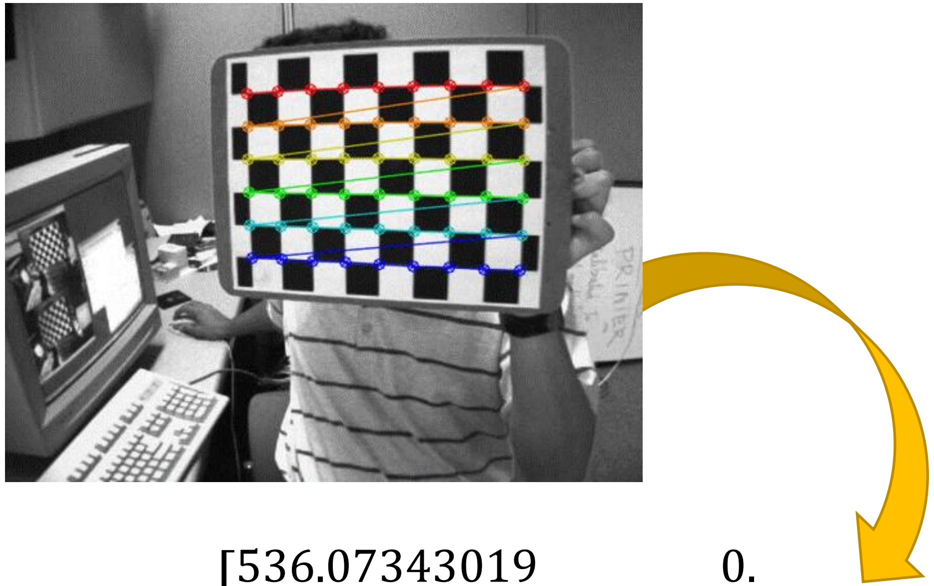
$$\mathbf{h}_1^\top \omega \mathbf{h}_1 = \mathbf{h}_2^\top \omega \mathbf{h}_2$$

- Those constraints are linear equations in ω
- With at least five of such constraints, $\omega = (\mathrm{K}\mathrm{K}^\top)^{-1}$can be estimated and K retrieved by factorization

Camera calibration



- We use several photos of a known **planar pattern** in **different orientation**
- The algorithm is implemented in computer vision libraries, such as OpenCV



$$K=\begin{bmatrix} 536.07343019 & 0. & 342.37038789 \\ 0. & 536.01634475 & 235.53685636 \\ 0. & 0. & 1. \end{bmatrix}$$

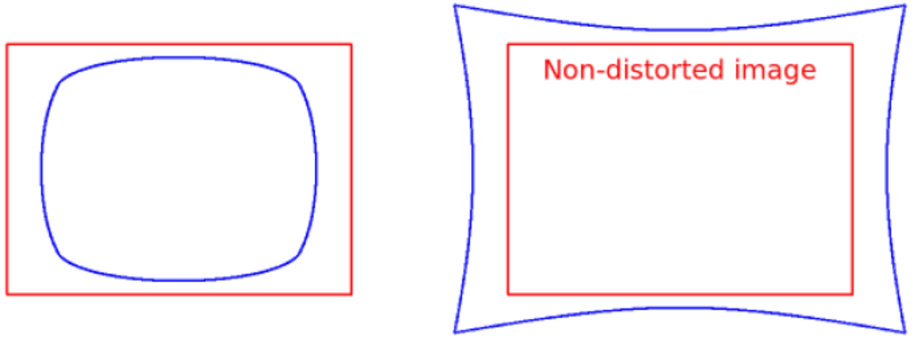
Camera calibration

- It is an effect introduced by **camera lenses**
- Straight line becomes **curves**
- It can be recovered together with the camera calibration

No distortion

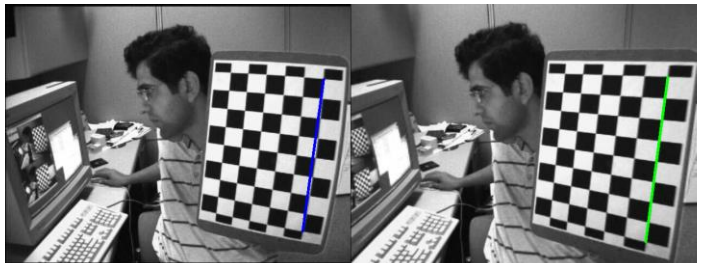
Negative radial distortion (Barrel distortion)

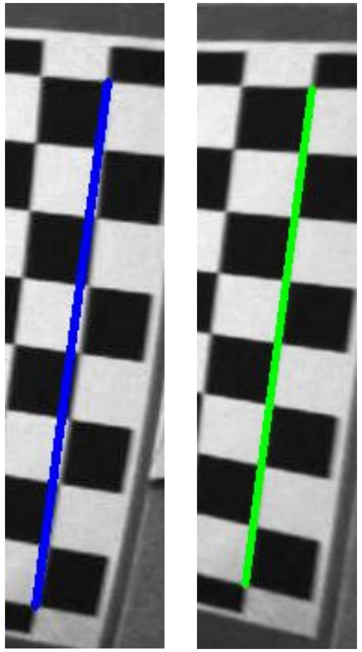
Positive radial distortion (Pincushion distortion)



Negative radial distortion (k1=-1.5) (Barrel distortion)

Positive radial distortion (k1=1.5) (Pincushion distortion)



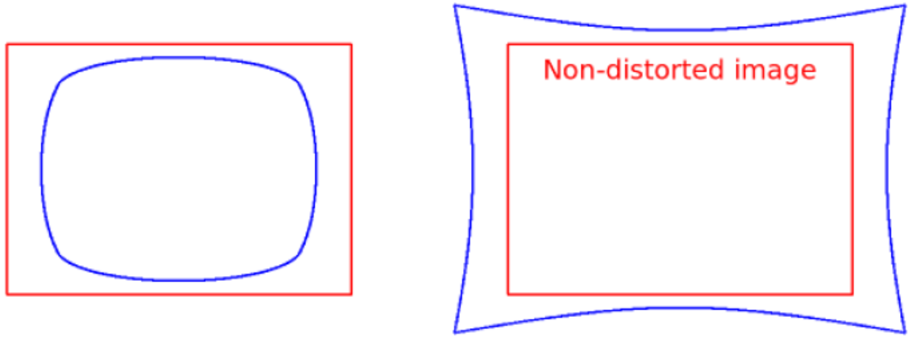


Camera calibration

No distortion

Negative radial distortion
(Barrel distortion)

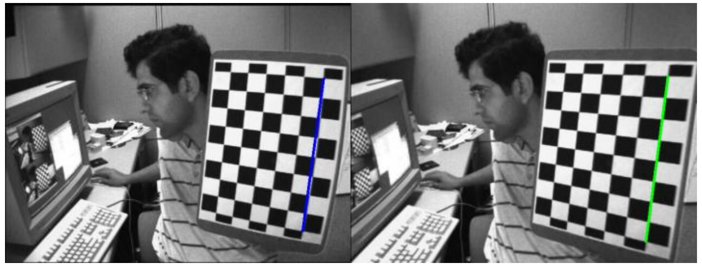
Positive radial distortion
(Pincushion distortion)

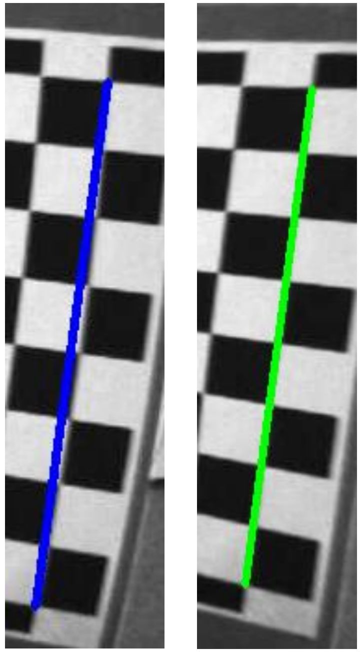


Negative radial distortion (k1=-1.5)
(Barrel distortion)

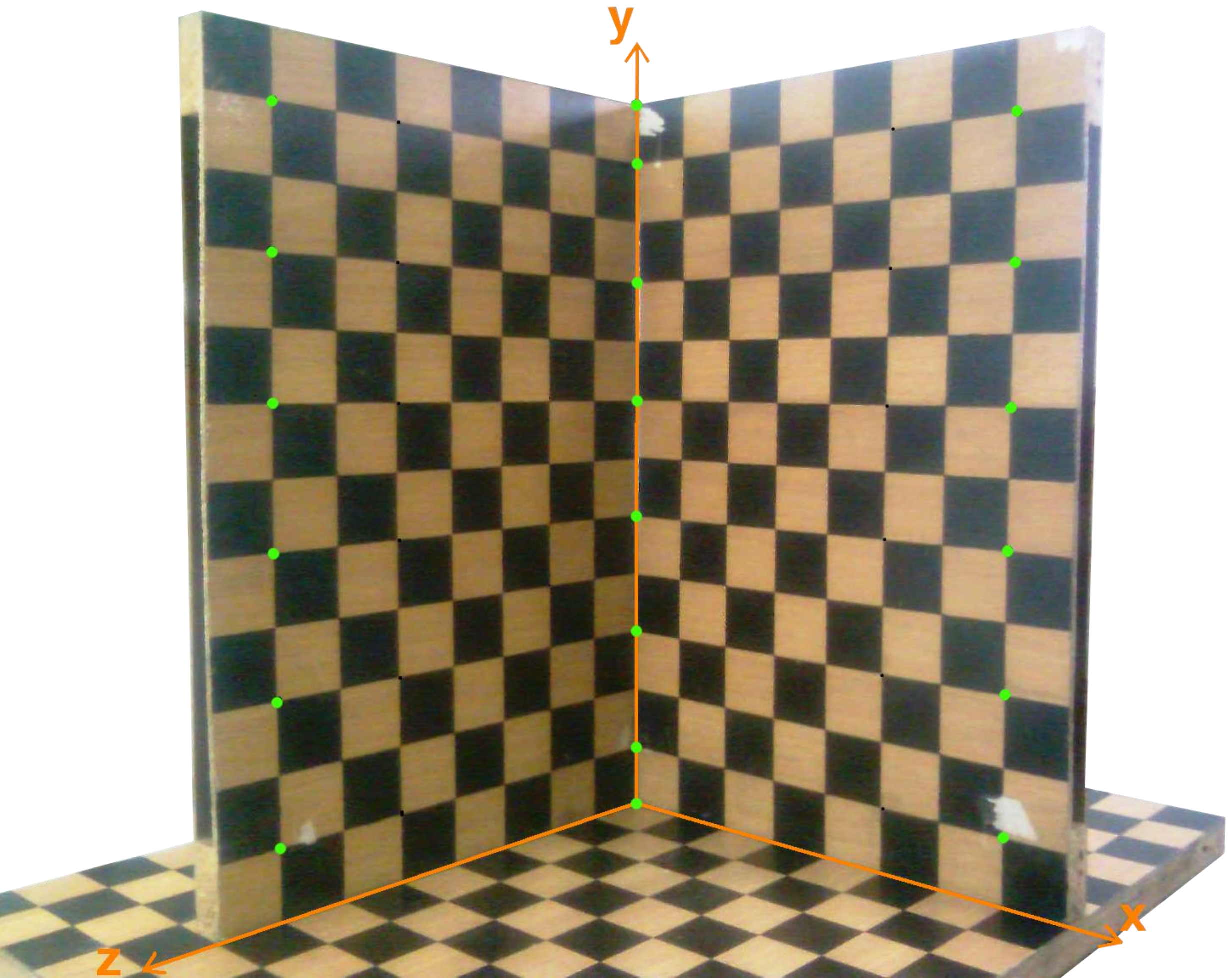
Positive radial distortion (k1=1.5)
(Pincushion distortion)

- It is an effect introduced by **camera lenses**
- Straight line becomes **curves**
- It can be recovered together with the camera calibration



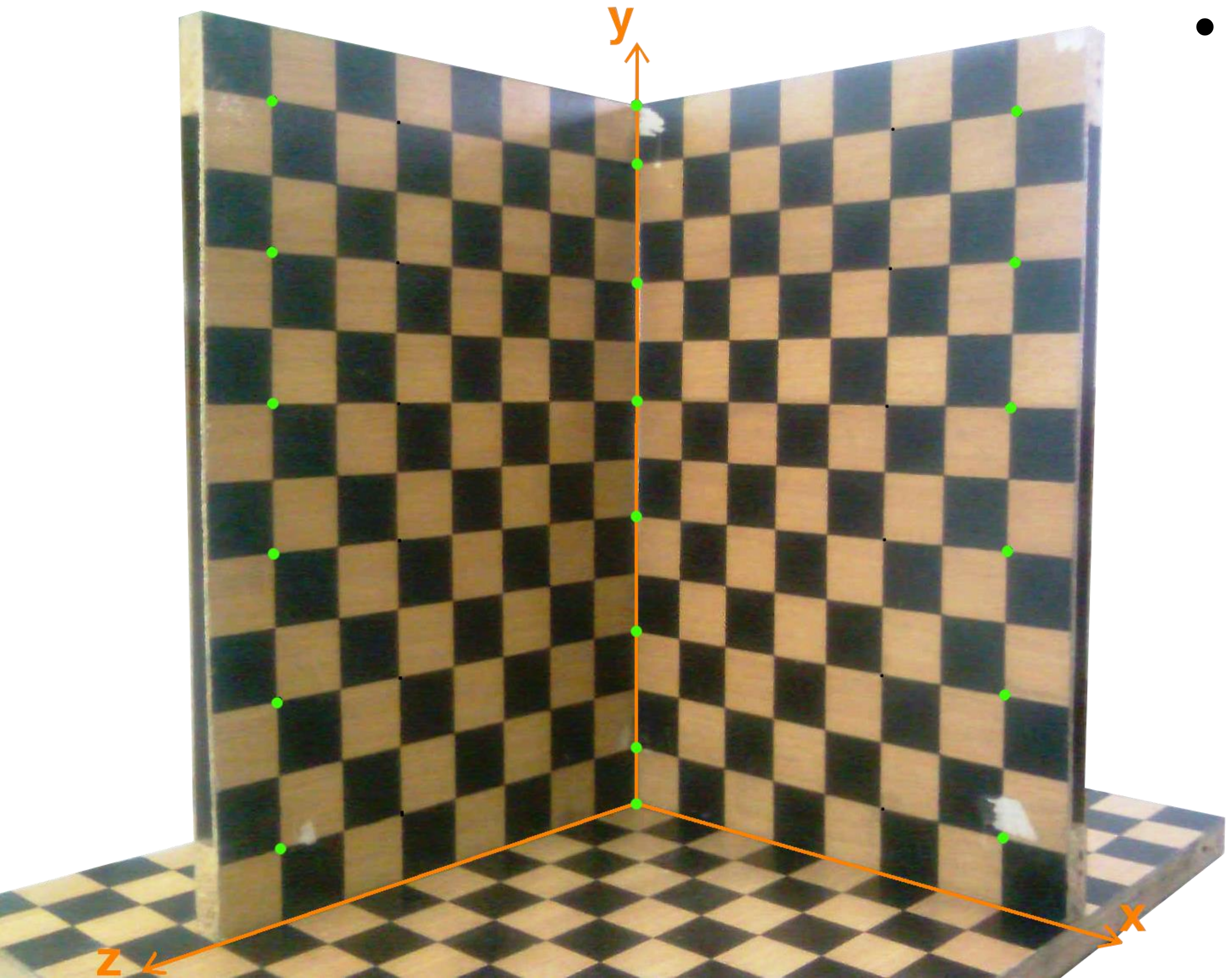


Camera calibration



- A different calibration technique exploit a 3D pattern

Camera calibration

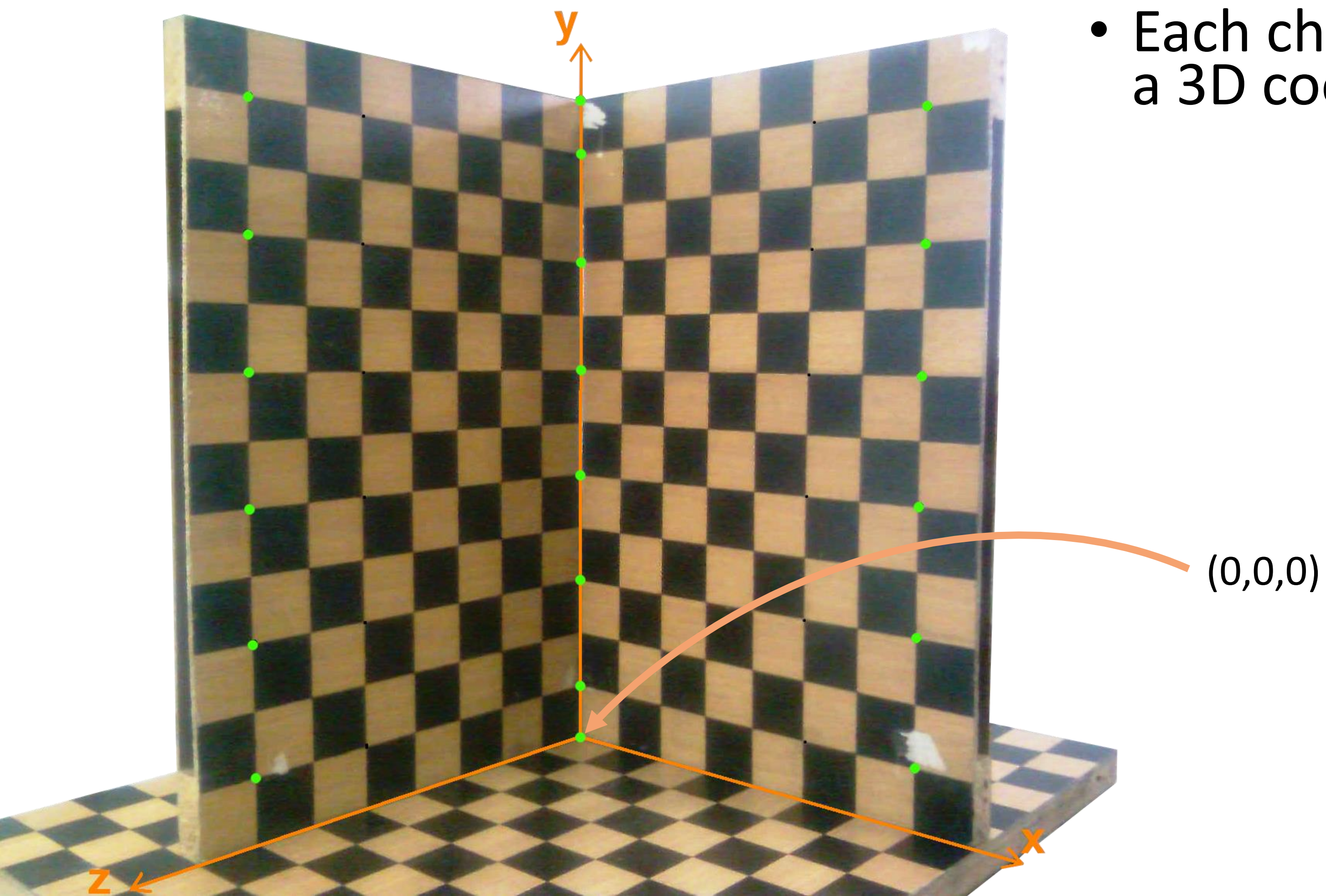


- A different calibration technique exploit a 3D pattern

- Each checkerboard corner can be assigned to a 3D coordinate (X, Y, Z)

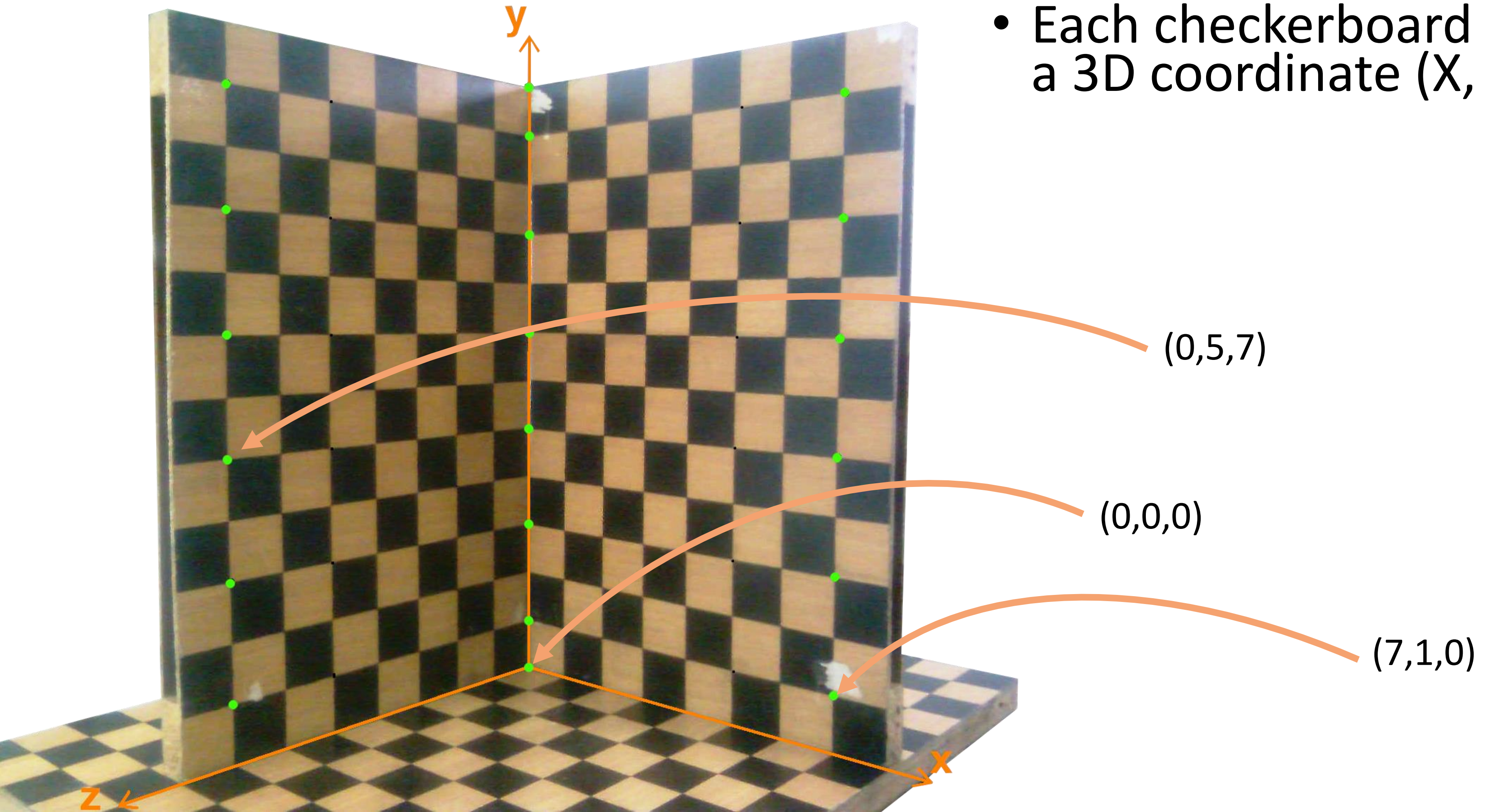
Camera calibration

- A different calibration technique exploit a 3D pattern
- Each checkerboard corner can be assigned to a 3D coordinate (X, Y, Z)

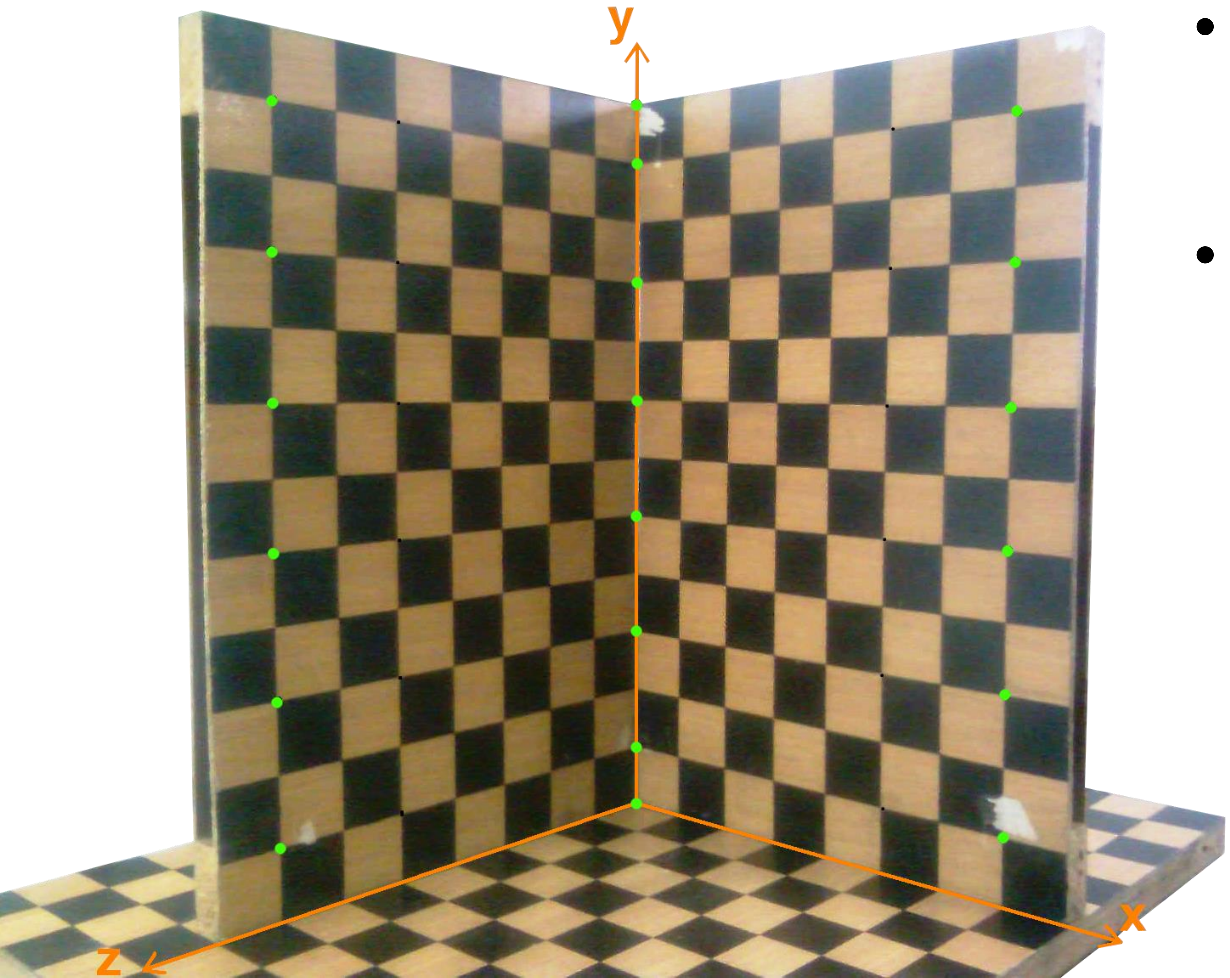


Camera calibration

- A different calibration technique exploit a 3D pattern
- Each checkerboard corner can be assigned to a 3D coordinate (X, Y, Z)



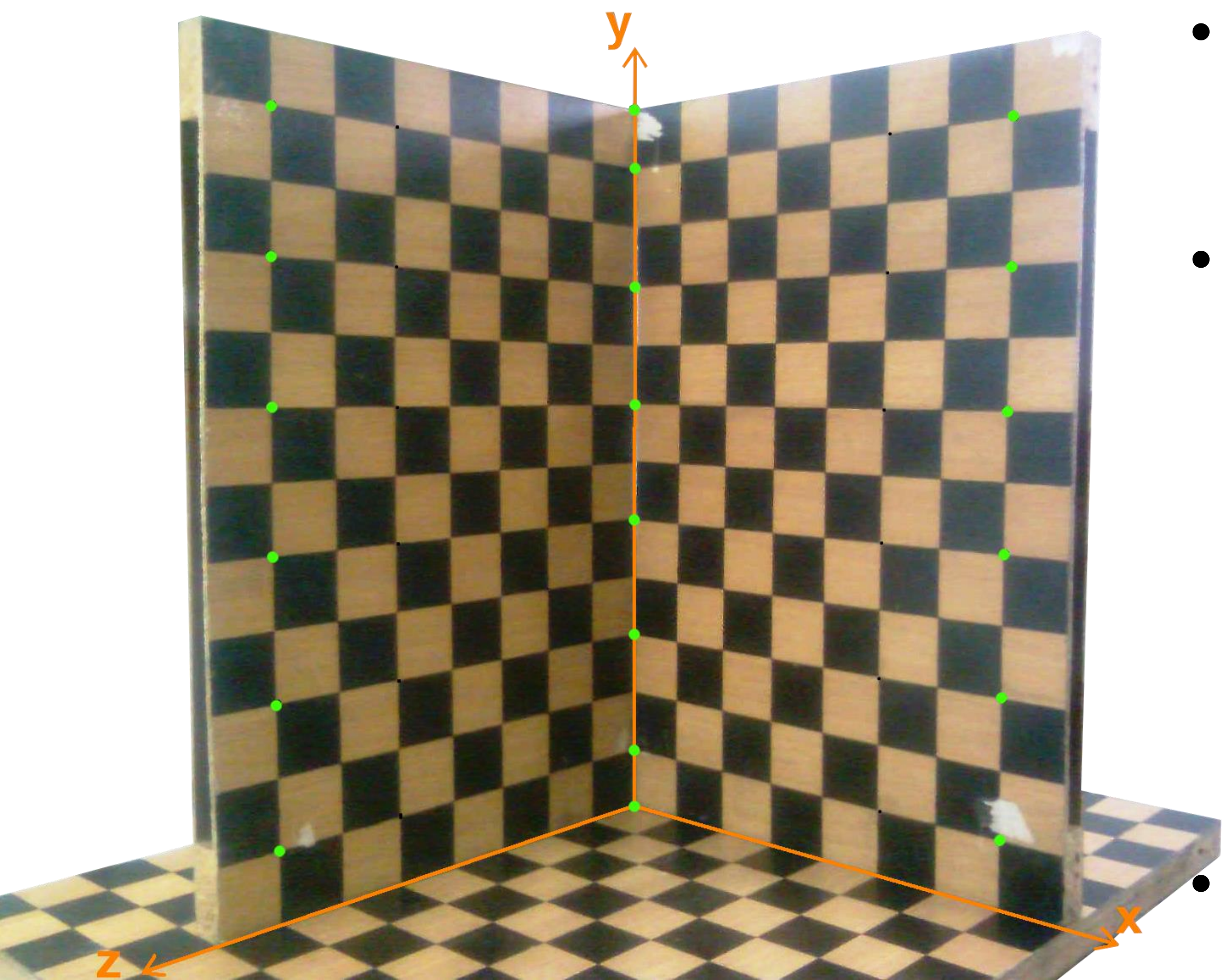
Camera calibration



- A different calibration technique exploit a 3D pattern
- Each checkerboard corner can be assigned to a 3D coordinate (X, Y, Z)
- Knowing the 2D/3D correspondences, the full camera matrix $\mathrm{P} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]$ can be obtained solving a linear system

$$\begin{bmatrix} \mathbf{0}^\top & -w_0\mathbf{X}_0^\top & y_0\mathbf{X}_0^\top \\ w_0\mathbf{X}_0^\top & \mathbf{0}^\top & -x_0\mathbf{X}_0^\top \\ & \vdots & \\ \mathbf{0}^\top & -w_n\mathbf{X}_n^\top & y_n\mathbf{X}_n^\top \\ w_n\mathbf{X}_n^\top & \mathbf{0}^\top & -x_n\mathbf{X}_n^\top \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \end{bmatrix} = \mathbf{0}$$

Camera calibration



- A different calibration technique exploit a 3D pattern

- Each checkerboard corner can be assigned to a 3D coordinate (X, Y, Z)

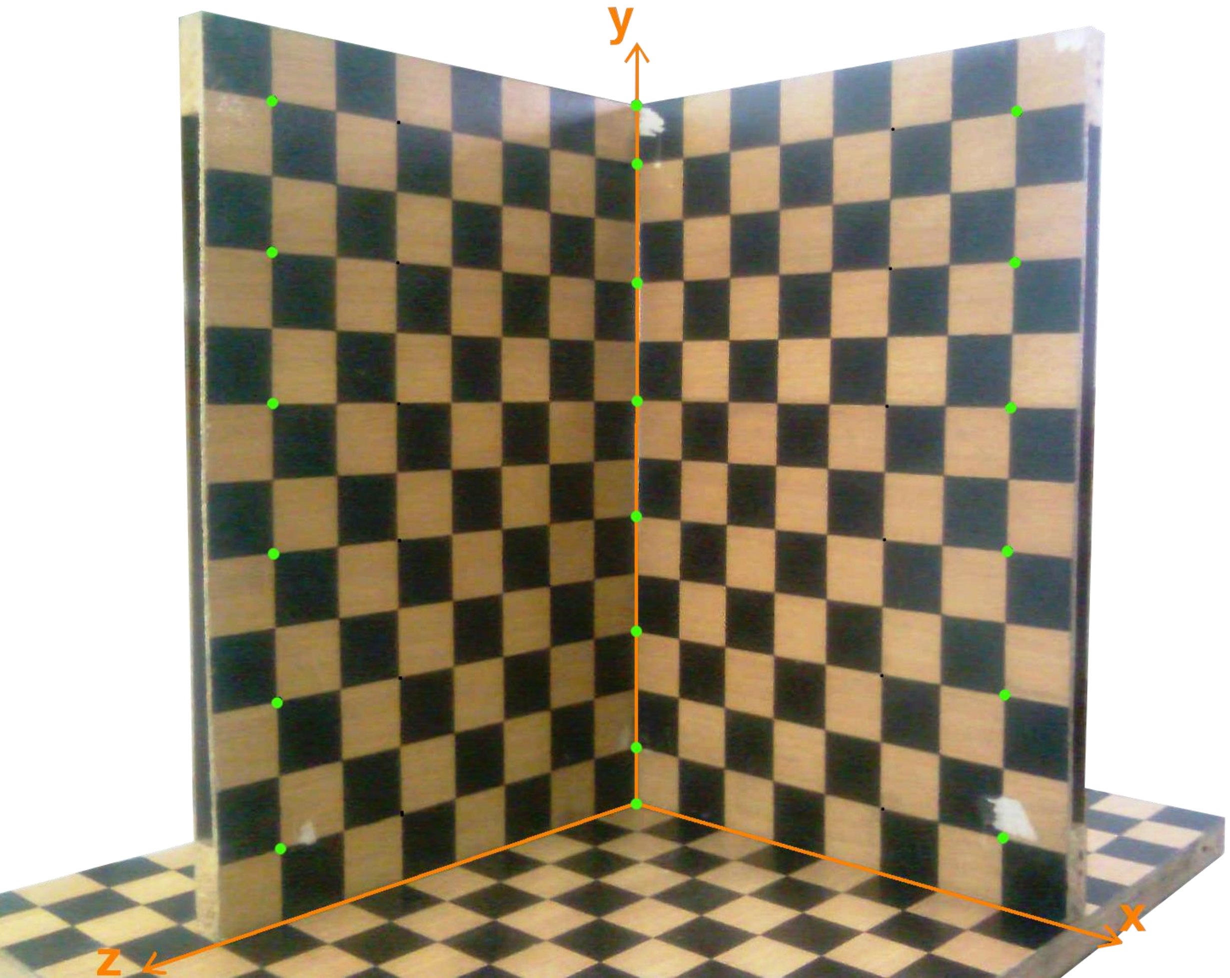
- Knowing the 2D/3D correspondences, the full camera matrix $\mathrm{P} = \mathrm{K}[\mathrm{R}\ \mathbf{t}]$ can be obtained solving a linear system

$$\begin{bmatrix} \mathbf{0}^\top & -w_0\mathbf{X}_0^\top & y_0\mathbf{X}_0^\top \\ w_0\mathbf{X}_0^\top & \mathbf{0}^\top & -x_0\mathbf{X}_0^\top \\ & \vdots & \\ \mathbf{0}^\top & -w_n\mathbf{X}_n^\top & y_n\mathbf{X}_n^\top \\ w_n\mathbf{X}_n^\top & \mathbf{0}^\top & -x_n\mathbf{X}_n^\top \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \end{bmatrix} = \mathbf{0}$$

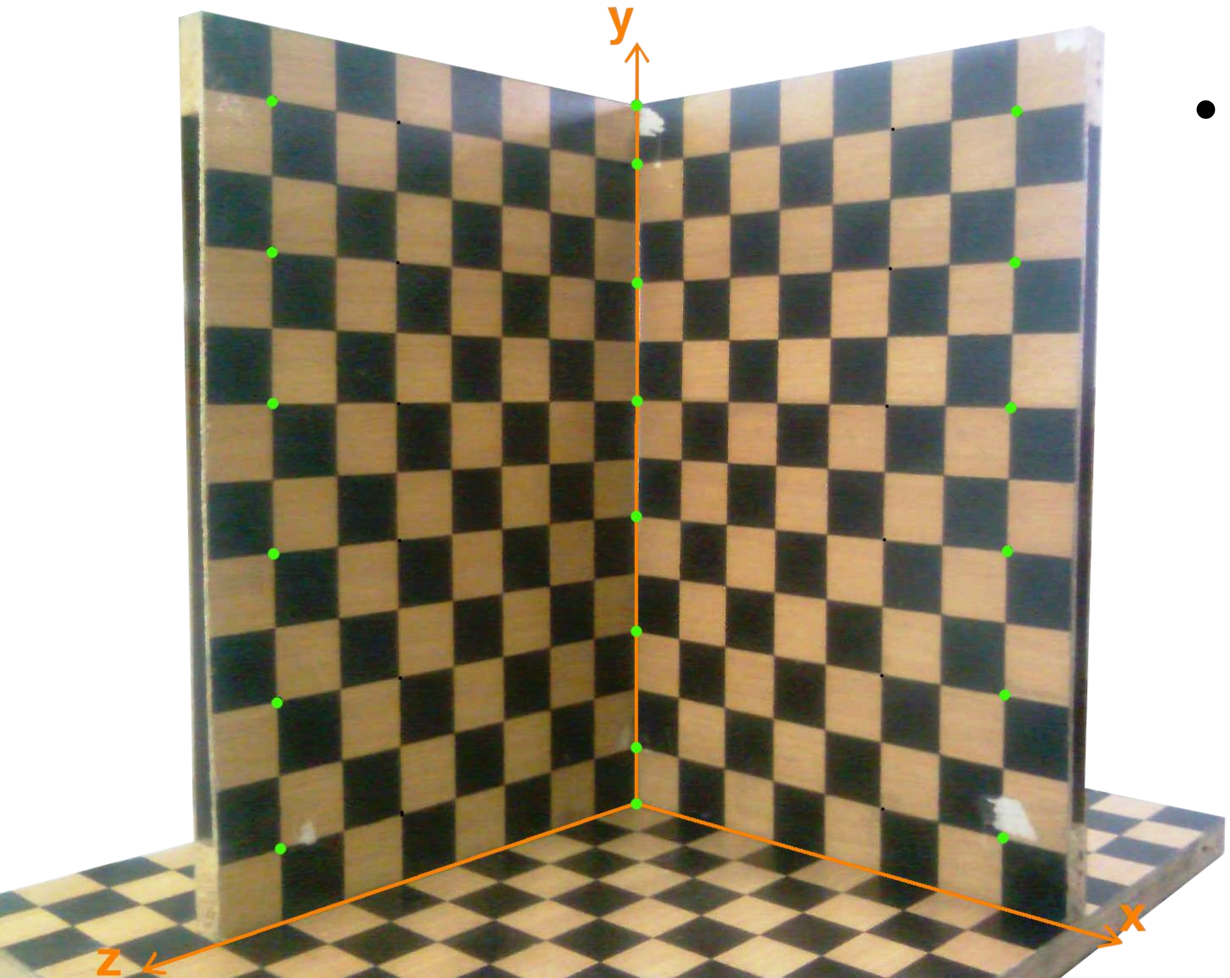
- Then, K can be retrieved by factorization

Camera calibration

- Using such a pattern, calibration can be achieved with a single image

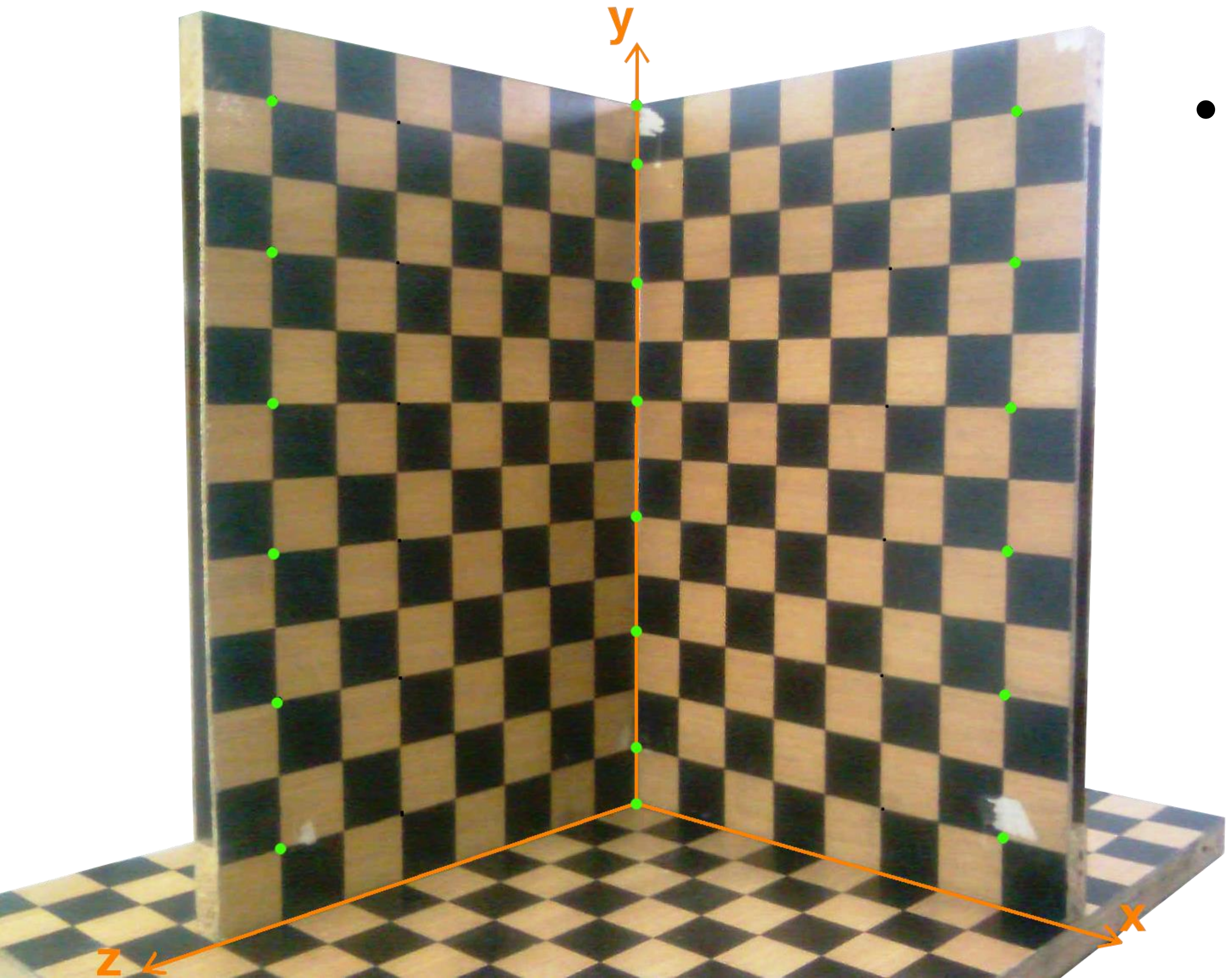


Camera calibration



- Using such a pattern, calibration can be achieved with a single image
- However
  - Corners are more difficult to detect, due to the foreshortening of the three planes

Camera calibration



- Using such a pattern, calibration can be achieved with a single image
- However
  - Corners are more difficult to detect, due to the foreshortening of the three planes
  - Factorization of the full camera matrix can lead to a less accurate estimation of K

Camera calibration

- Other partial calibration solution are
  - Based on the relation between the fundamental and essential matrix
  - Exploiting the vanishing points of three orthogonal directions

Camera calibration from Fundamental matrix

- We know that $\mathrm{E} = \mathrm{K}^{\intercal}\mathrm{F}\mathrm{K}$ and that F can be estimated using image correspondences (8-point algorithm). Then, we can exploit the properties of E

Mendonça and Cipolla. "A simple technique for self-calibration." *CVPR*, 1999.

Camera calibration from Fundamental matrix

- We know that $\mathrm{E} = \mathrm{K}^{\top}\mathrm{F}\mathrm{K}$ and that F can be estimated using image correspondences (8-point algorithm). Then, we can exploit the properties of E

- Given the SVD decomposition of E as

$$\mathrm{SVD}(\mathrm{E}) = \mathrm{U}\mathrm{D}\mathrm{V}^{\top}$$

with

$$\mathrm{D} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$$

where $\sigma_1 = \sigma_2$ and $\sigma_3 = 0$

Mendonça and Cipolla. "A simple technique for self-calibration." *CVPR*, 1999.

Camera calibration from Fundamental matrix

- We know that $\mathrm{E} = \mathrm{K}^\top \mathrm{F} \mathrm{K}$ and that F can be estimated using image correspondences (8-point algorithm). Then, we can exploit the properties of E

- Given the SVD decomposition of E as

$$\mathrm{SVD}(\mathrm{E}) = \mathrm{U}\mathrm{D}\mathrm{V}^\top$$

with

$$\mathrm{D} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$$

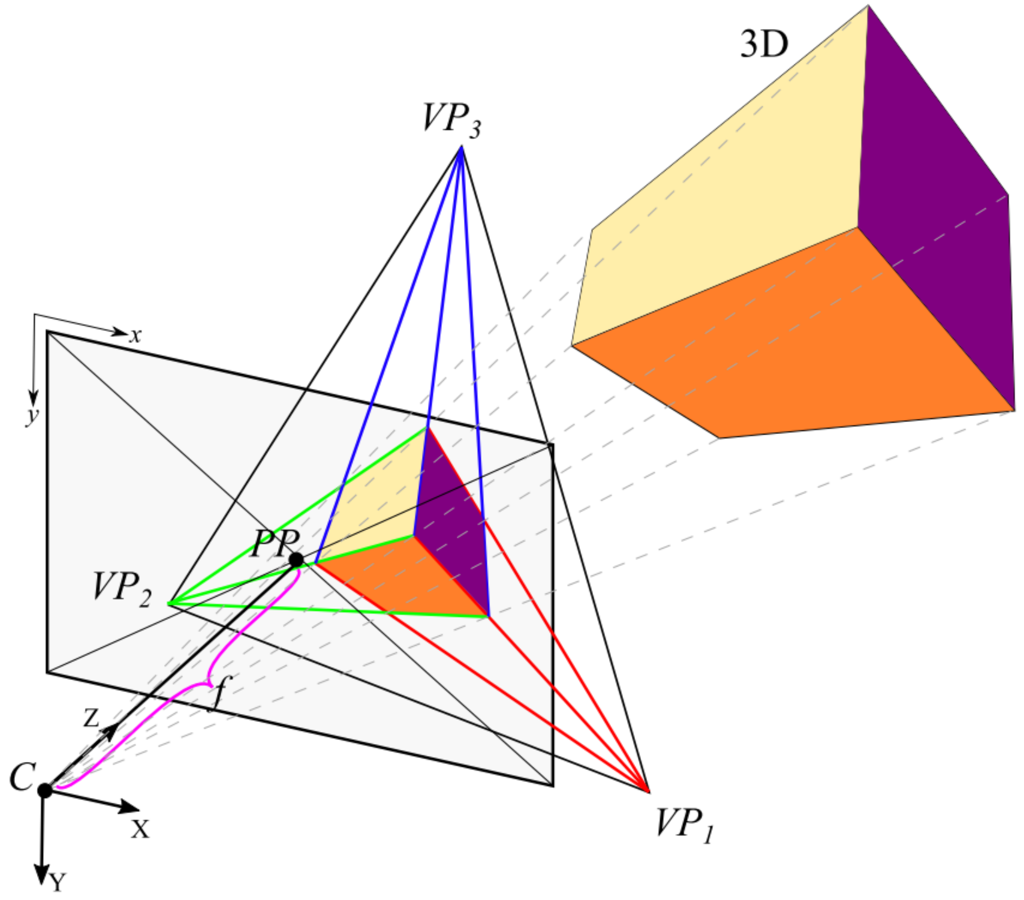
where $\sigma_1 = \sigma_2$ and $\sigma_3 = 0$

- So we can find one of the five DoF of K (for example the focal length), by searching for the K matrix such that the $\mathrm{SVD}(\mathrm{K}^\top \mathrm{F} \mathrm{K})$ produce the first two singular values to be equal

Mendonça and Cipolla. "A simple technique for self-calibration." *CVPR*, 1999.

Camera calibration with Vanishing Points

- Vanishing points are images of 3D directions



Camera calibration with Vanishing Points

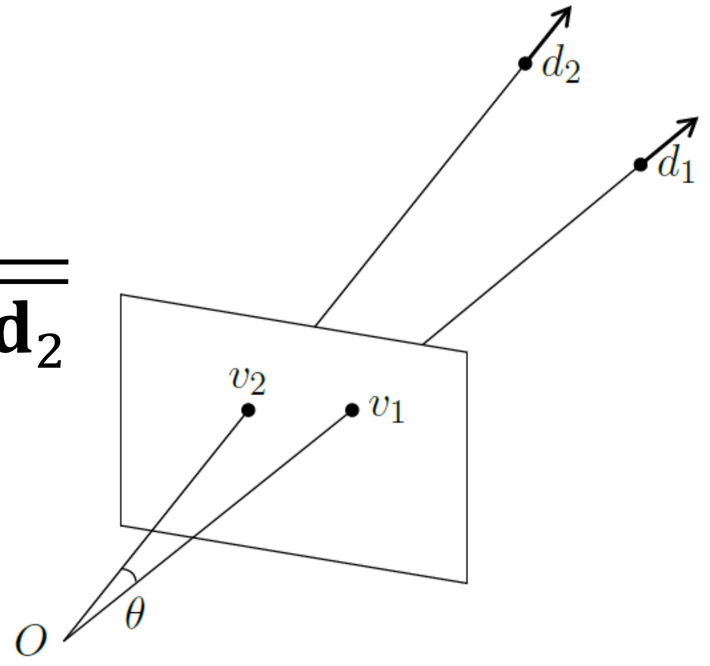
- Vanishing points are images of 3D directions

- A direction can be expressed as a point on the plane at infinity π_∞

Camera calibration with Vanishing Points

- Vanishing points are images of 3D directions
- A direction can be expressed as a point on the plane at infinity π_∞
- Given two directions, the angle between them can be estimated by evaluating their normalized dot product

$$\cos(\theta) = \frac{\mathbf{d}_1}{\|\mathbf{d}_1\|}\frac{\mathbf{d}_2}{\|\mathbf{d}_2\|} = \frac{\mathbf{d}_1}{\sqrt{\mathbf{d}_1^\top\mathbf{d}_1}}\frac{\mathbf{d}_2}{\sqrt{\mathbf{d}_2^\top\mathbf{d}_2}}$$

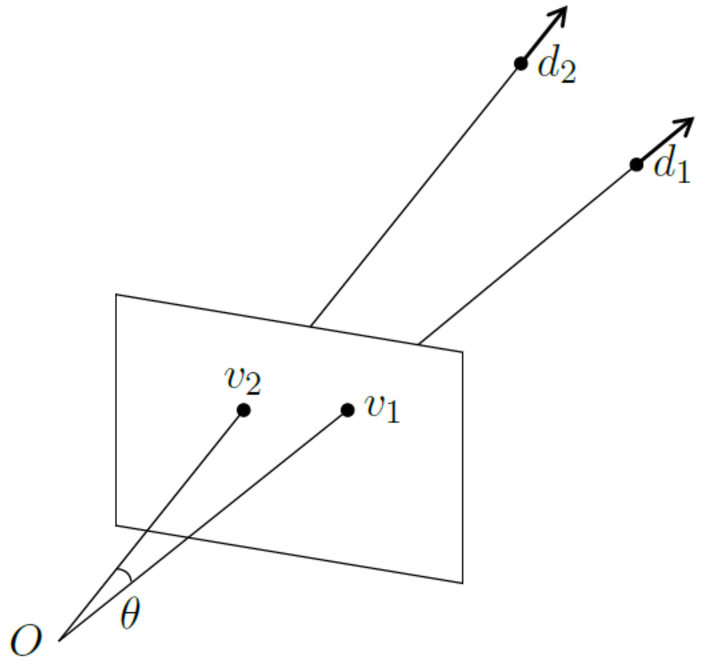


Camera calibration with Vanishing Points

- Since $\mathbf{v}_i = \mathrm{K}\mathbf{d}_1$ and $\mathrm{d}_i = \mathrm{K}^{-1}\mathbf{v}_1$

$$\cos(\theta) = \frac{\mathbf{d}_1}{\|\mathbf{d}_1\|}\frac{\mathbf{d}_2}{\|\mathbf{d}_2\|} = \frac{\mathbf{d}_1^\top \mathbf{d}_2}{\sqrt{\mathbf{d}_1^\top \mathbf{d}_1}\sqrt{\mathbf{d}_2^\top \mathbf{d}_2}} =$$

$$= \frac{\mathbf{v}_1^\top K^{-\top} K^{-1} \mathbf{v}_2}{\sqrt{\mathbf{v}_1^\top K^{-\top} K^{-1} \mathbf{v}_1}\sqrt{\mathbf{v}_2^\top K^{-\top} K^{-1} \mathbf{v}_2}}$$



Camera calibration with Vanishing Points

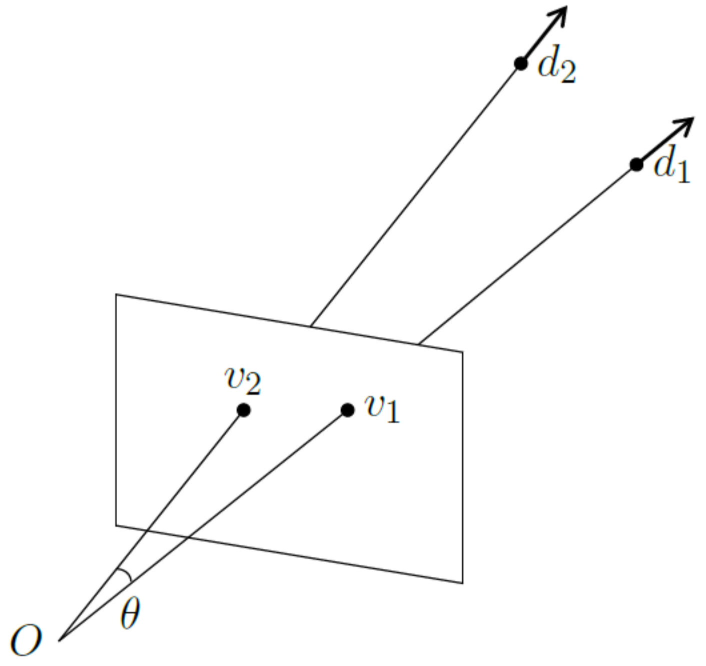
- Since $\mathbf{v}_i = \mathrm{K}\mathbf{d}_1$ and $\mathrm{d}_i = \mathrm{K}^{-1}\mathbf{v}_1$

$$\cos(\theta) = \frac{\mathbf{d}_1}{\|\mathbf{d}_1\|}\frac{\mathbf{d}_2}{\|\mathbf{d}_2\|} = \frac{\mathbf{d}_1^\top \mathbf{d}_2}{\sqrt{\mathbf{d}_1^\top \mathbf{d}_1}\sqrt{\mathbf{d}_2^\top \mathbf{d}_2}} =$$

$$= \frac{\mathbf{v}_1^\top K^{-\top} K^{-1} \mathbf{v}_2}{\sqrt{\mathbf{v}_1^\top K^{-\top} K^{-1} \mathbf{v}_1}\sqrt{\mathbf{v}_2^\top K^{-\top} K^{-1} \mathbf{v}_2}}$$

- Since $\omega = K^{-\top} K^{-1} = (\mathrm{K}\mathrm{K}^\top)^{-1}$

$$\cos(\theta) = \frac{\mathbf{v}_1^\top \omega \mathbf{v}_2}{\sqrt{\mathbf{v}_1^\top \omega \mathbf{v}_1}\sqrt{\mathbf{v}_2^\top \omega \mathbf{v}_2}}$$



Camera calibration with Vanishing Points

- If $\mathbf{v}_1 \perp \mathbf{v}_2$ then $\cos(\theta) = 0$ and we obtain

$$0 = \frac{\mathbf{v}_1^\top \omega \mathbf{v}_2}{\sqrt{\mathbf{v}_1^\top \omega \mathbf{v}_1}\sqrt{\mathbf{v}_2^\top \omega \mathbf{v}_2}} = \mathbf{v}_1^\top \omega \mathbf{v}_2$$

Camera calibration with Vanishing Points

- If $\mathbf{v}_1 \perp \mathbf{v}_2$ then $\cos(\theta) = 0$ and we obtain

$$0 = \frac{\mathbf{v}_1^\top \omega \mathbf{v}_2}{\sqrt{\mathbf{v}_1^\top \omega \mathbf{v}_1}\sqrt{\mathbf{v}_2^\top \omega \mathbf{v}_2}} = \mathbf{v}_1^\top \omega \mathbf{v}_2$$

- So, each pair of orthogonal directions in an image impose a linear constraints on ω

Camera calibration with Vanishing Points

- If $\mathbf{v}_1 \perp \mathbf{v}_2$ then $\cos(\theta) = 0$ and we obtain

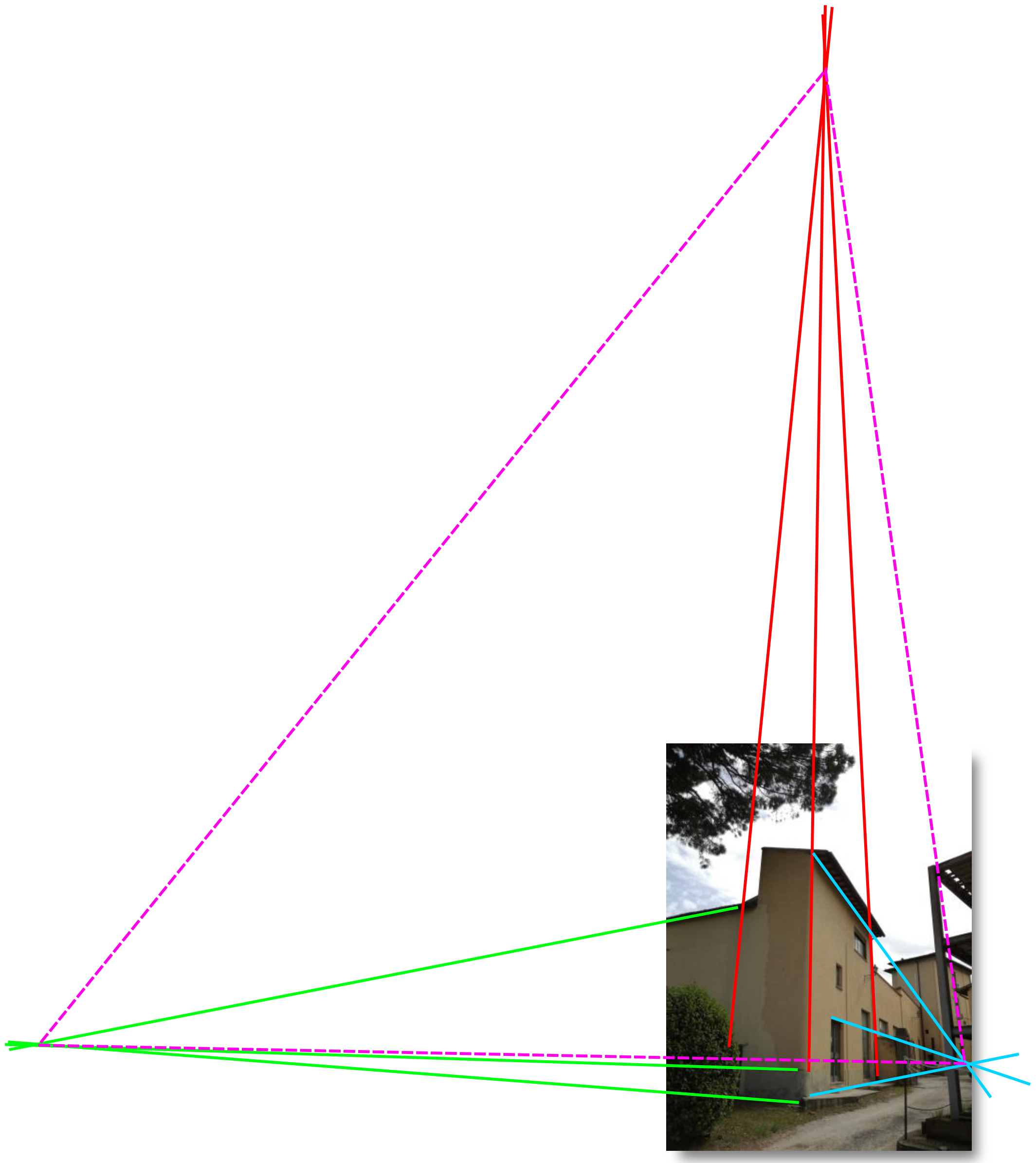
$$0 = \frac{\mathbf{v}_1^\top \omega \mathbf{v}_2}{\sqrt{\mathbf{v}_1^\top \omega \mathbf{v}_1}\sqrt{\mathbf{v}_2^\top \omega \mathbf{v}_2}} = \mathbf{v}_1^\top \omega \mathbf{v}_2$$

- So, each pair of orthogonal directions in an image impose a linear constraints on ω

- With three **mutually orthogonal directions**, we can fix three of five DoF of ω by solving a linear system

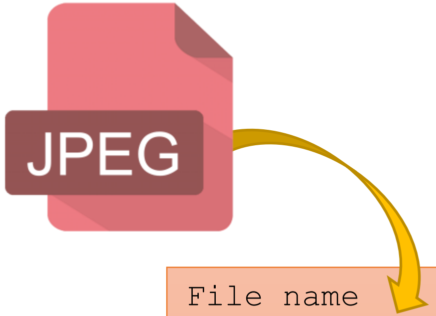
$$\begin{cases} \mathbf{v}_1^\top \omega \mathbf{v}_2 = 0 \\ \mathbf{v}_2^\top \omega \mathbf{v}_3 = 0 \\ \mathbf{v}_3^\top \omega \mathbf{v}_1 = 0 \end{cases}$$

Camera calibration with Vanishing Points



Camera calibration from metadata

- K can be (partially) recovered **image metadata** (*exif* files)

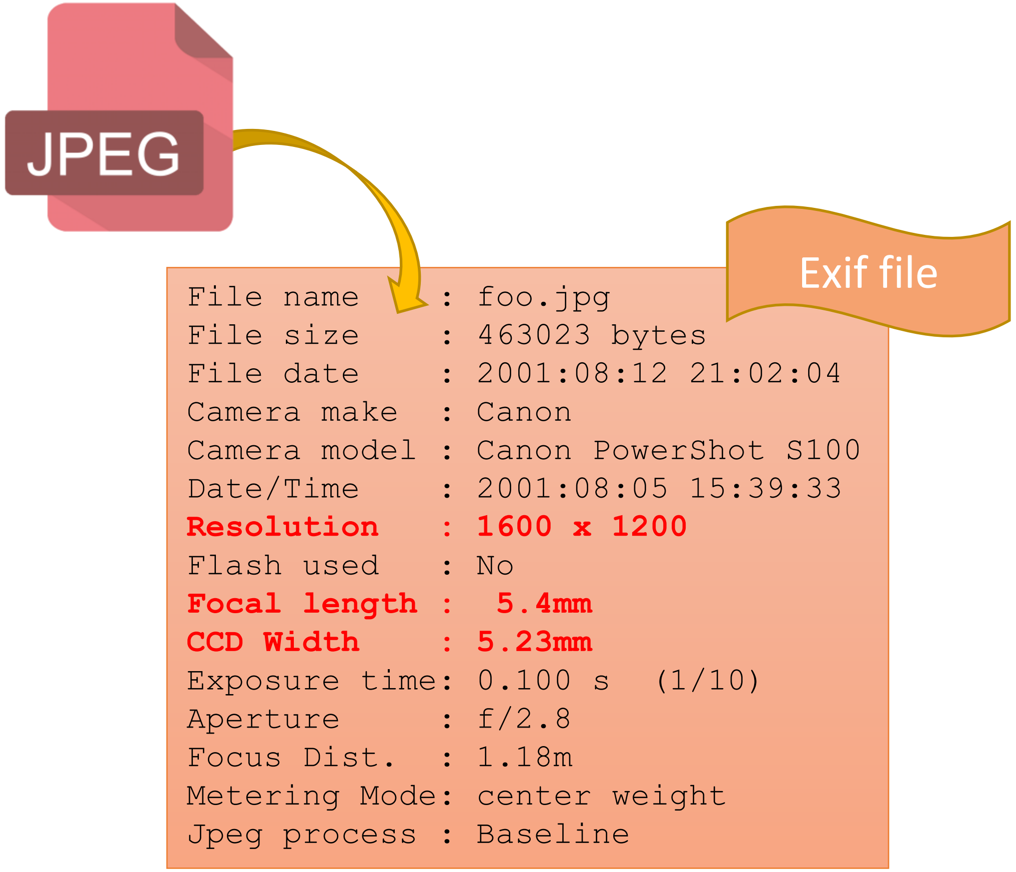


Exif file

```
File name    : foo.jpg
File size    : 463023 bytes
File date    : 2001:08:12 21:02:04
Camera make  : Canon
Camera model : Canon PowerShot S100
Date/Time    : 2001:08:05 15:39:33
Resolution   : 1600 x 1200
Flash used   : No
Focal length :  5.4mm
CCD Width    : 5.23mm
Exposure time: 0.100 s  (1/10)
Aperture     : f/2.8
Focus Dist.  : 1.18m
Metering Mode: center weight
Jpeg process : Baseline
```

Camera calibration from metadata

- K can be (partially) recovered **image metadata** (*exif* files)



Principal point = image resolution / 2

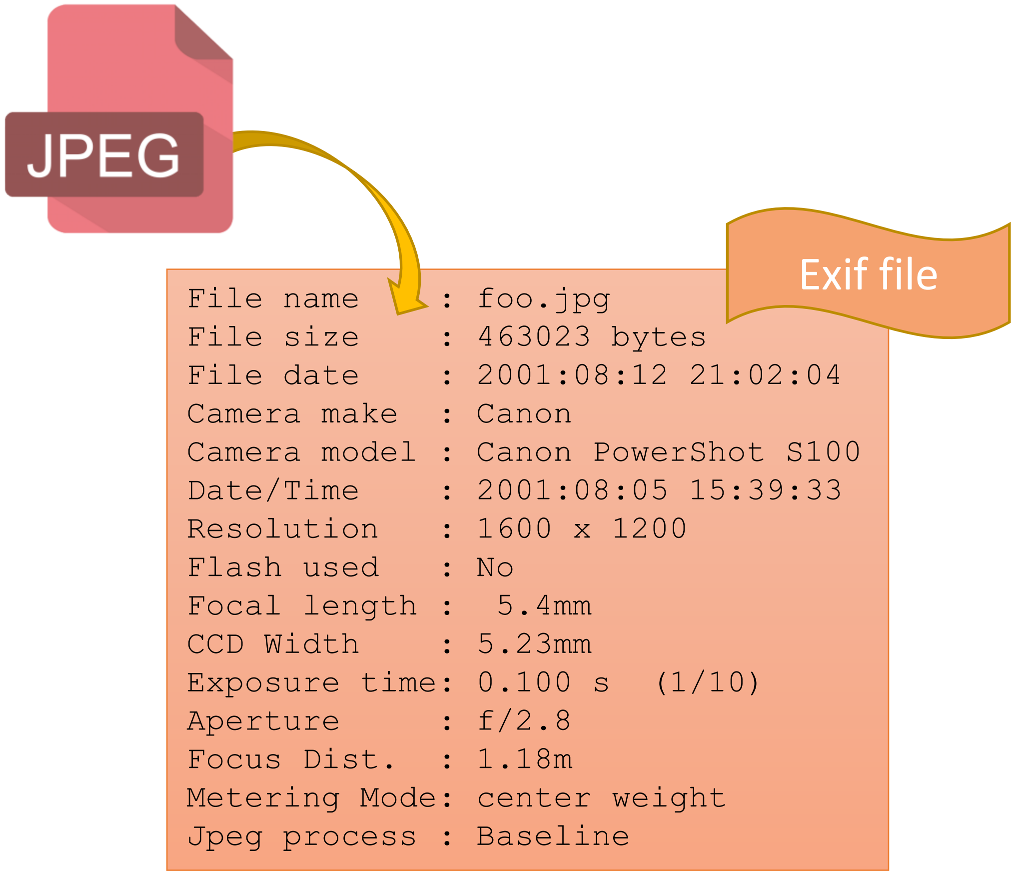
Focal (px) = (image width in pixels) * (focal length in mm) / (CCD width in mm)

See http://phototour.cs.washington.edu/focal.html

Camera calibration from metadata

- K can be (partially) recovered **image metadata** (*exif* files)

- **Global optimization** (e.g., Bundle Adjustment) can be used to refine initial estimates



3D Reconstruction

Stereo

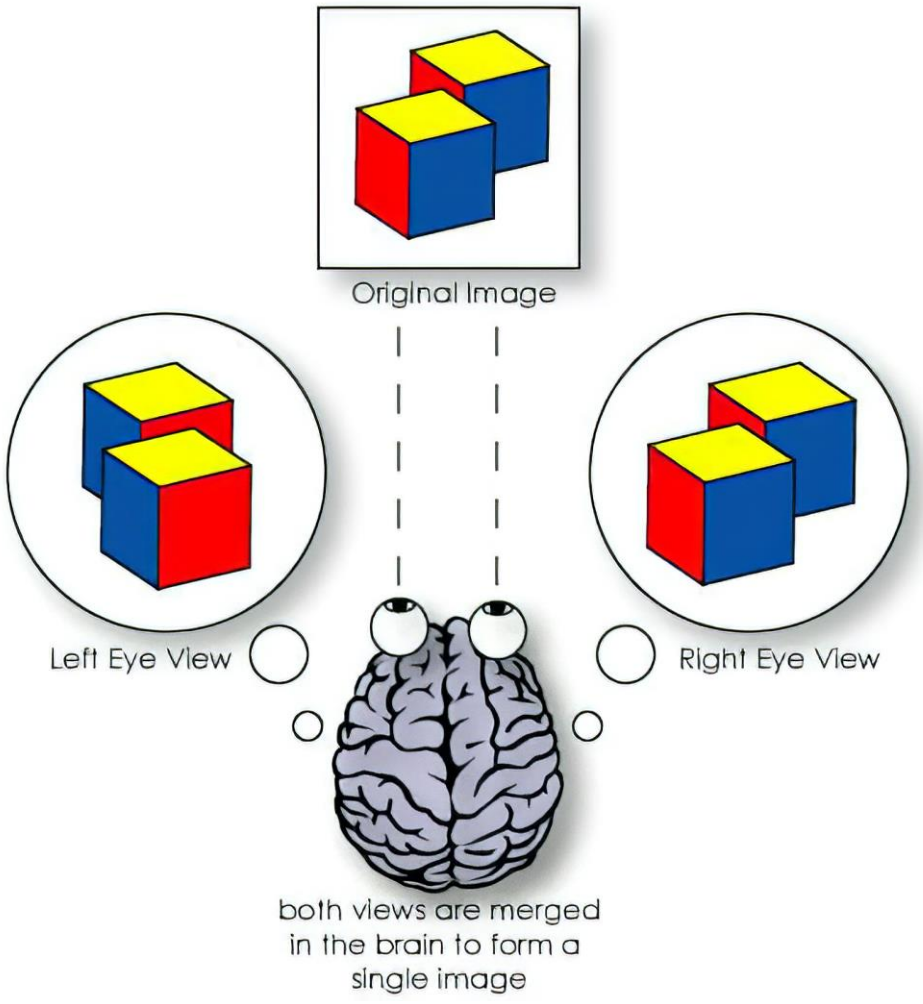
3D Reconstruction

- **Objective**: get back the 3D scene from its images

- Image projection is **not invertible**
  - The image projection, $f: \mathbb{R}^3 \rightarrow \mathbb{R}^2$,is a process that reduce information
  - During projection **point depths are lost**!

- At least **two images** are required to retrieve 3D information

Stereo Vision



Stereo Vision

Stalk-Eyed Fly

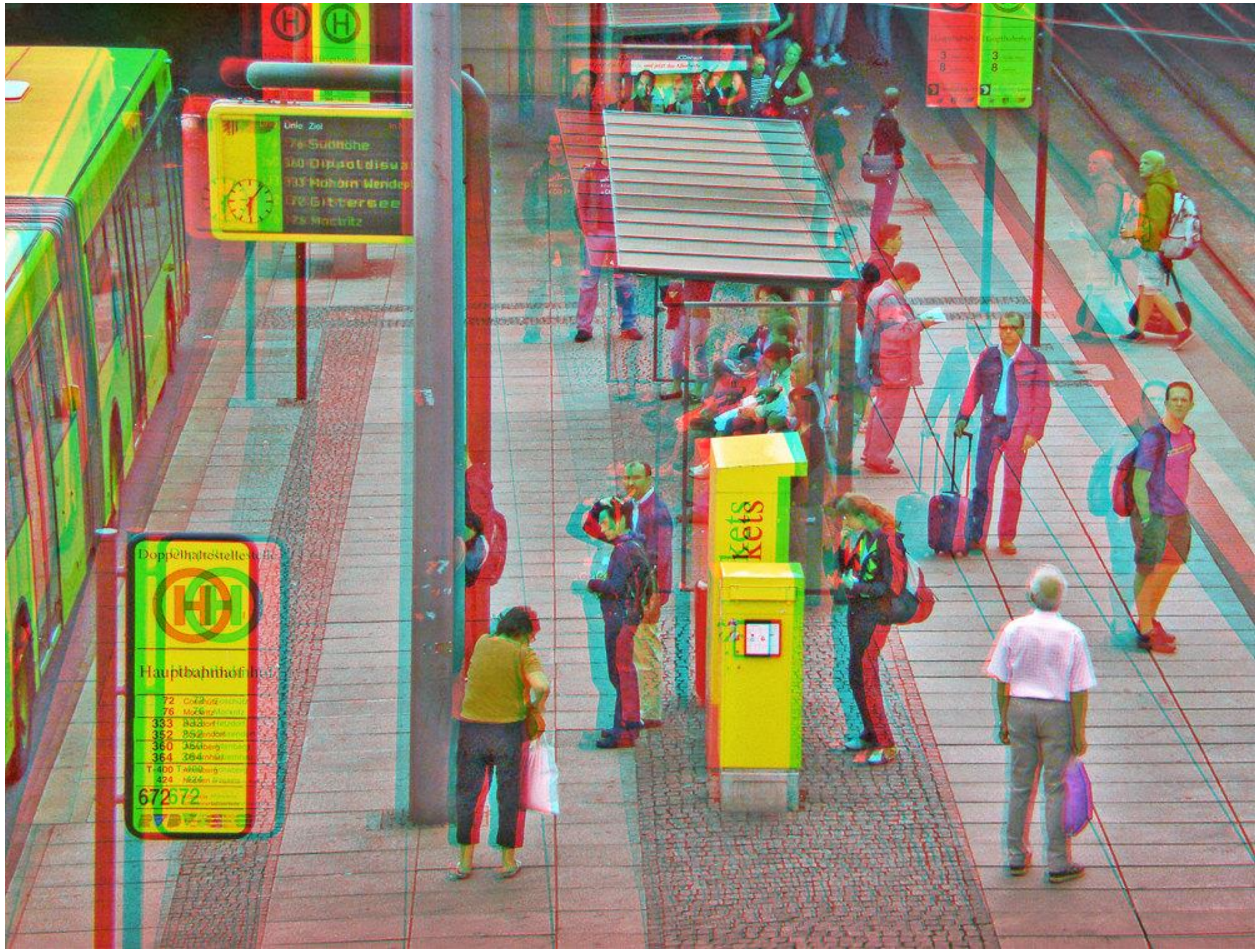
Stereoscopic Rangefinder

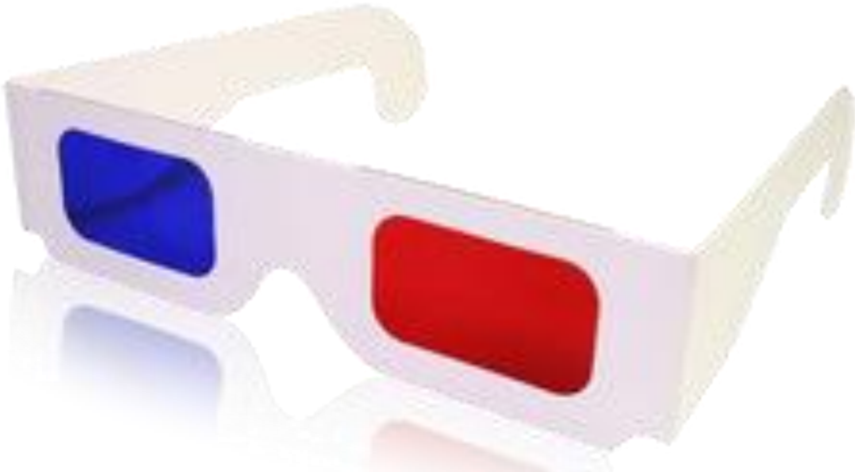
Slide from A. Geiger

Stereo Vision

Florence, XIX century by A. Hautmann

Stereo Vision





Stereo Vision

Stereo Reconstruction


Image A

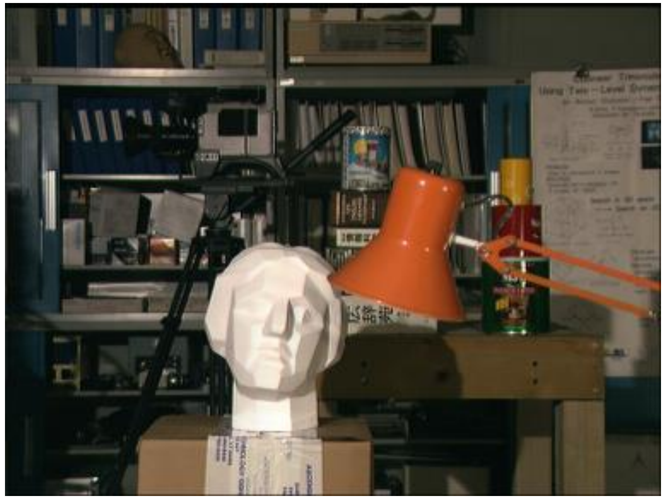
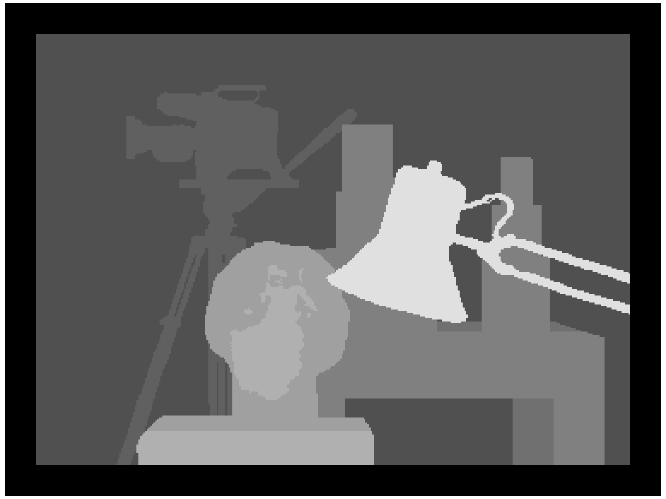


Image B


Disparity map

- **Goal**: given two images of the same scene, compute their **disparity map**
- A disparity map encodes for each pixel its **shift from the first to the second image**
  - Closer points will have higher disparity
  - Further points will have lower (or zero) disparity
- Problem: **occluded points** cannot be estimated!

Stereo Reconstruction

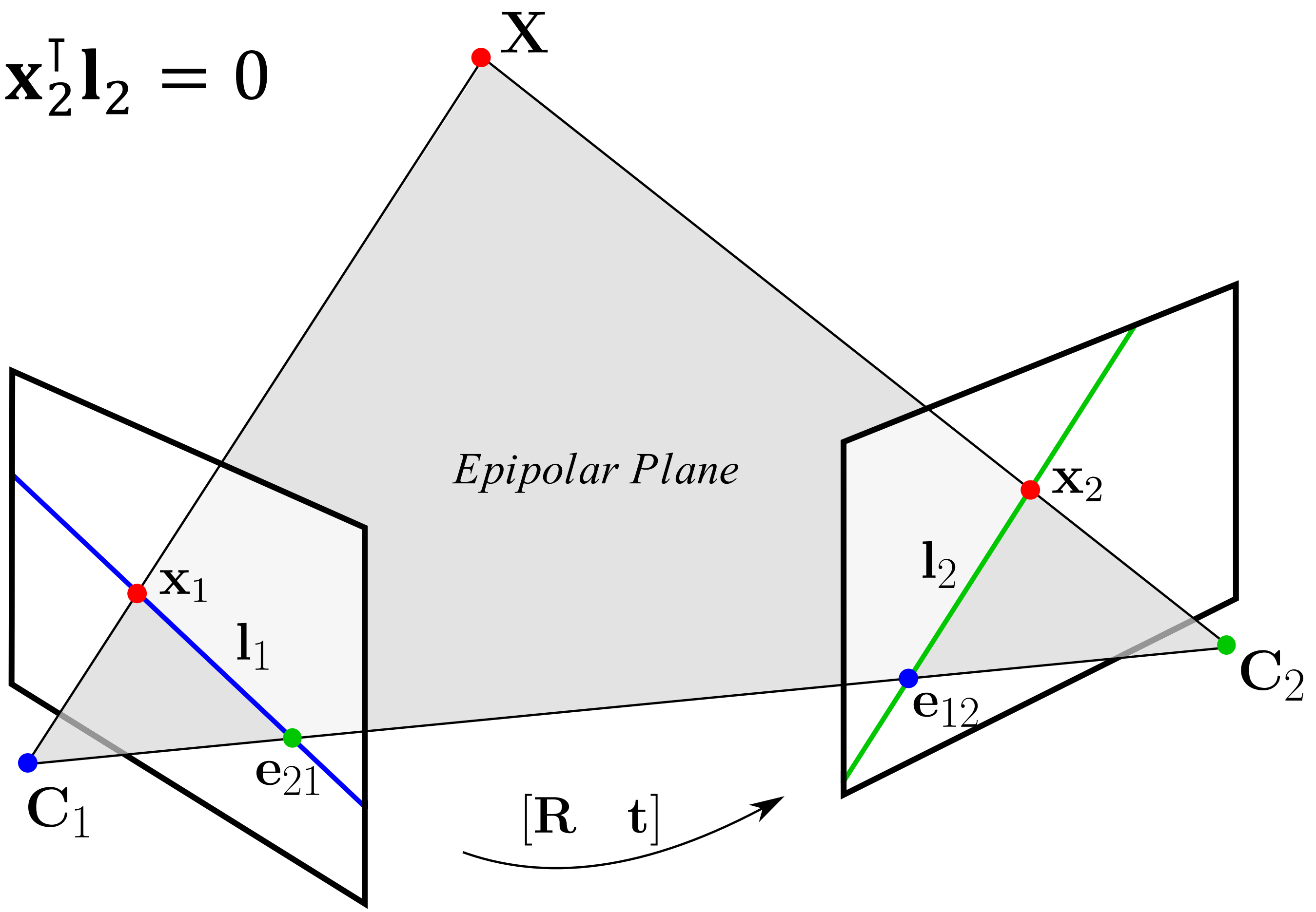
- We know that between two images exist the F matrix such that

$$\mathbf{x}_2^{\top} \mathrm{F} \mathbf{x}_1 = 0$$

and it holds that $\mathrm{F}\mathbf{x}_1 = \mathbf{l}_2$ and $\mathbf{x}_2^{\top}\mathbf{l}_2 = 0$

- So, matching points **lies on the respective epipolar** lines

- To find matching points, we can **limit the search along the epipolar lines**!



Stereo Reconstruction

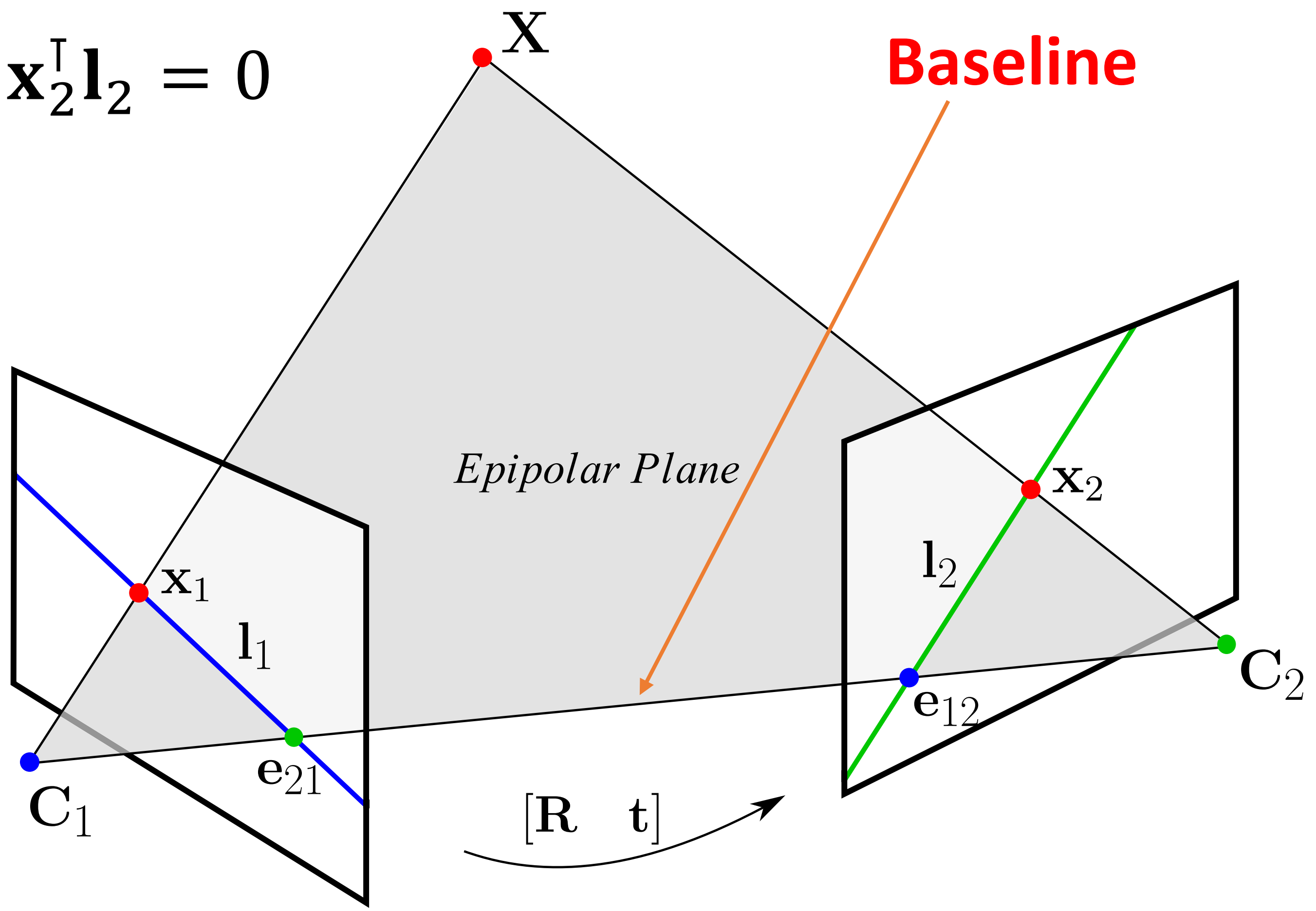
- We know that between two images exist the F matrix such that

$$\mathbf{x}_2^T F \mathbf{x}_1 = 0$$

and it holds that $F\mathbf{x}_1 = \mathbf{l}_2$ and $\mathbf{x}_2^T \mathbf{l}_2 = 0$

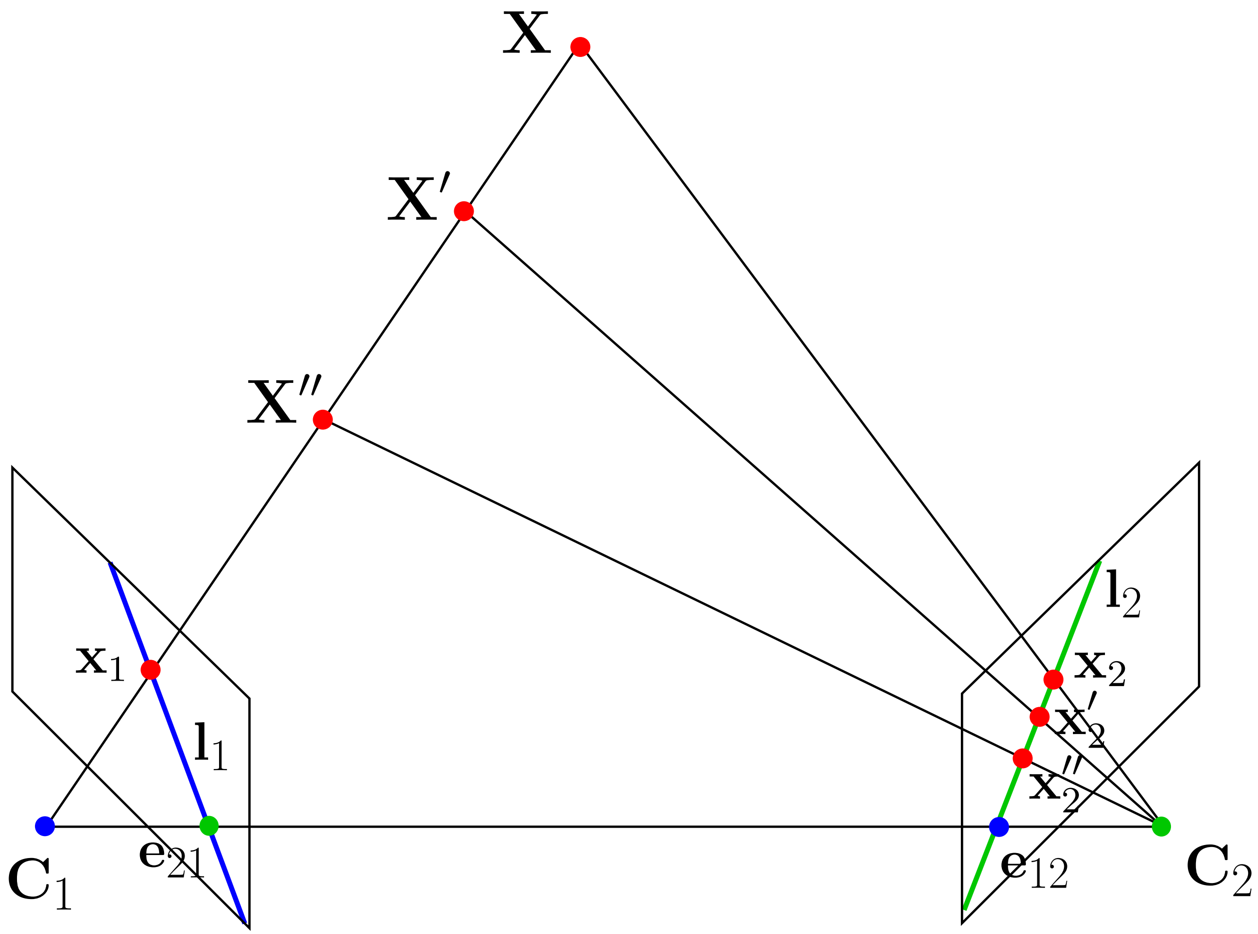
- So, matching points **lies on the respective epipolar** lines

- To find matching points, we can **limit the search along the epipolar lines**!



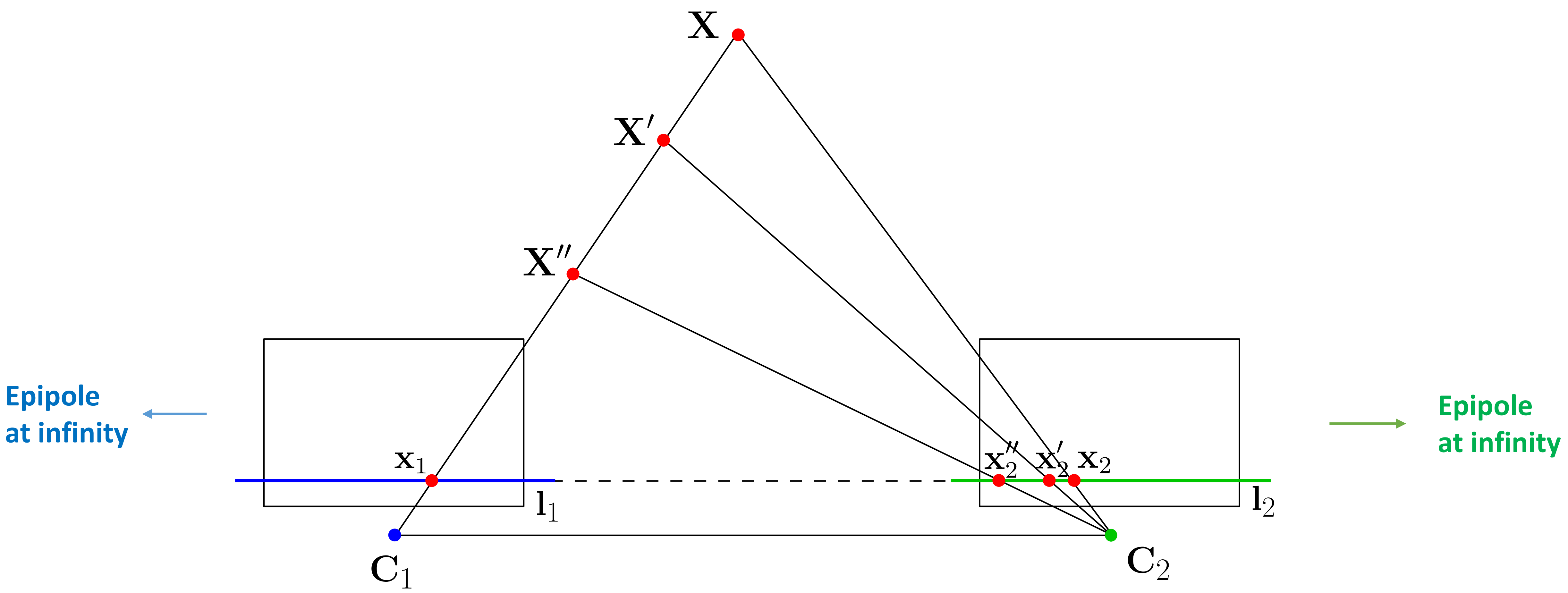
Stereo Reconstruction - Rectification

- To ease the search of matching points, stereo images can be rectified



Stereo Reconstruction - Rectification

- To ease the search of matching points, stereo images can be rectified



Stereo Reconstruction - Rectification

- To rectify a stereo pair:
  1. Compute E, and obtain R and $\mathbf{t}/||\mathbf{t}||$
  2. Define $R_{rect} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3]^\intercal$, where

$$\mathbf{r}_1 = \mathbf{t}/||\mathbf{t}||$$
$$\mathbf{r}_2 = [0 \quad 0 \quad 1]^\intercal \times \mathbf{r}_1$$
$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$

  3. Warp the pixels in the first image as

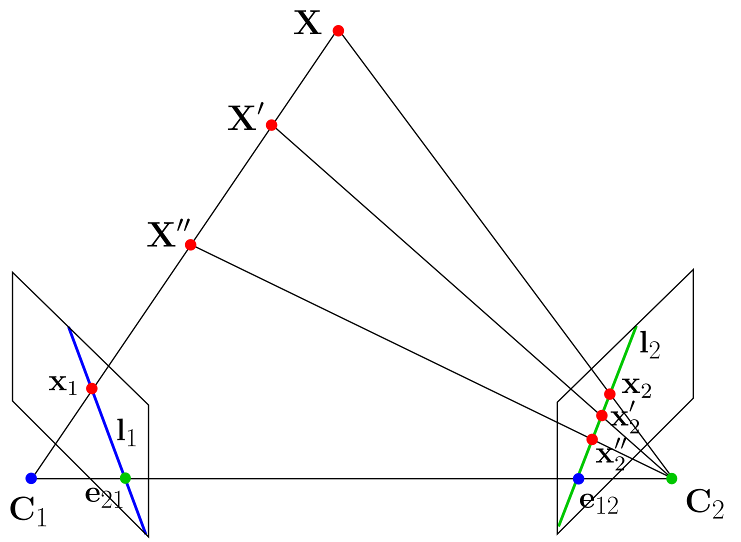
$$\mathbf{x}_1' = \mathrm{K}\mathrm{R}_{rect}\mathrm{K}_1^{-1}\mathbf{x}_1$$

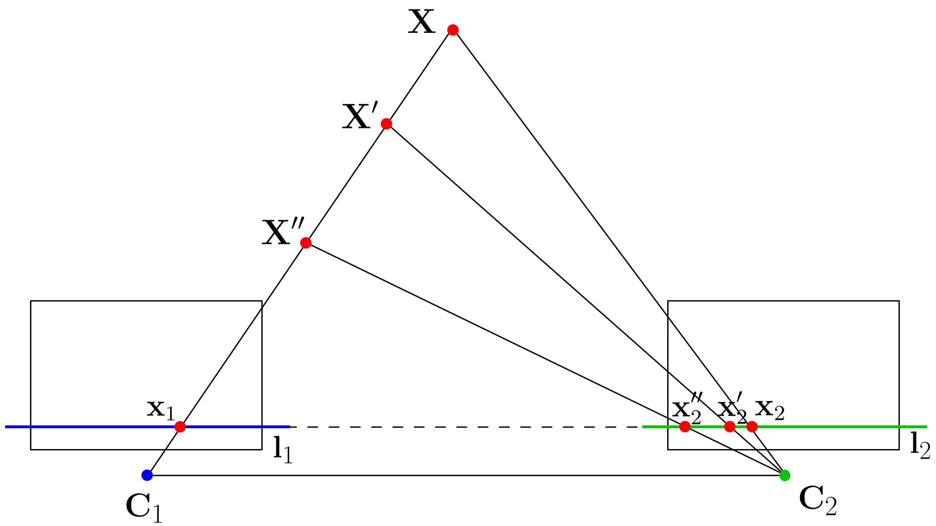
  4. Warp the pixels in the second image as

$$\mathbf{x}_2' = \mathrm{K}\mathrm{R}\mathrm{R}_{rect}\mathrm{K}_2^{-1}\mathbf{x}_2$$

- After rectification, K is the common calibration and

$$\mathrm{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{t} = \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix}$$



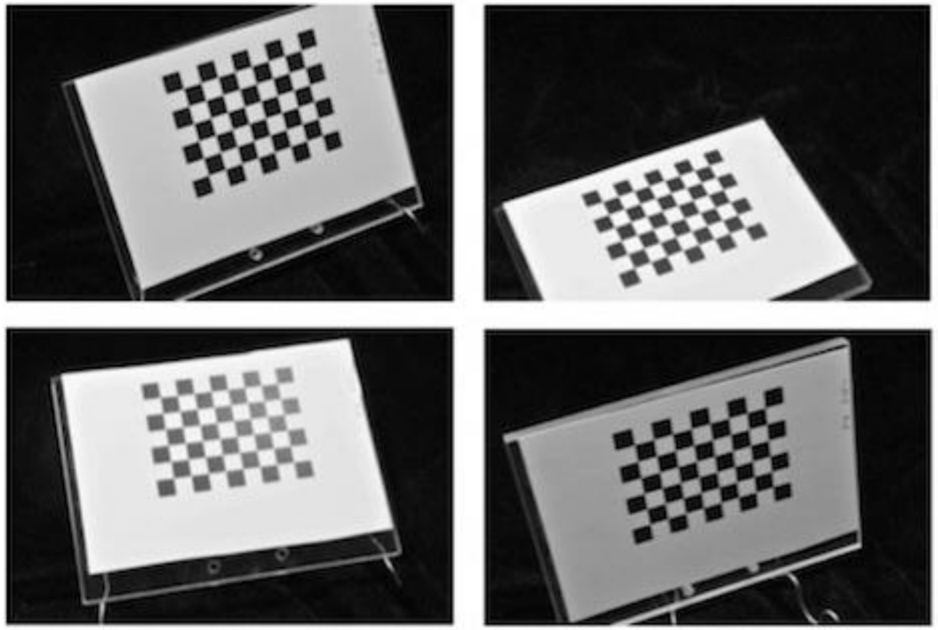


Stereo Reconstruction - Rectification

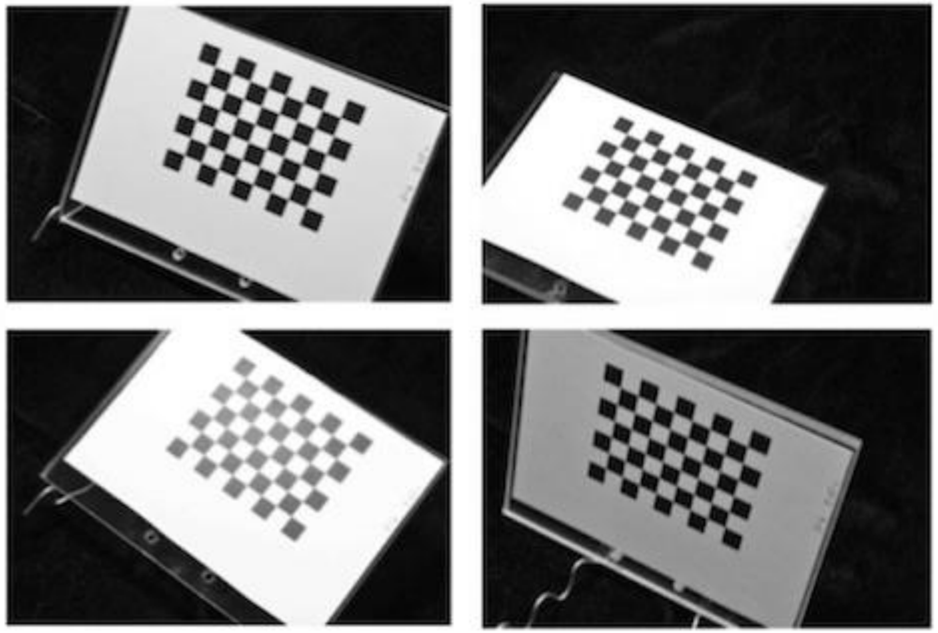
Slide from A. Geiger

Stereo Reconstruction - Rectification

- Stereo rectification can be carried out together with camera calibration
- See for example the `stereoCalibrate` function in OpenCV



Left view



Right view

Stereo Reconstruction - Depth

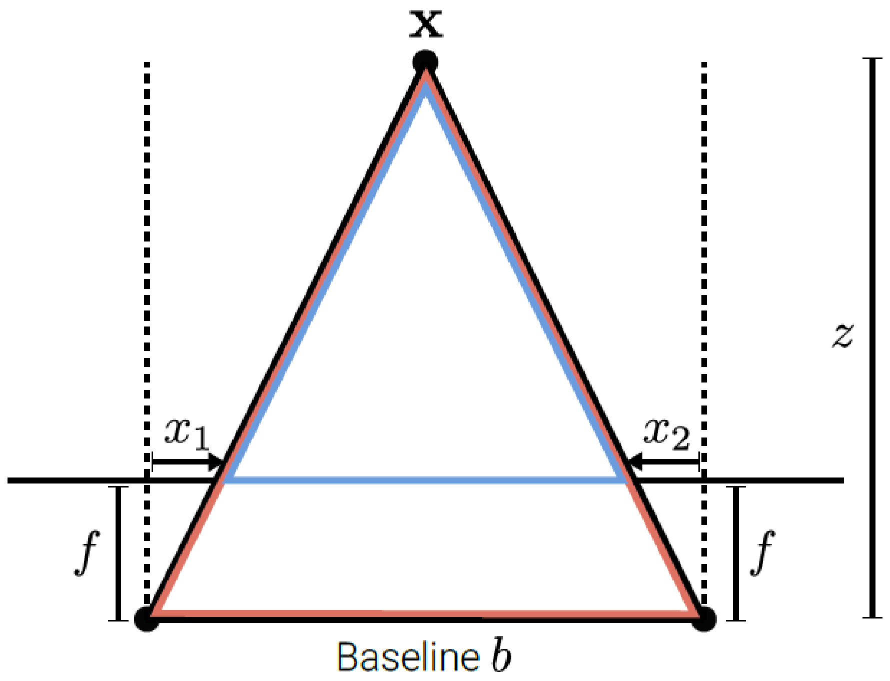
- From the disparity **we can compute the depth** (i.e. the Z coordinate for each point)
- Let $d = x_1 - x_2$, then

$$\frac{Z - f}{b - d} = \frac{Z}{b}$$

$$Zb - fb = Zb - Zd$$

$$Z = \frac{fb}{d}$$

- Note that, as $d \to 0$ we get $Z \to \infty$ and the depth error on Z grows



Stereo Reconstruction – Failure cases

- Perspective distortion

Slide from S. Mattoccia

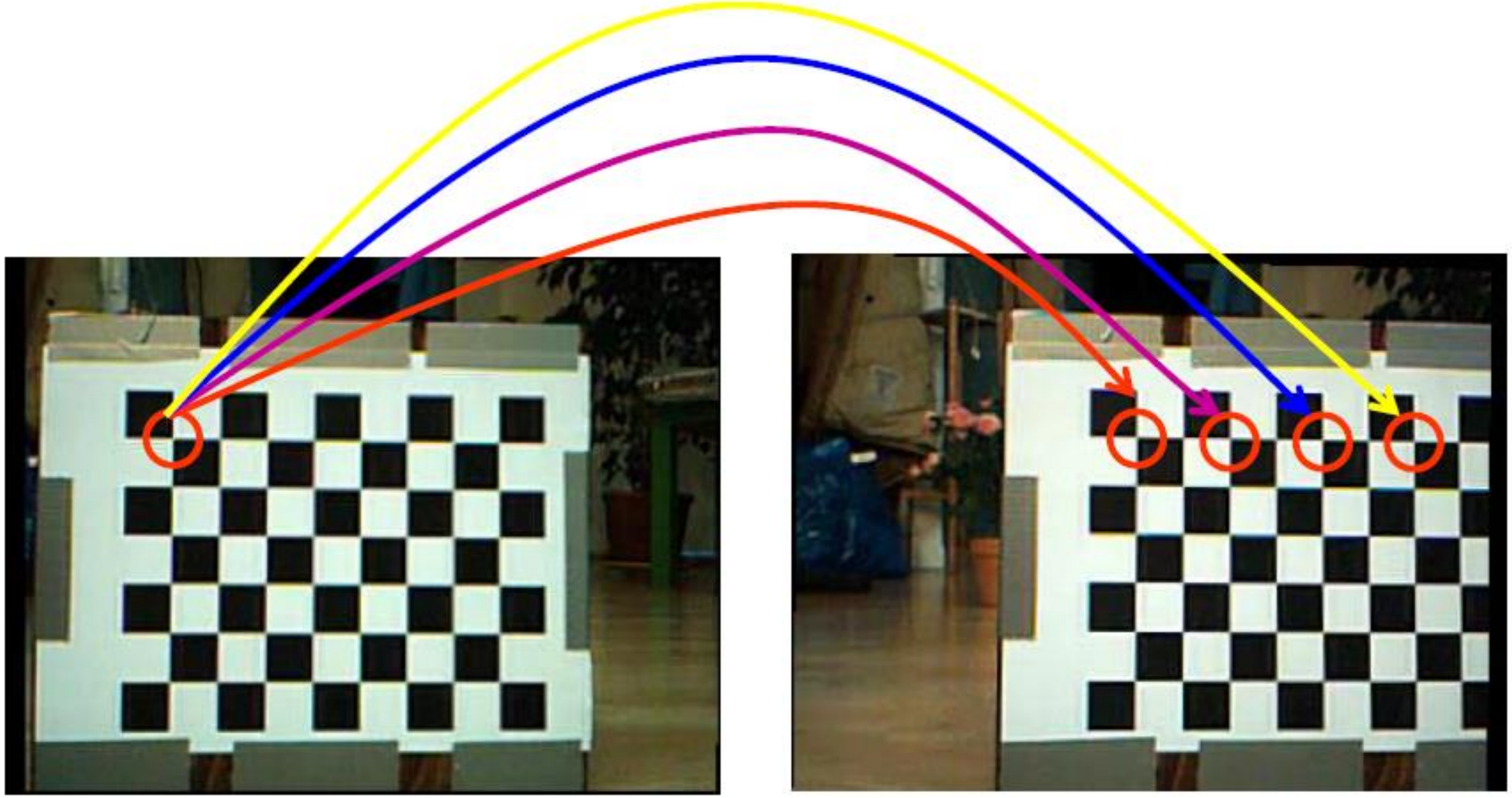
Stereo Reconstruction – Failure cases

- Perspective distortion
- Ambiguous regions

Slide from S. Mattoccia

Stereo Reconstruction – Failure cases

- Perspective distortion
- Ambiguous regions
- Repetitive patterns

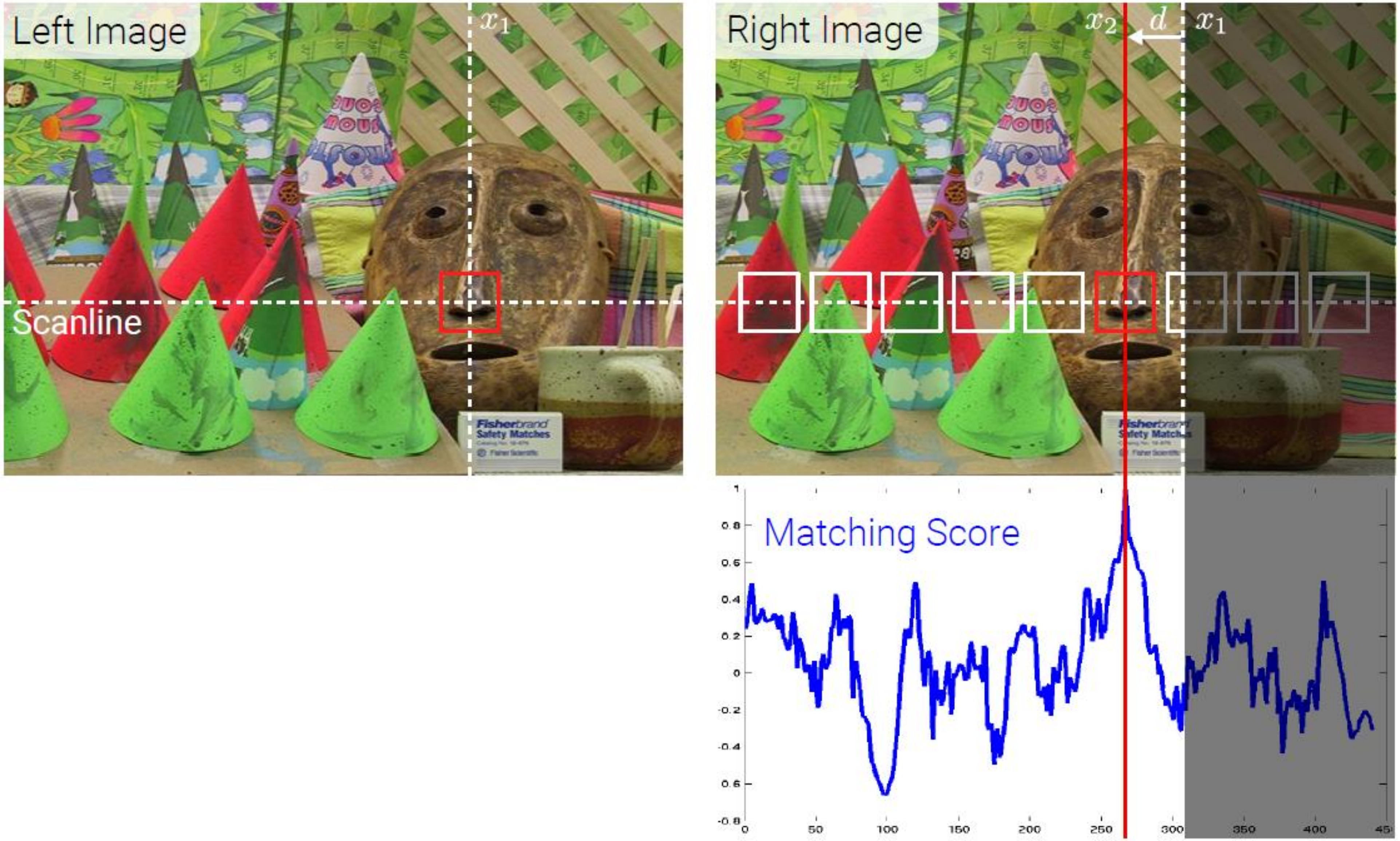


Slide from S. Mattoccia

Stereo Reconstruction – Failure cases

- Perspective distortion
- Ambiguous regions
- Repetitive patterns
- Specular surfaces

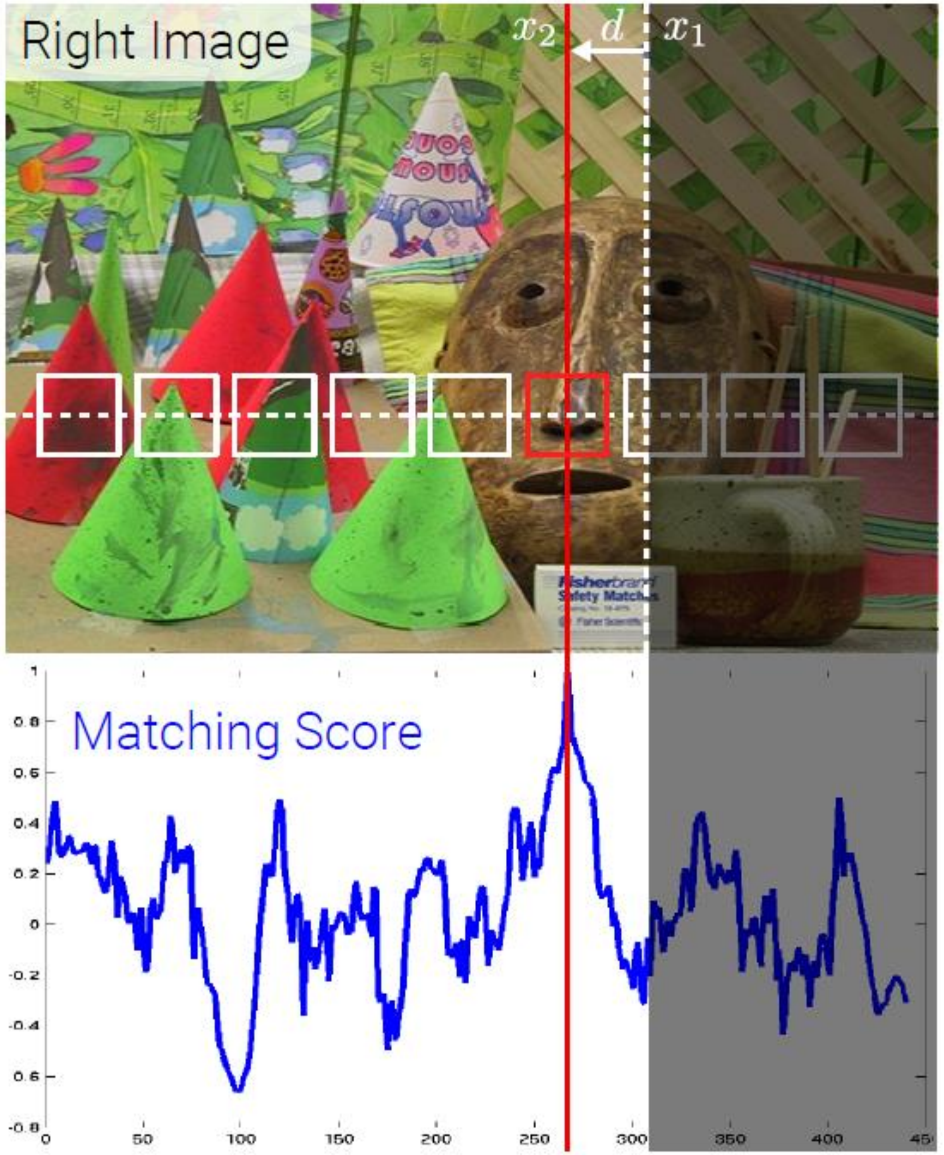
Stereo Reconstruction – Block Matching



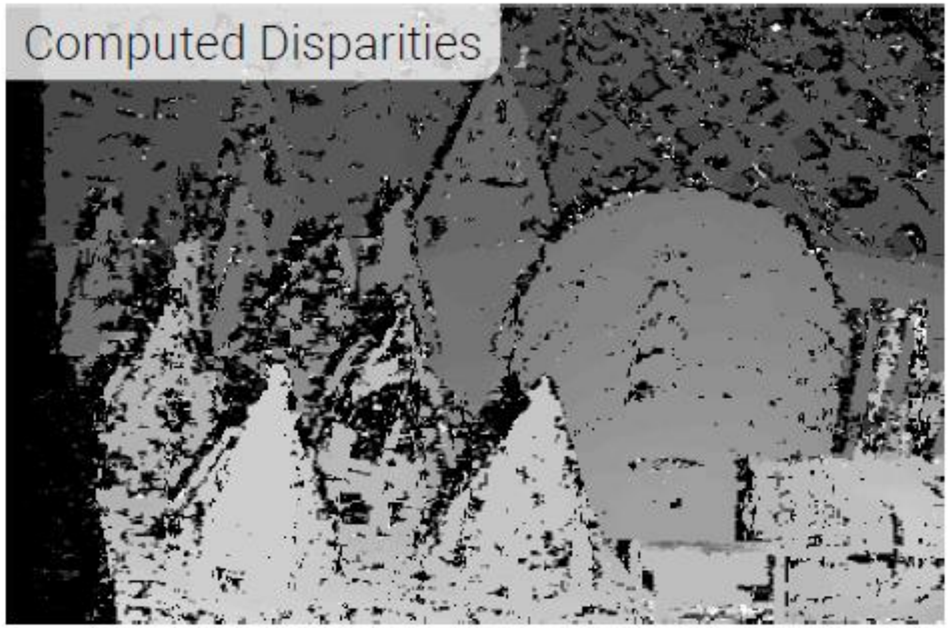
Slide from A. Geiger

Stereo Reconstruction – Block Matching

- Several metrics can be used for such a task:
  - Normalized cross correlation
  - Sum of squared distances
  - Sum of absolute difference
  - …

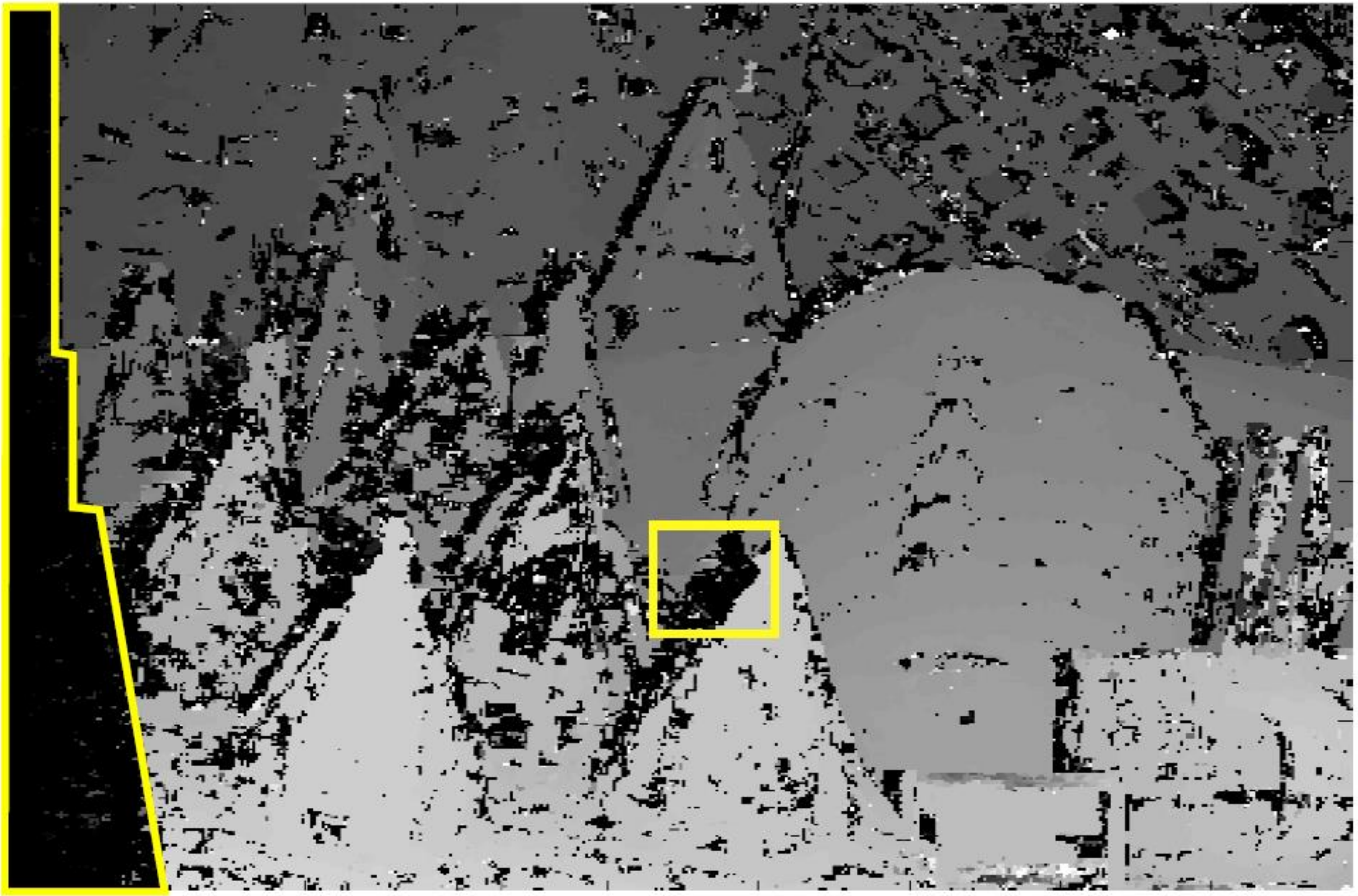


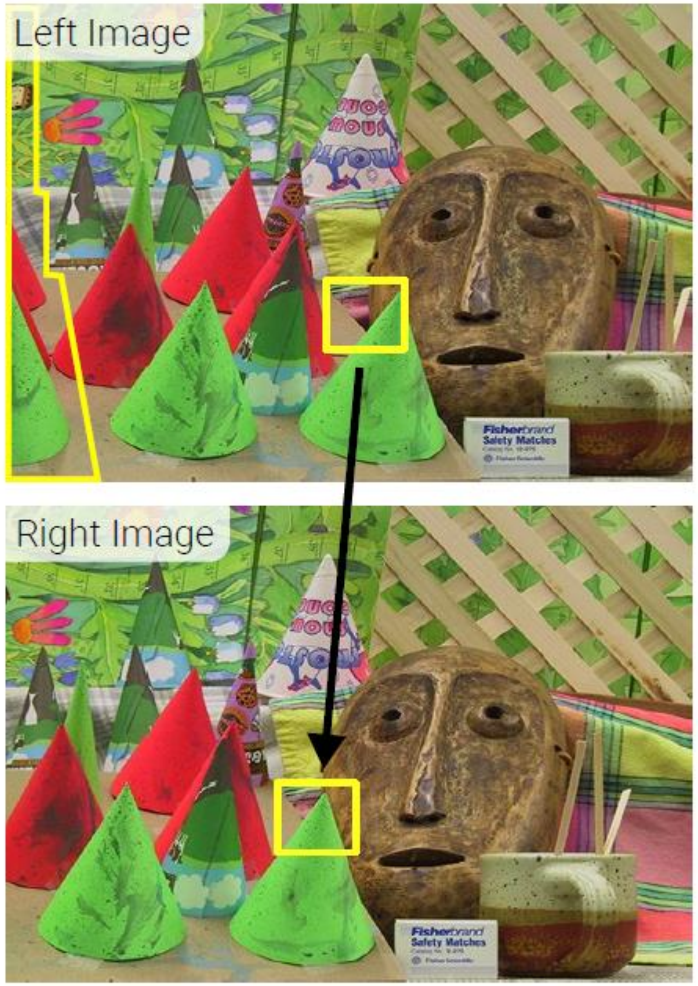
Stereo Reconstruction – Block Matching



Slide from A. Geiger

Stereo Reconstruction – Occlusions

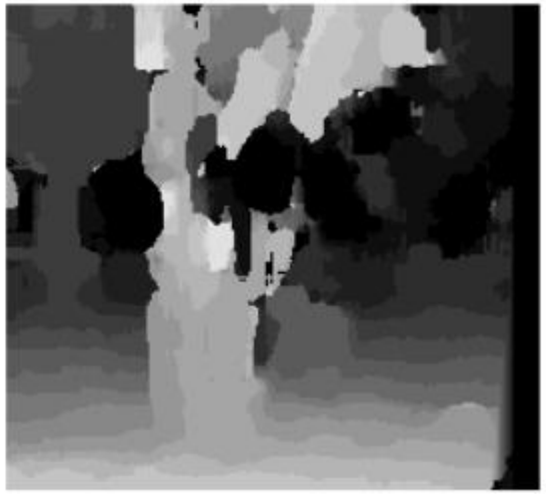




Slide from A. Geiger

Stereo Reconstruction – Window size

W = 3



W = 20

- Smaller window
  - + More detail
  - – More noise
- Larger window
  - + Smoother disparity maps
  - – Less detail

Slide from L. Lazebnik

Scanline optimization

- By solving stereo using Block Matching, we minimize

$$E(D) = \sum_{\mathbf{x} \in I_l} \Delta(\mathbf{x}, D_{\mathbf{x}})$$

Scanline optimization

- By solving stereo using Block Matching, we minimize

$$E(D) = \sum_{\mathbf{x} \in I_l} \Delta(\mathbf{x}, D_{\mathbf{x}})$$

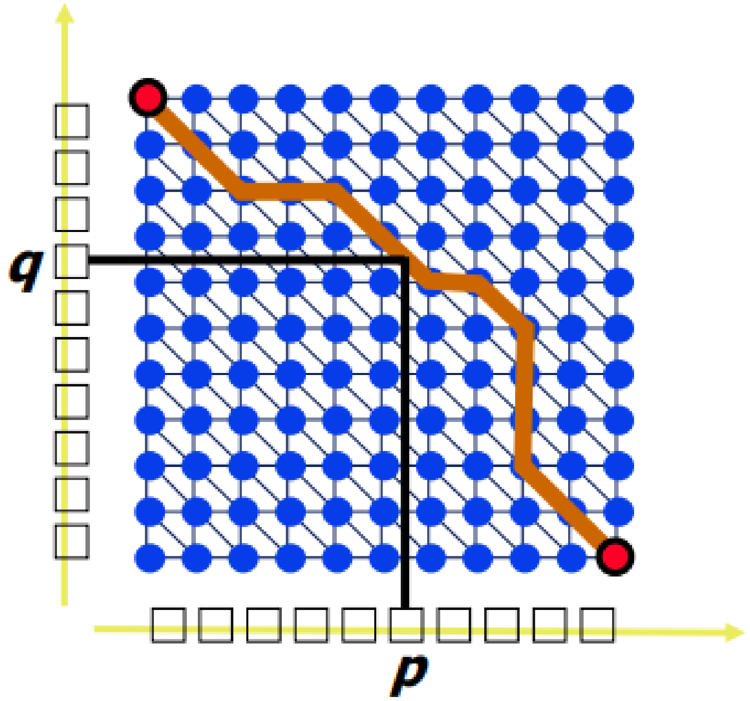
- Since depth is generally smooth, we can add a regularization term

$$E(D) = \sum_{\mathbf{x} \in I_l} \Delta(\mathbf{x}, D_{\mathbf{x}}) + \sum_{x_i, x_j \in N} R(D_{x_i}, D_{x_j})$$

$$R\left(D_{x_i}, D_{x_j}\right) = \begin{cases} 0 & D_{x_i} = D_{x_j} \\ P_1 & \left|D_{x_i} - D_{x_j}\right| = 1 \\ P_2 & \left|D_{x_i} - D_{x_j}\right| > 1 \end{cases}$$

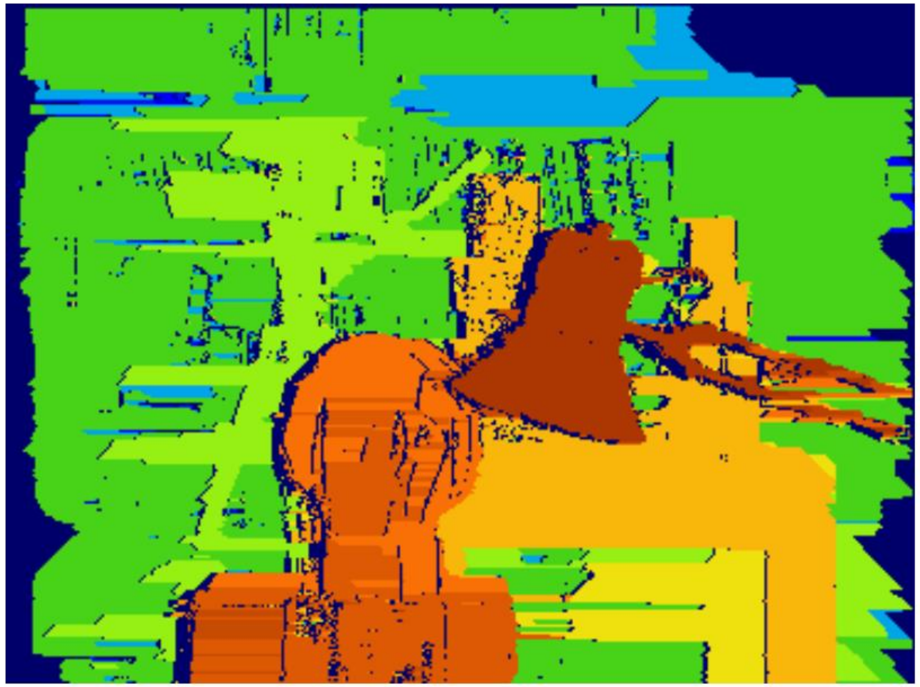
with $P_1 < P_2$

Scanline optimization



To solve the optimization on a **single scanline** we can search for the **shortest path** in the **dissimilarity matrix**, where each row represent $\Delta(\mathbf{x}, D_{\mathbf{x}})$ for a pixel $\mathbf{x}$

Scanline optimization

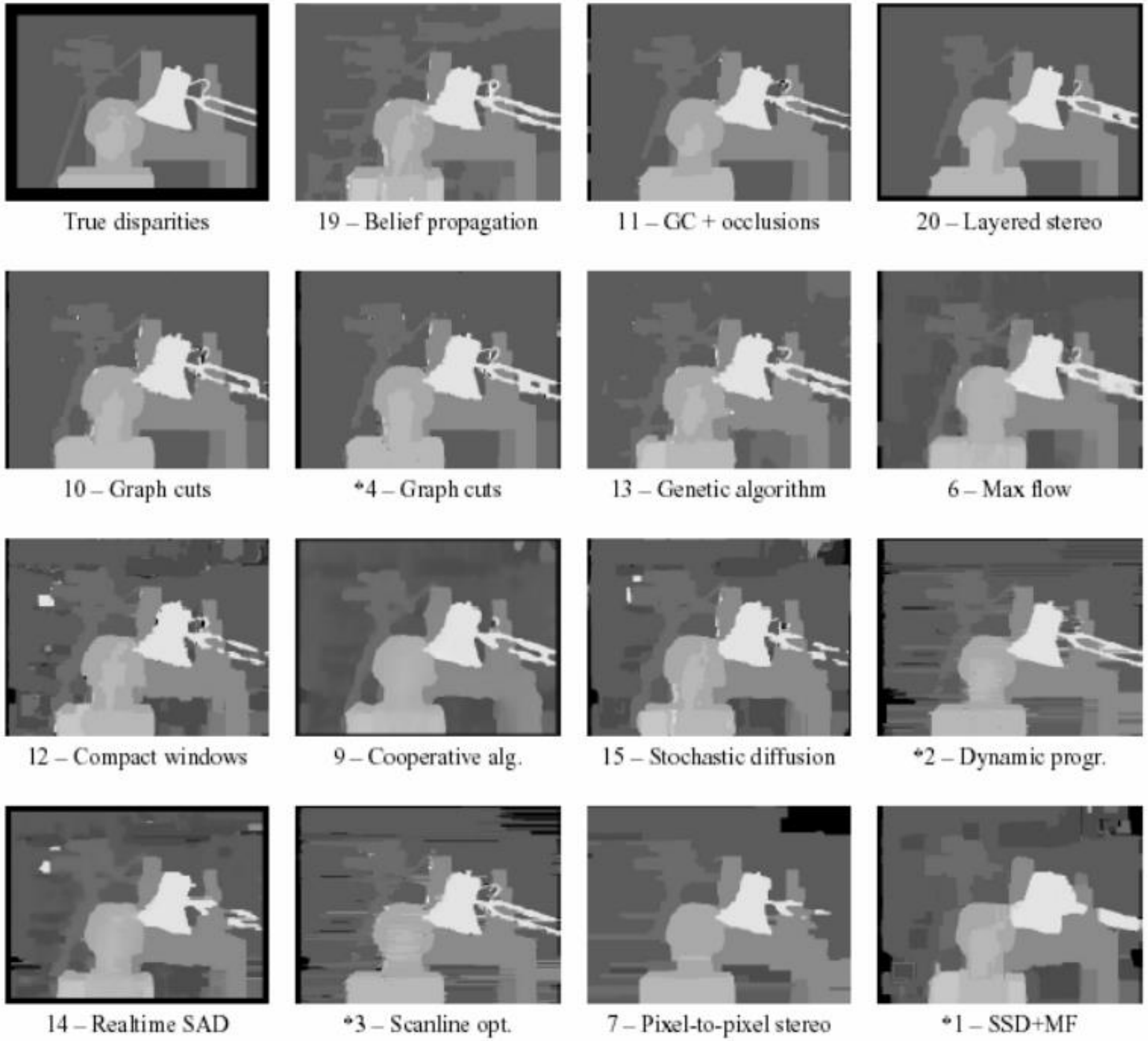


This however introduce **streaking artifacts**

Semi-Global Matching (SGM)

By expanding the regularization on **different directions**, the disparity can be improved

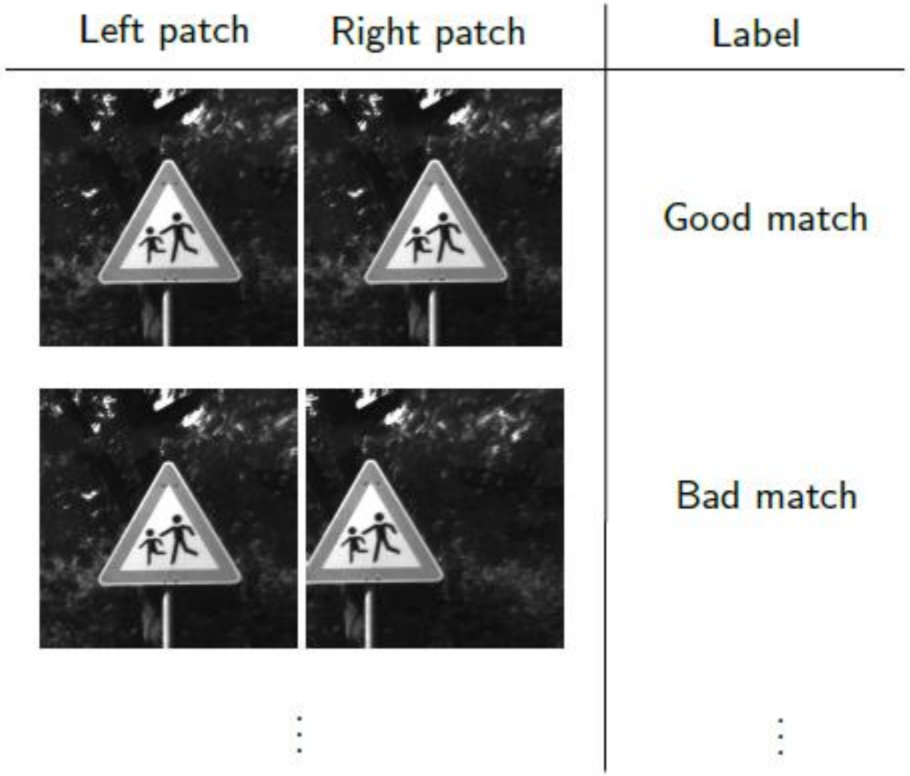
Stereo Matching



Scharstein and Szeliski

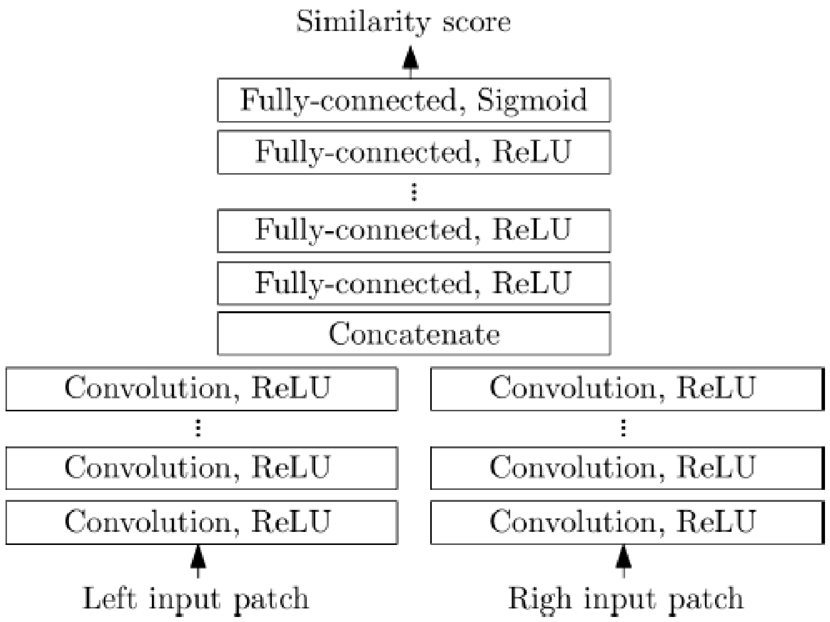
Siamese network

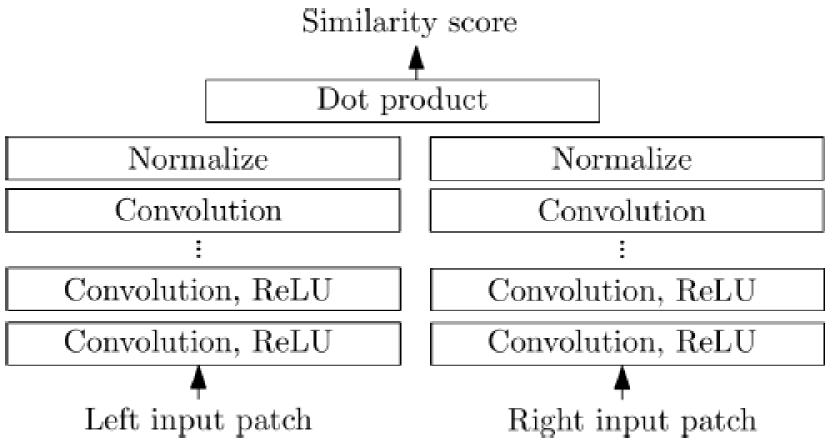
- The matching problem is cast to a sort of classification problem
- The net is trained to identify matching and non-matching patches
- The net output is a patch similarity score



Zbontar and LeCun “Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches”, JMLR, 2016.

Siamese network





Accurate architecture

- Learn feature and similarity metric

Fast architecture

- Learn feature and eval dot product

Zbontar and LeCun "Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches", JMLR, 2016.

Siamese network

Training set composed by positive and negative examples

$$\left(w_l\left(\mathbf{x}_l^{ref}\right), w_r\left(\mathbf{x}_r^{pos}\right)\right) \text{ and } \left(w_l\left(\mathbf{x}_l^{ref}\right), w_r\left(\mathbf{x}_r^{neg}\right)\right)$$

Zbontar and LeCun "Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches", JMLR, 2016.

Siamese network

Training set composed by positive and negative examples

$$\left(w_l\left(\mathbf{x}_l^{ref}\right), w_r\left(\mathbf{x}_r^{pos}\right)\right) \text{ and } \left(w_l\left(\mathbf{x}_l^{ref}\right), w_r\left(\mathbf{x}_r^{neg}\right)\right)$$

- $\mathbf{x}_r^{pos} = \left(x_l^{ref} - d + o_{pos}, y_l^{ref}\right)$, where o_{pos} sampled in $[-P, \dots, P]$
- $\mathbf{x}_r^{neg} = \left(x_l^{ref} - d + o_{neg}, y_l^{ref}\right)$, where o_{neg} sampled in $[-N_h, \dots, -N_l, N_l, \dots, N_h]$
- $P = 1$, while $N_l = 3$ and $N_h = 6$

Zbontar and LeCun “Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches”, JMLR, 2016.

Siamese network

Training set composed by positive and negative examples

$$\left(w_l\left(\mathbf{x}_l^{ref}\right), w_r\left(\mathbf{x}_r^{pos}\right)\right) \text{ and } \left(w_l\left(\mathbf{x}_l^{ref}\right), w_r\left(\mathbf{x}_r^{neg}\right)\right)$$

- $\mathbf{x}_r^{pos} = \left(x_l^{ref} - \boxed{d} + o_{pos}, y_l^{ref}\right)$, where o_{pos} sampled in $[-P, \dots, P]$

Ground truth disparity

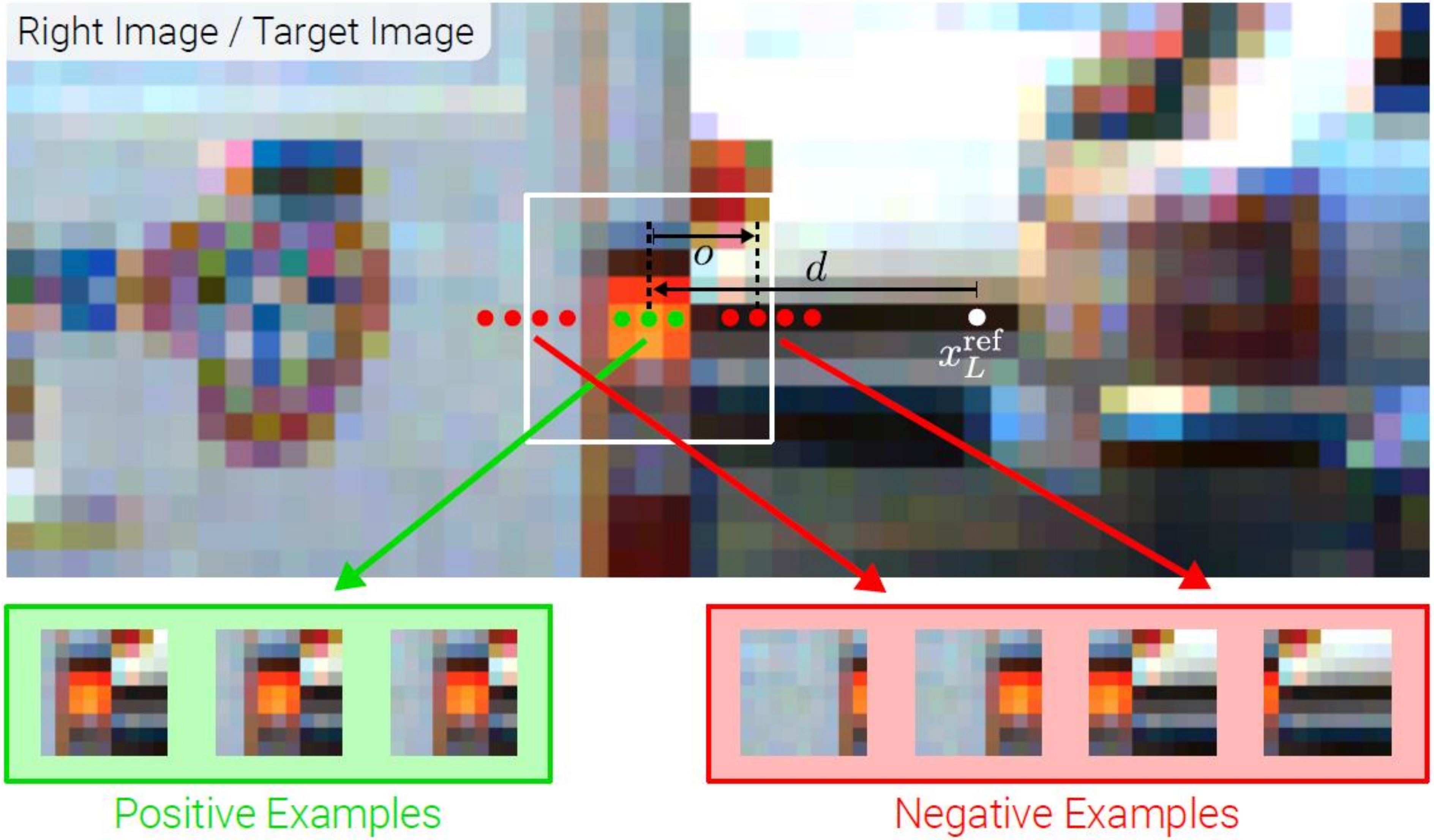
- $\mathbf{x}_r^{neg} = \left(x_l^{ref} - \boxed{d} + o_{neg}, y_l^{ref}\right)$, where o_{neg} sampled in $[-N_h, \dots, -N_l, N_l, \dots, N_h]$

- $P = 1$, while $N_l = 3$ and $N_h = 6$

Zbontar and LeCun “Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches”, JMLR, 2016.

Siamese network

- .



Slide from A. Geiger

Zbontar and LeCun "Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches", JMLR, 2016.

Siamese network

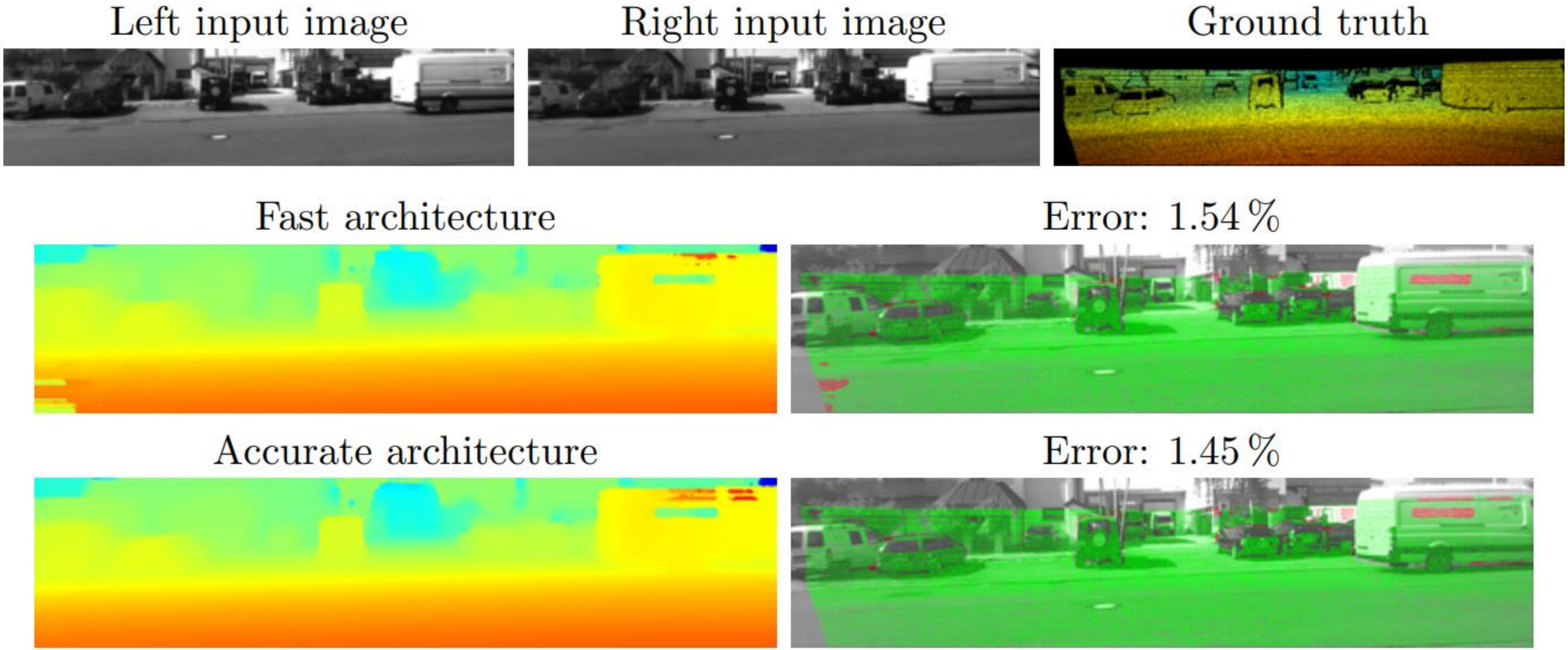
Hinge Loss

$$\mathcal{L} = \max(0, m + s_- - s_+)$$

- s_-/s_+ are the net score for negative/positive example
- $\mathcal{L} = 0$ if $s_+ > s_- + m$, where m is a margin set in the paper at 0.2

Zbontar and LeCun “Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches”, JMLR, 2016.

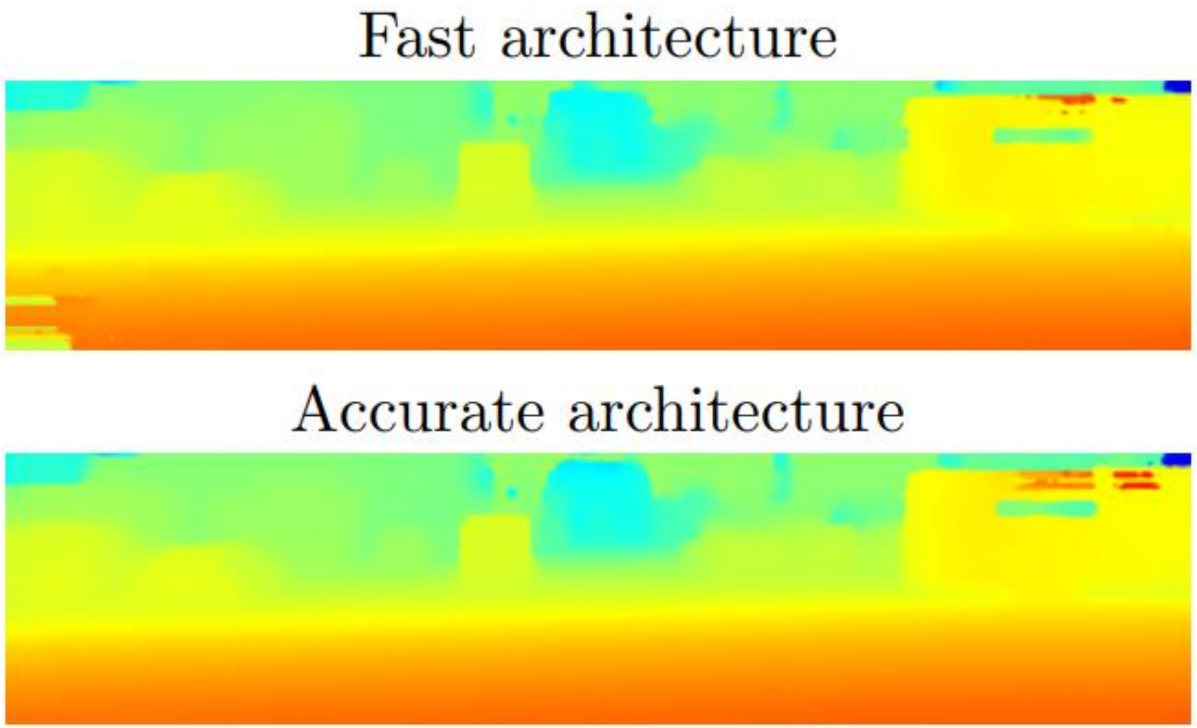
Siamese network



Zbontar and LeCun “Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches”, JMLR, 2016.

Siamese network

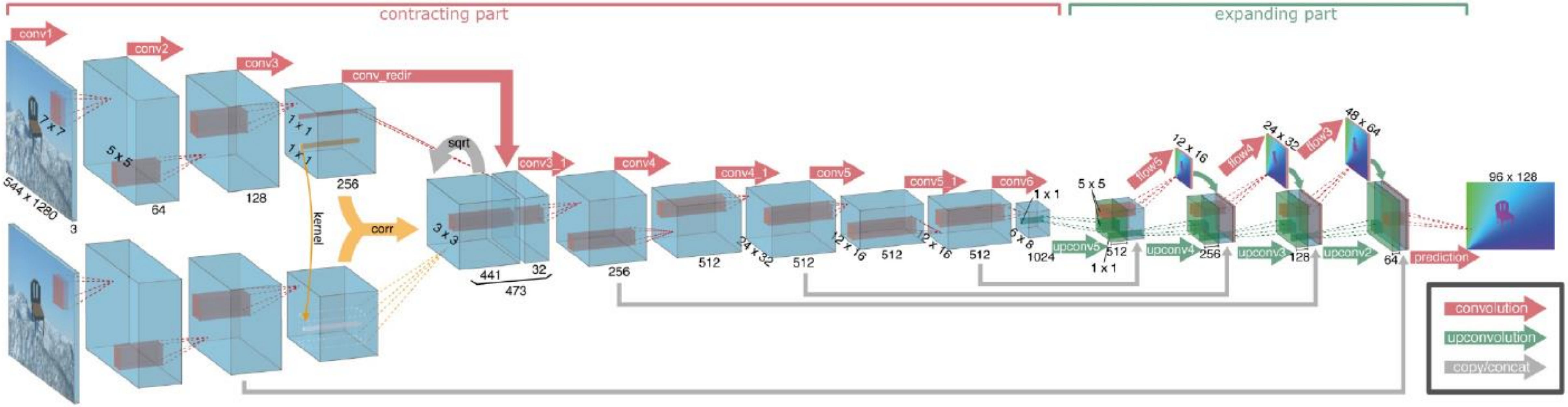
- These are refined disparities, obtained considering also
  - Cross-based Cost Aggregation
  - Semiglobal Matching
  - Interpolation
  - Subpixel Enhancement
  - Refinement



- The net is used to define an initia disparity cost volume

Zbontar and LeCun "Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches", JMLR, 2016.

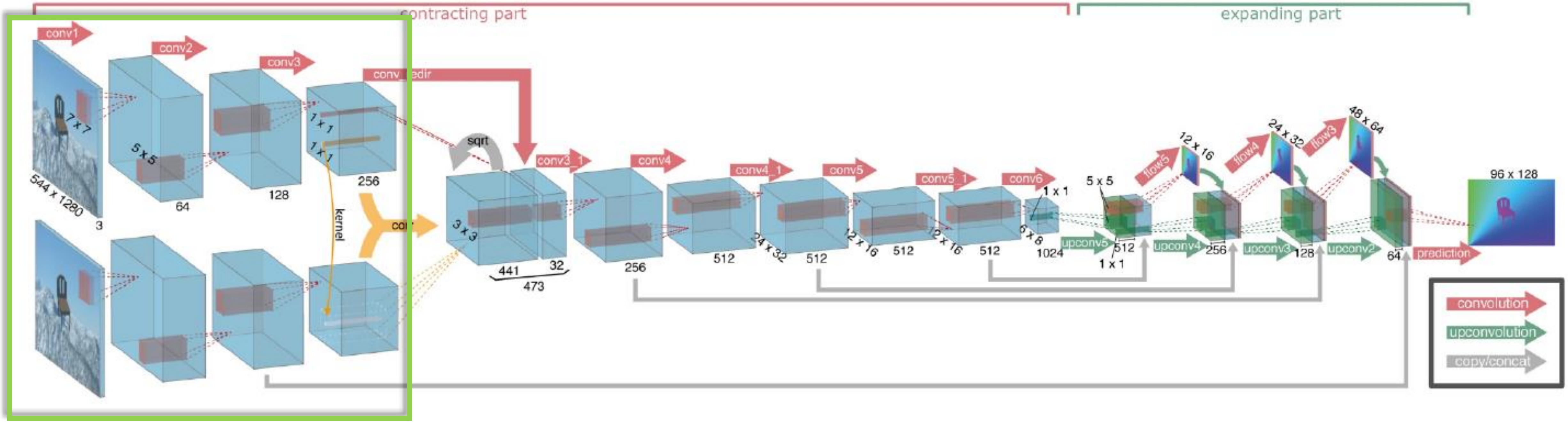
DispNet



- Based on the FlowNet architecture
- Input: a stereo pair
- Output: the disparity map
- The net is trained evaluating the differences between the predicted and the ground truth disparity

Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

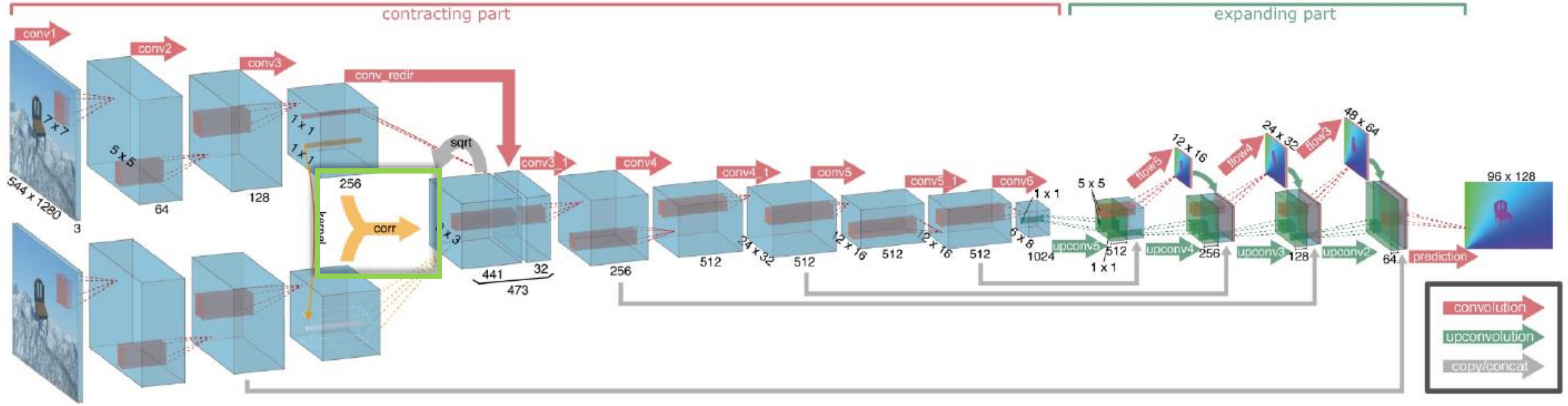
DispNet



- Feature extraction

Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

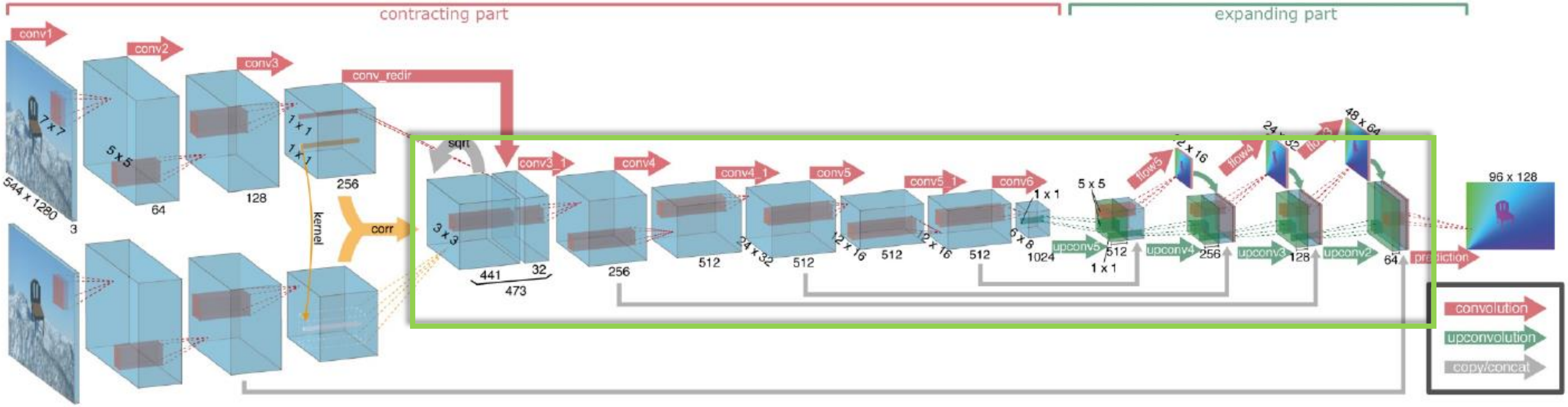
DispNet



- Feature extraction
- Feature 1D-correlation

Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

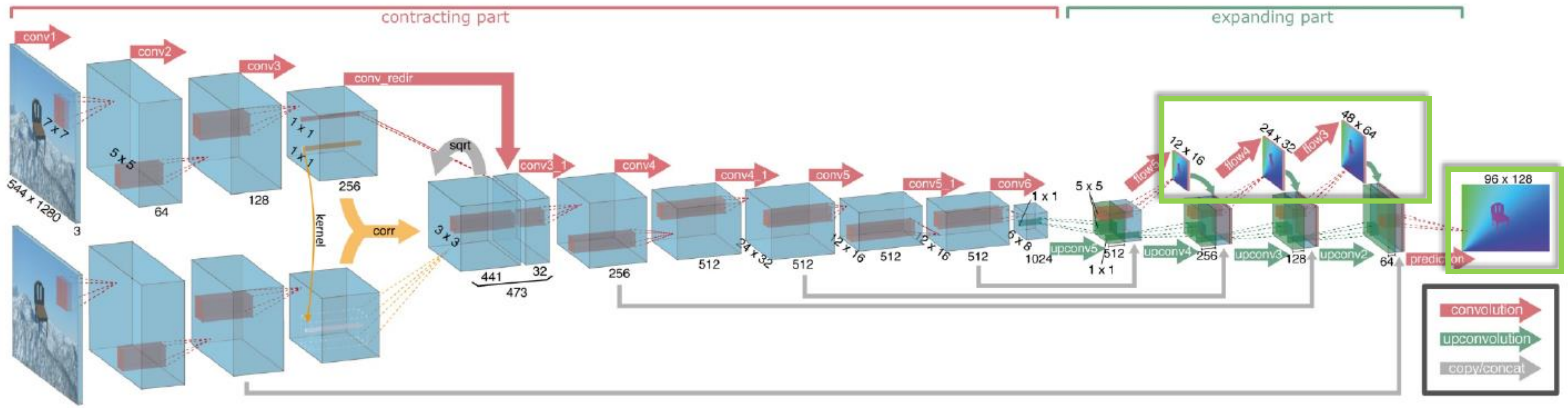
DispNet



- Feature extraction
- Feature 1D-correlation
- U-Net like architecture with skip connections

Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

DispNet



- Feature extraction
- Feature 1D-correlation
- U-Net like architecture with skip connections
- The net use also multi-scale losses and curriculum-learning (from easy to hard example)

Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

DispNet

- Train such an architecture requires an **huge labelled dataset**

- Synthetic dataset were used to train the net

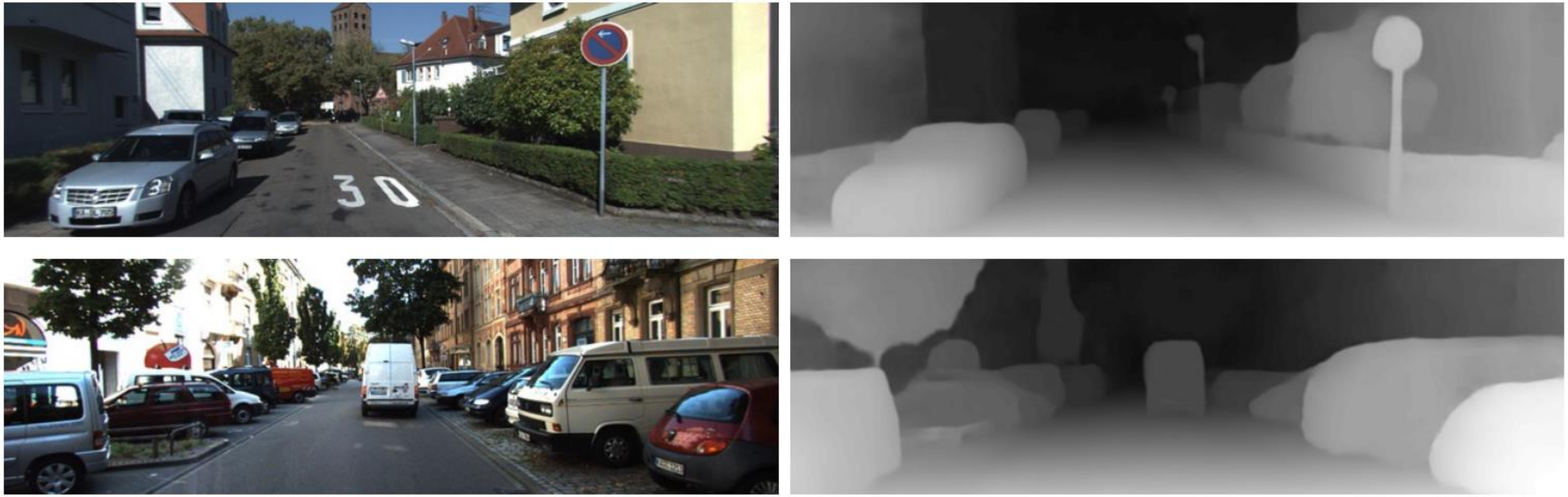
- Real (few) examples was used to fine-tune the net on specific domain

FlyingThings3D

Monkaa

Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

DispNet



Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

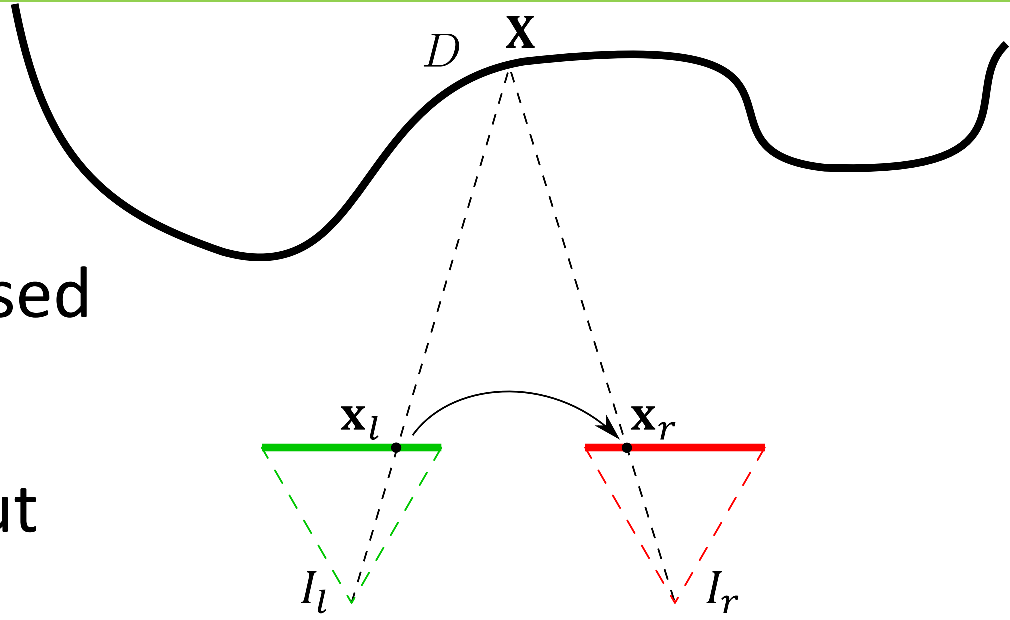
Unsupervised depth learning

- **Goal**: given a single image, predict its depth
- During training, **calibrated stereo images** are used
- During inference, a **single image** is given in input

Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

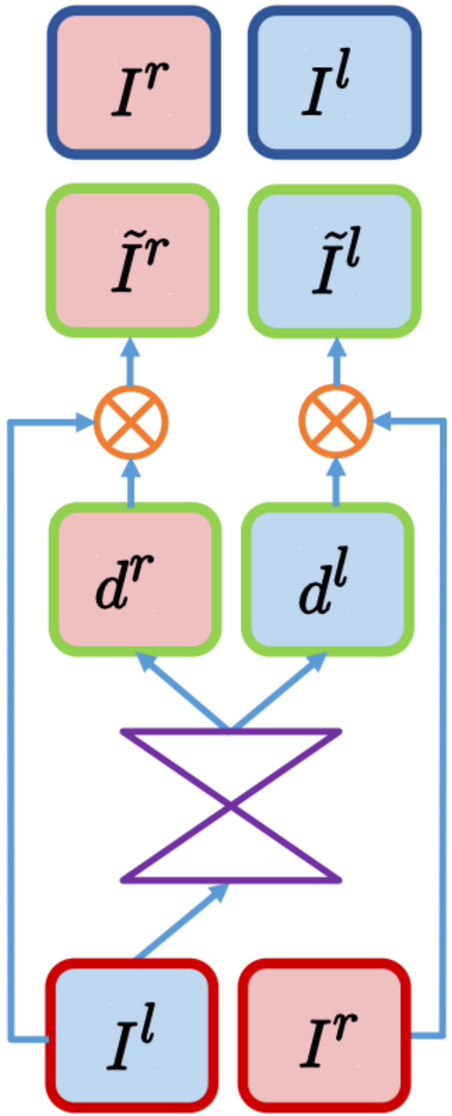
Unsupervised depth learning

- **Goal**: given a single image, predict its depth
- During training, **calibrated stereo images** are used
- During inference, a **single image** is given in input
- Unsupervised training (i.e., no ground truth depth is required) is possible by exploiting **image resynthesis** and enforcing **left-right consistency**



Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

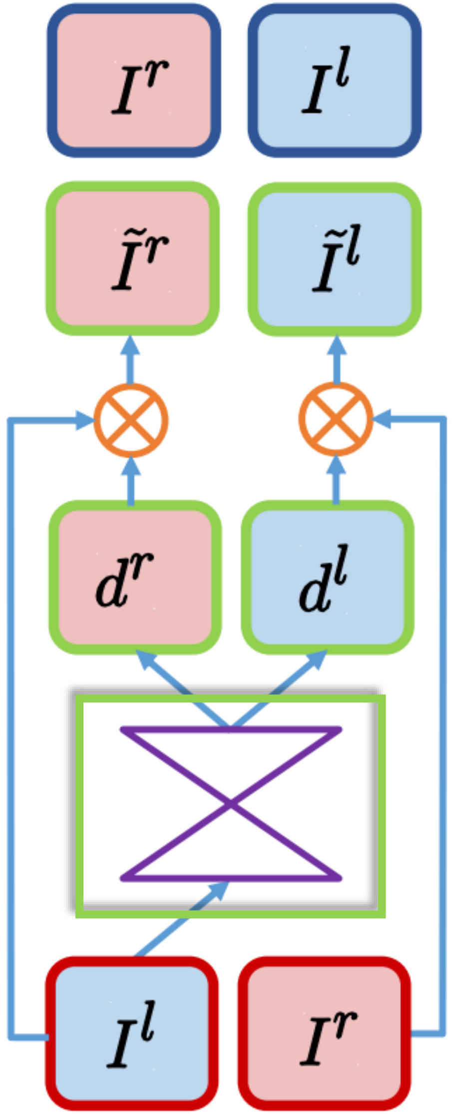
Unsupervised depth learning



Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

Unsupervised depth learning

- The used architecture is inspired by the DispNet, with fully convolutional encoder and decoder



Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

Unsupervised depth learning

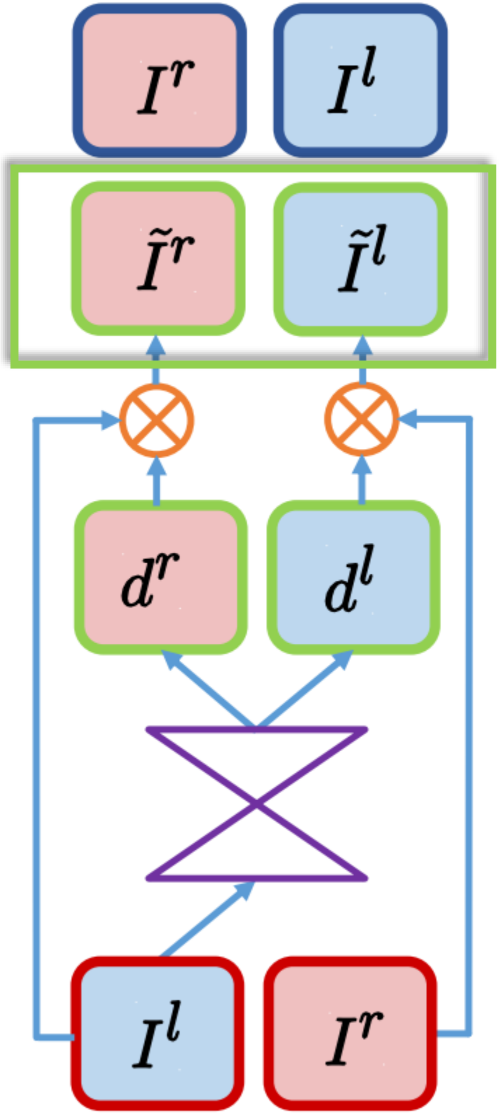
- The used architecture is inspired by the DispNet, with fully convolutional encoder and decoder
- At training time, I^l is used to produce both the **left** (d^l) and **right** (d^r) **disparity maps**

- The used architecture is inspired by the DispNet, with fully convolutional encoder and decoder
- At training time, I^l is used to produce both the **left** (d^l) and **right** (d^r) **disparity maps**

Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

Unsupervised depth learning

- The used architecture is inspired by the DispNet, with fully convolutional encoder and decoder
- At training time, I^l is used to produce both the **left** (d^l) and **right** (d^r) **disparity maps**
- Then the disparities are used to **reconstruct the input images**
    - $\tilde{I}^r = I^l(d^r)$
    - $\tilde{I}^l = I^r(d^l)$



Godard et al., "Unsupervised monocular depth estimation with left-right consistency", CVPR, 2017.

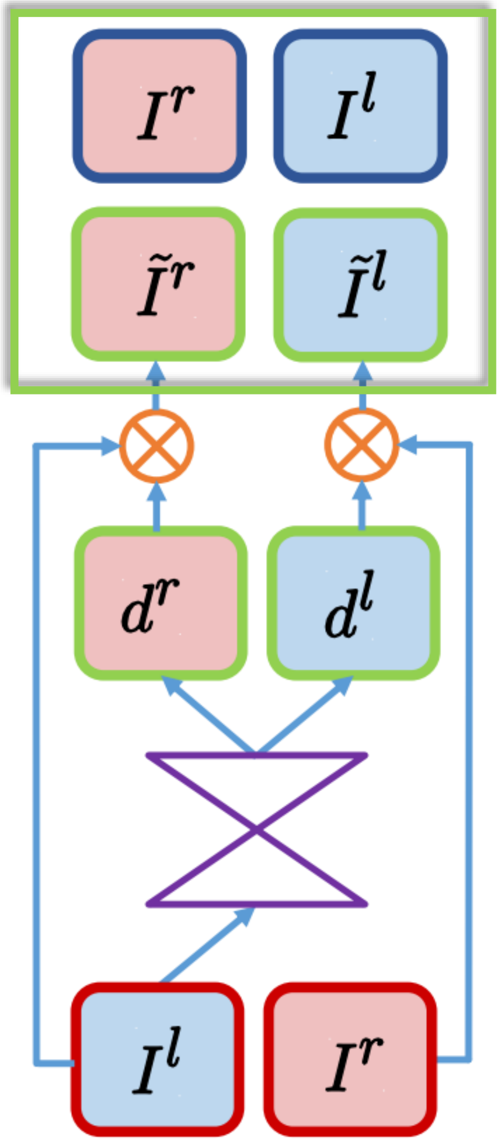
Unsupervised depth learning

- To train the net, the reconstructed images $(\tilde{I}^l, \tilde{I}^r)$ are compared with the original ones (I^l, I^r)

- The loss function, evaluated at multiple scale, is composed by several parts

$$C = \sum_{s=1}^{4} C_s$$

$$C_s = \alpha_{ap}\left(C_{ap}^l + C_{ap}^r\right) + \alpha_{ds}\left(C_{ds}^l + C_{ds}^r\right) + \alpha_{lr}\left(C_{lr}^l + C_{lr}^r\right)$$

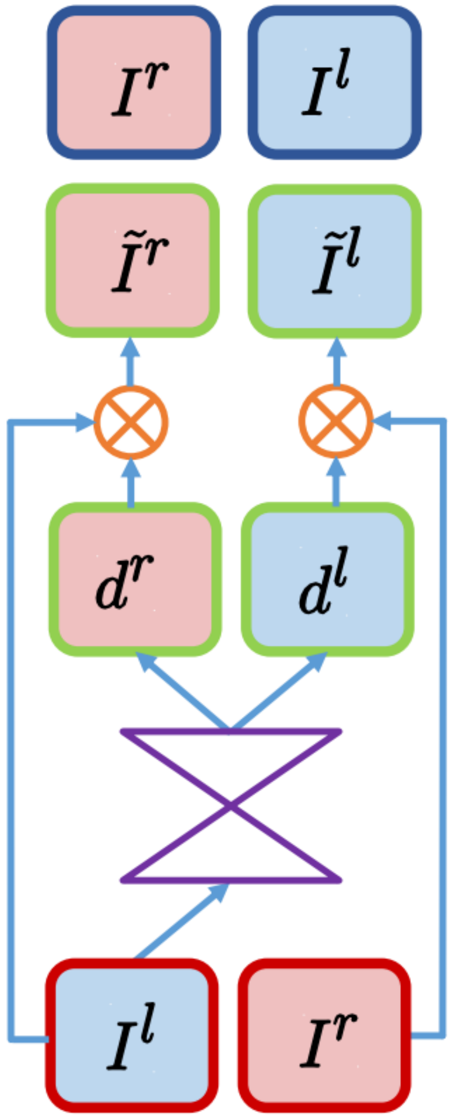


Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

Unsupervised depth learning

$$C_s = \alpha_{ap}\left(C_{ap}^l + C_{ap}^r\right) + \alpha_{ds}\left(C_{ds}^l + C_{ds}^r\right) + \alpha_{lr}\left(C_{lr}^l + C_{lr}^r\right)$$

$$C_{ap}^l = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - SSIM\left(I_{ij}^l, \tilde{I}_{ij}^l\right)}{2} + (1-\alpha)\left|I_{ij}^l - \tilde{I}_{ij}^l\right|$$



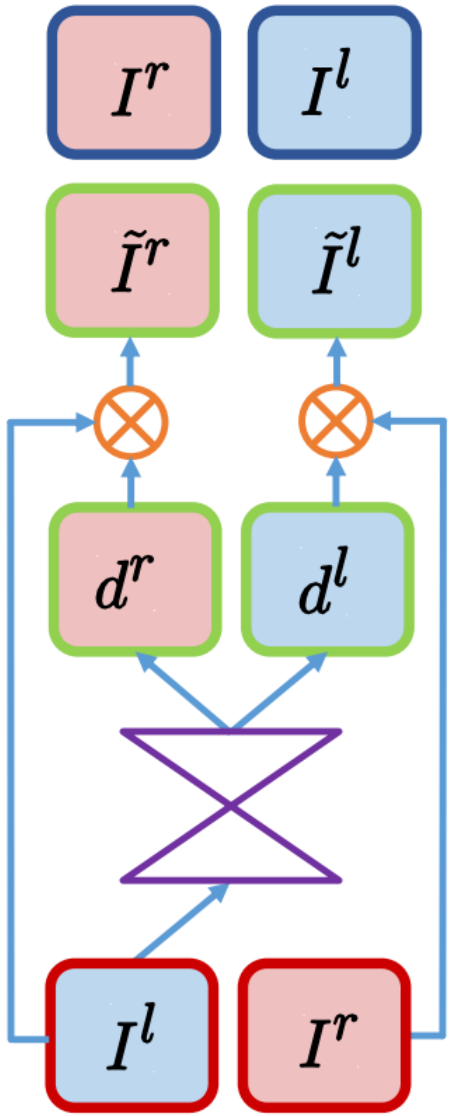
- Appearance matching loss
  - Encourages the reconstructed images to appear similar to the input images

Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

Unsupervised depth learning

$$C_s = \alpha_{ap}\left(C_{ap}^l + C_{ap}^r\right) + \alpha_{ds}\left(C_{ds}^l + C_{ds}^r\right) + \alpha_{lr}\left(C_{lr}^l + C_{lr}^r\right)$$

$$C_{ds}^l = \frac{1}{N}\sum_{i,j}\left|\partial_x d_{ij}^l\right| e^{-\left|\partial_x I_{ij}^l\right|} + \left|\partial_y d_{ij}^l\right| e^{-\left|\partial_y I_{ij}^l\right|}$$



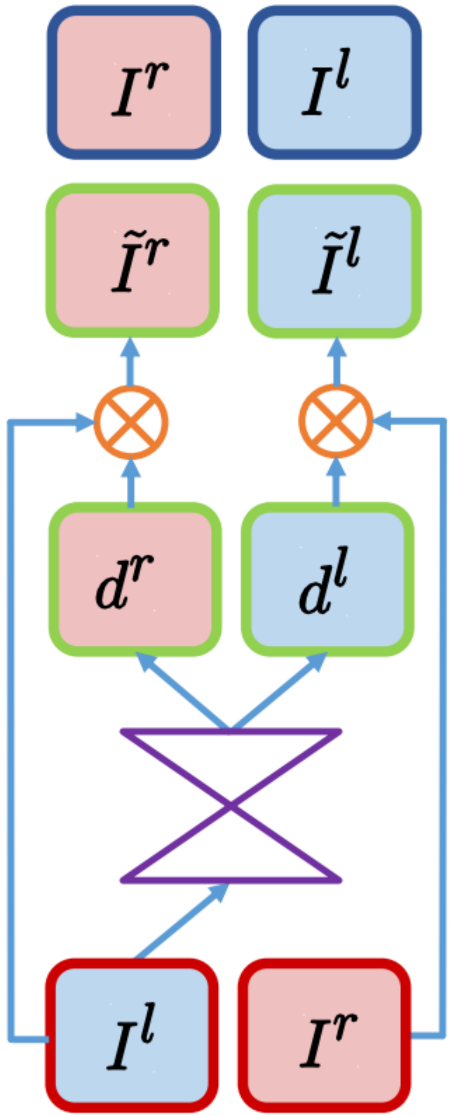
- Disparity smoothness loss
  - Encourages the predicted disparity maps to be smooth evaluating their gradients, weighted with the image gradient to take edges into account

Godard et al., "Unsupervised monocular depth estimation with left-right consistency", CVPR, 2017.

Unsupervised depth learning

$$C_s = \alpha_{ap}\left(C_{ap}^l + C_{ap}^r\right) + \alpha_{ds}\left(C_{ds}^l + C_{ds}^r\right) + \alpha_{lr}\left(C_{lr}^l + C_{lr}^r\right)$$

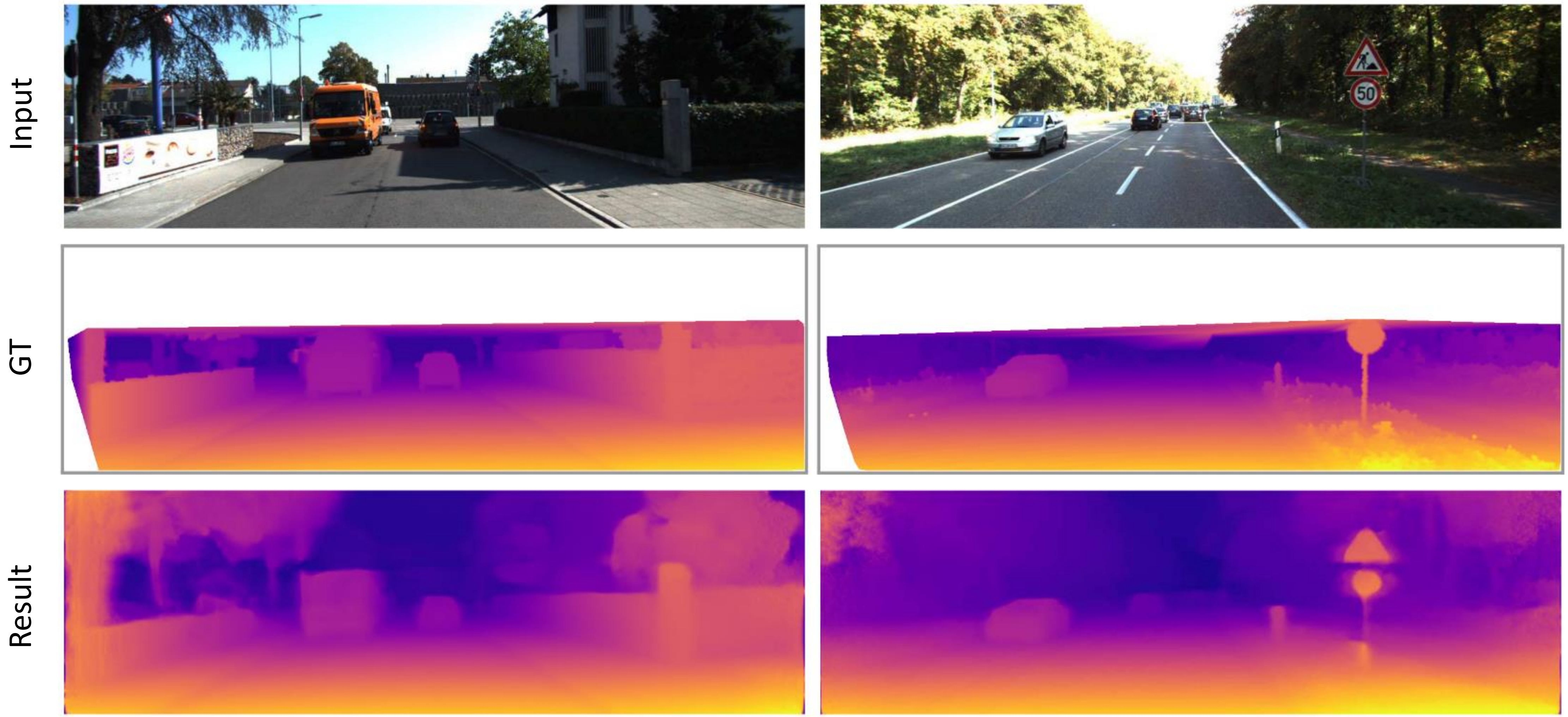
$$C_{lr}^l = \frac{1}{N}\sum_{i,j}\left|d_{ij}^l - d_{ij-d_{ij}^l}^r\right|$$



- Left-right disparity consistency loss
  - To force the predicted disparity (both obtained from the left image only) to be consistent, i.e. have the left disparity equal to the projected right-view disparity

Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

Unsupervised depth learning



Godard et al., "Unsupervised monocular depth estimation with left-right consistency“, CVPR, 2017.

3D Reconstruction

Structure from Motion

Structure from Motion

- Input:
  - Unordered image collection
  - Videos

- Output:
  - 3D (sparse) structure
  - Camera positions

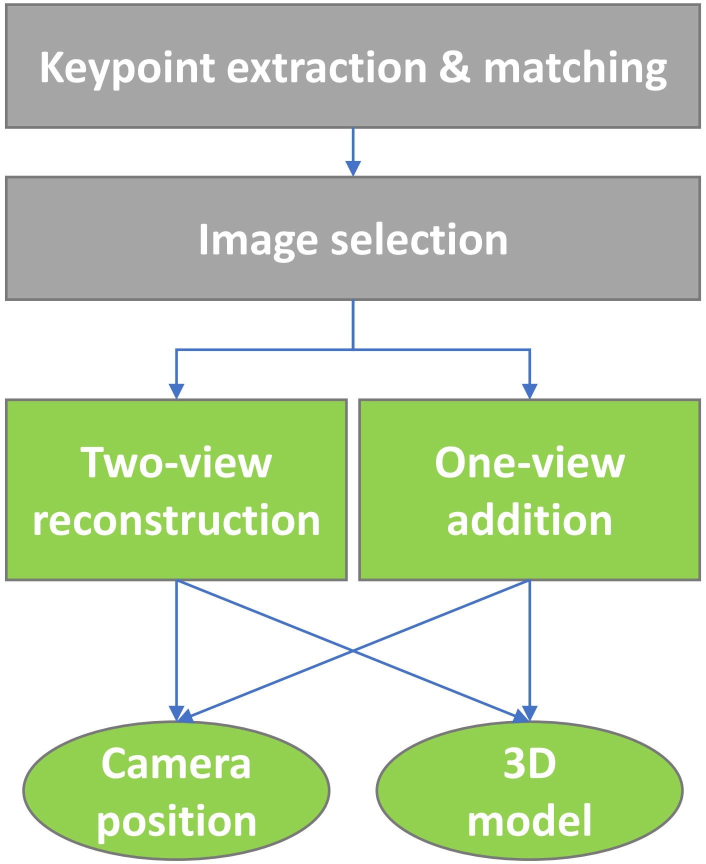
Structure from Motion - Pipeline

1. Image analysis

Structure from Motion - Pipeline

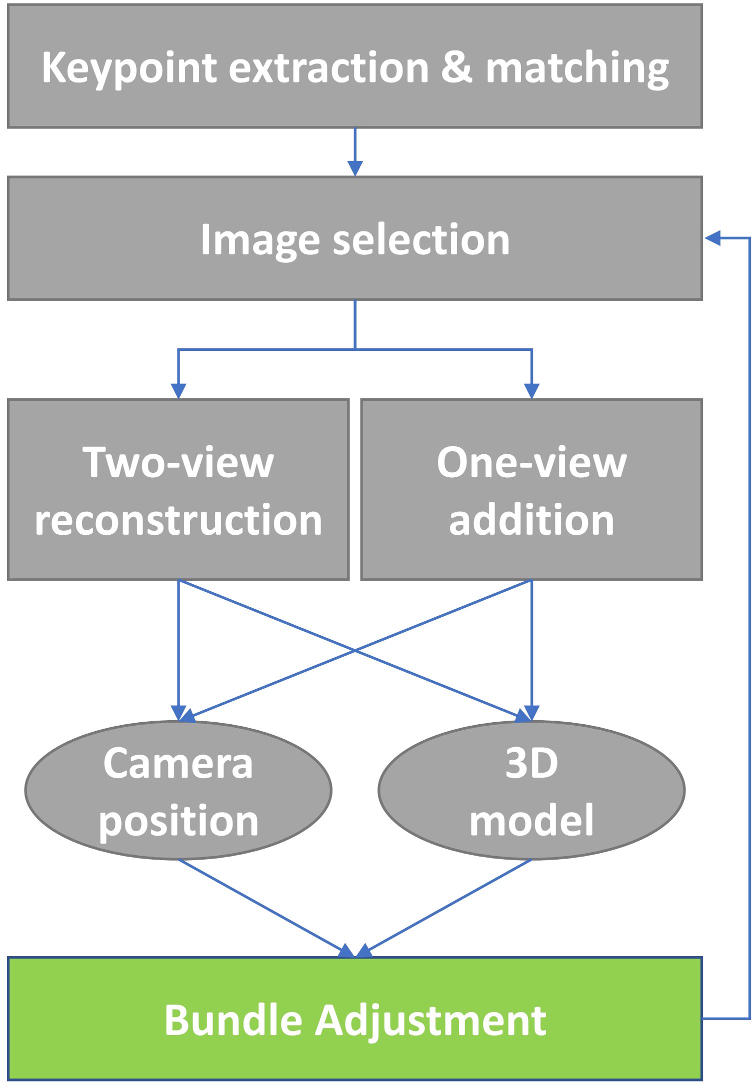
1. Image analysis

2. Geometric estimation

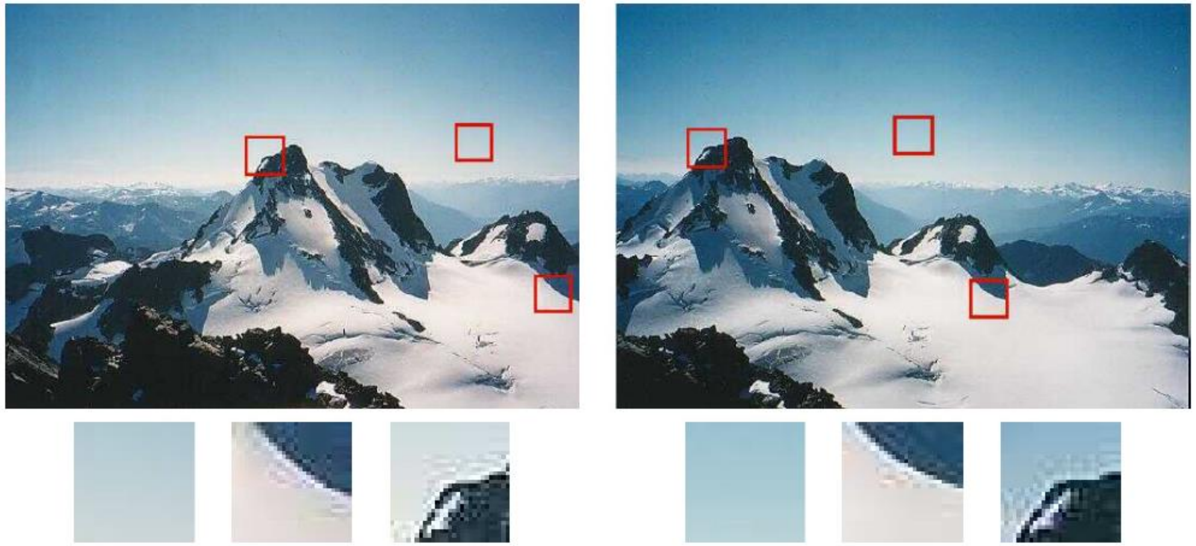


Structure from Motion - Pipeline

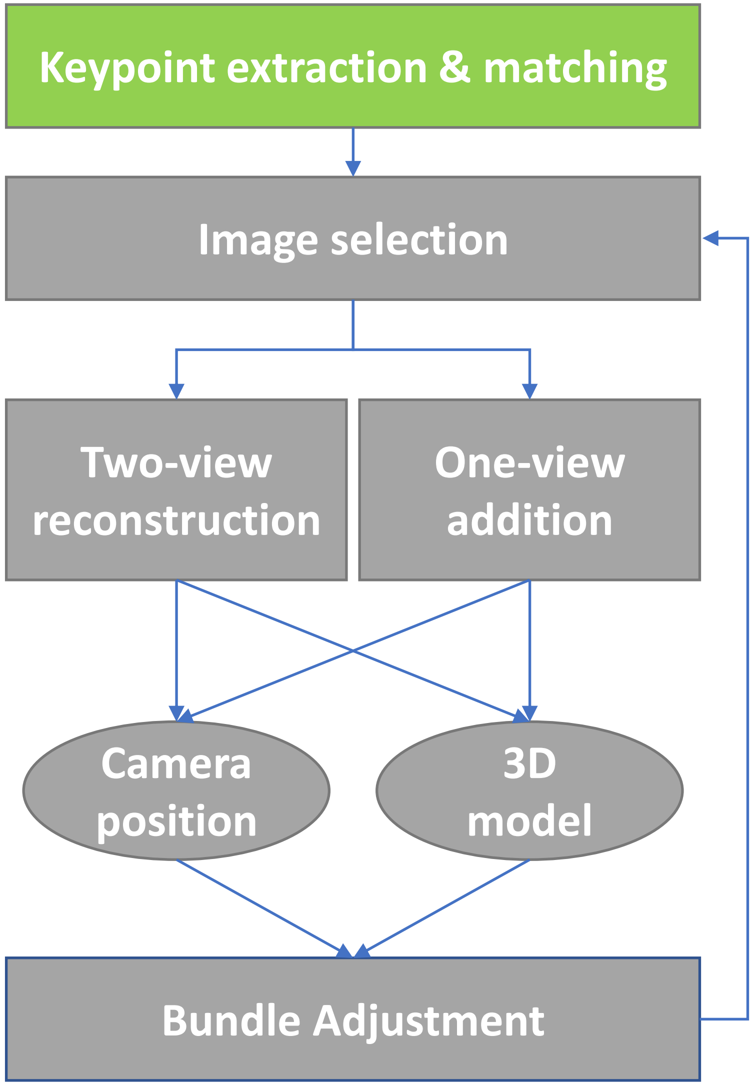
1. Image analysis
2. Geometric estimation
3. Local/global optimization



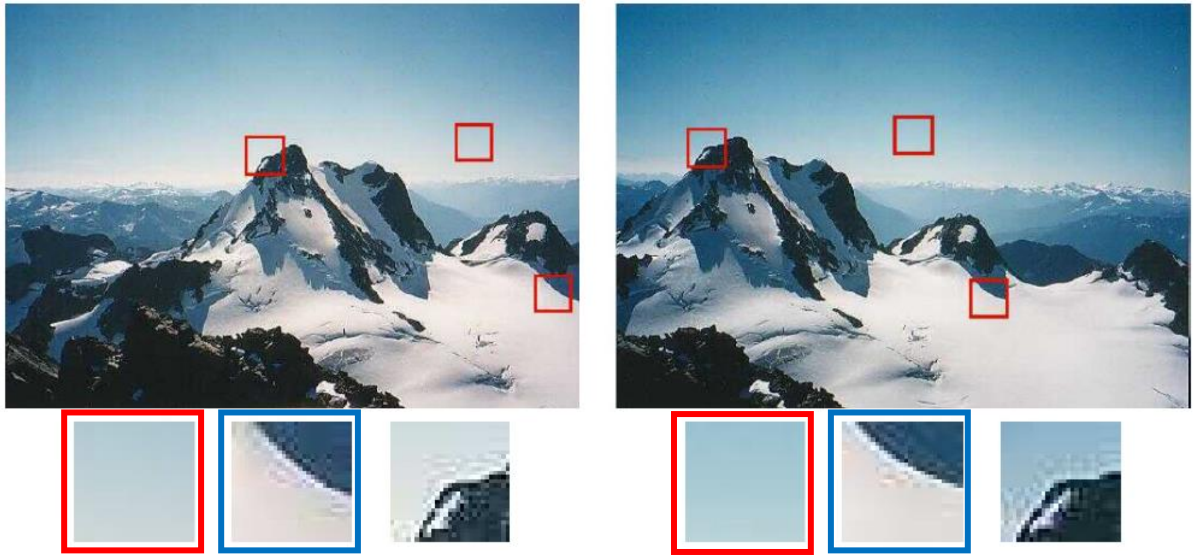
Point matching



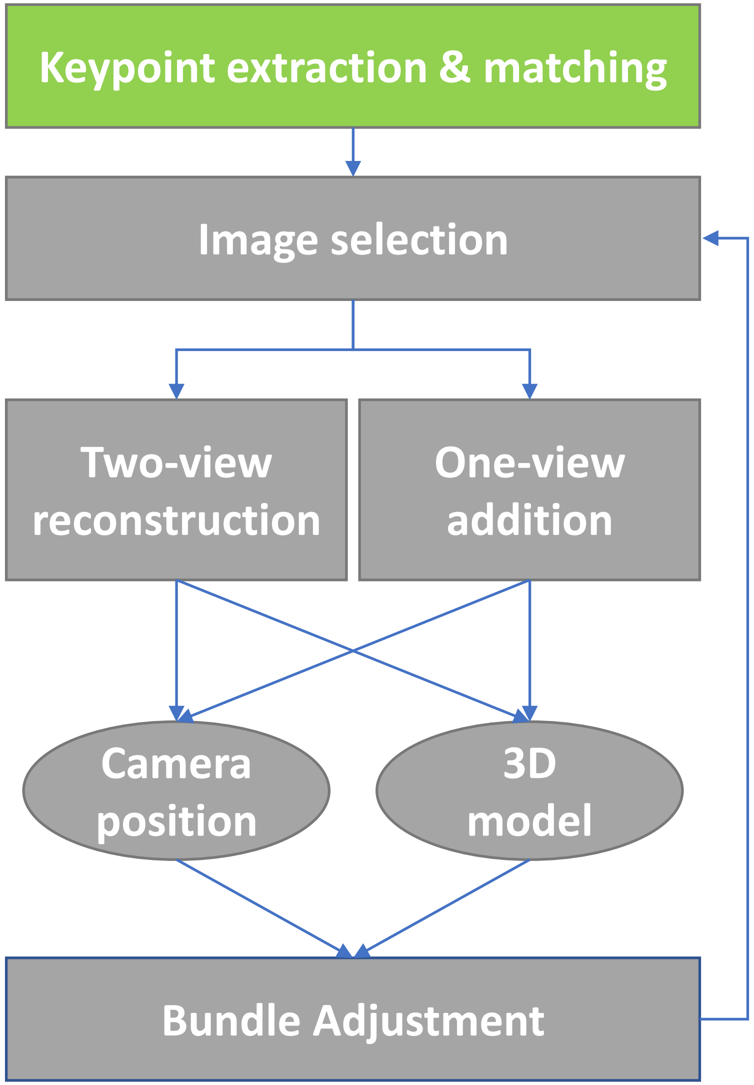
- We want to find patches that are recognizable among different images of the same scene



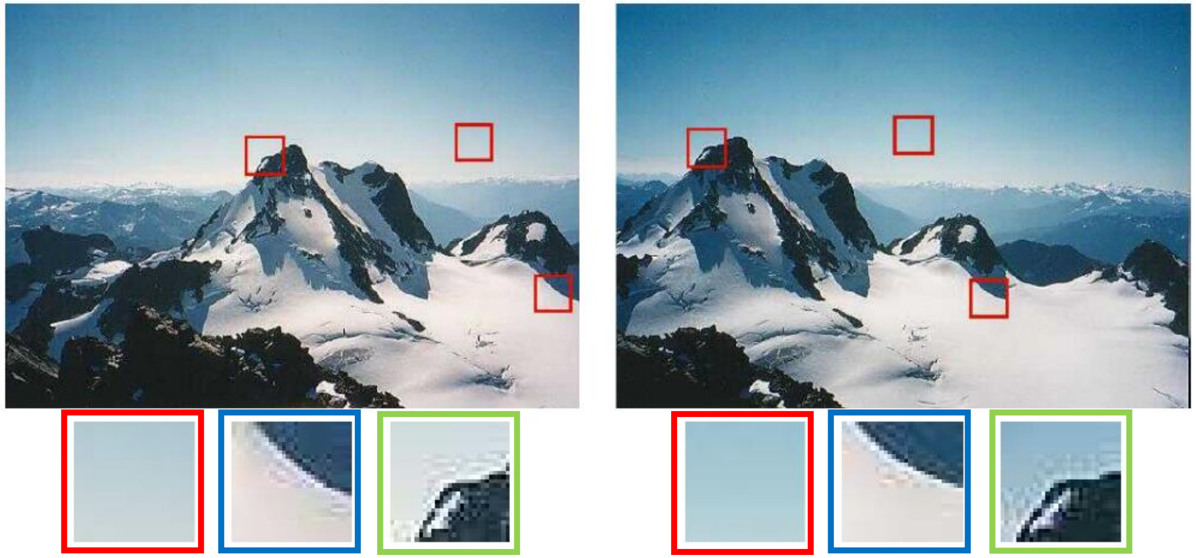
Point matching



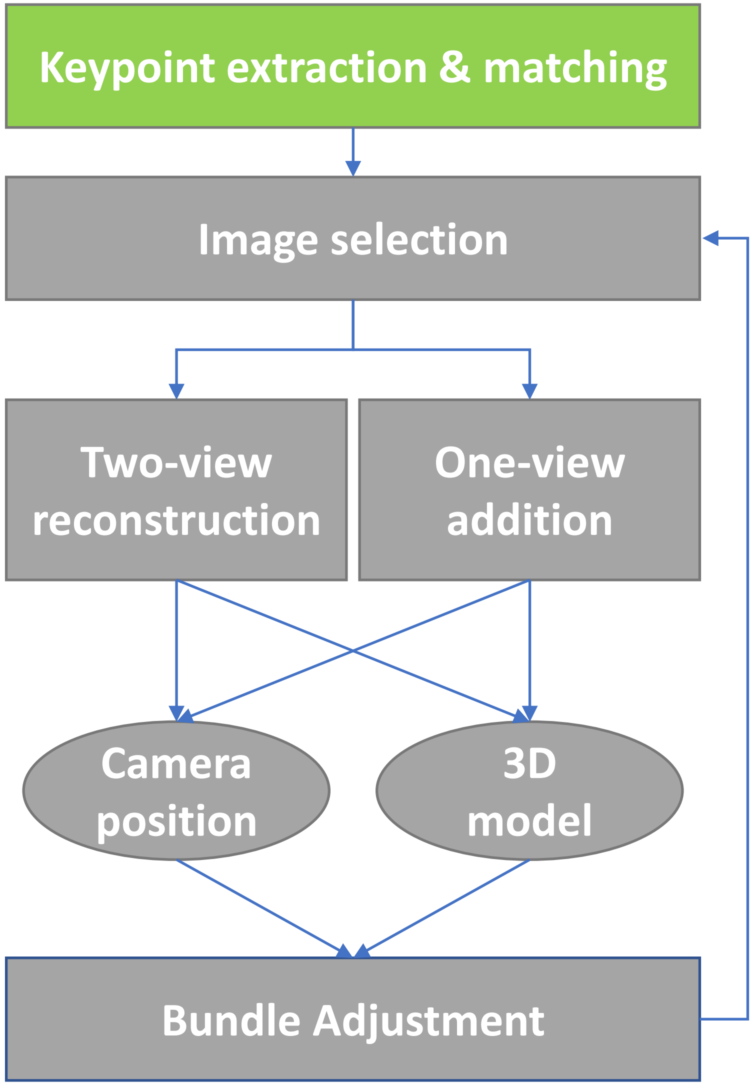
- We want to find patches that are recognizable among different images of the same scene
- Not all patches are good
  - Low texture content
  - Ambiguous region



Point matching

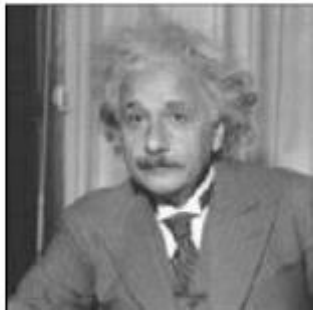


- We want to find patches that are recognizable among different images of the same scene
- Not all patches are good
  - Low texture content
  - Ambiguous region
- We want patches with distinctive local appearance



SIFT – Scale Invariant Feature Transform

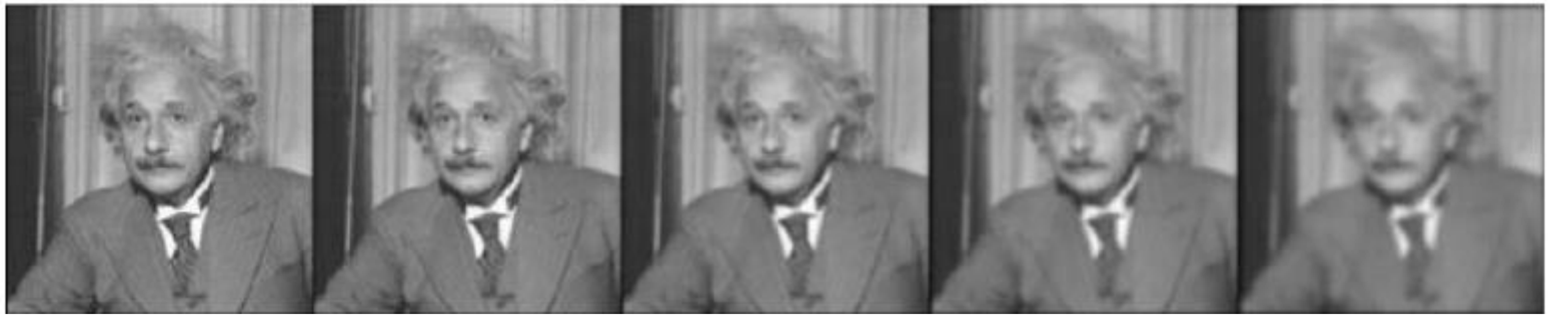
- First step: build a scale-space representation



Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004).

SIFT – Scale Invariant Feature Transform

- First step: build a scale-space representation
  - Filter the image with a Gaussian kernel with increasing σ



Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004).

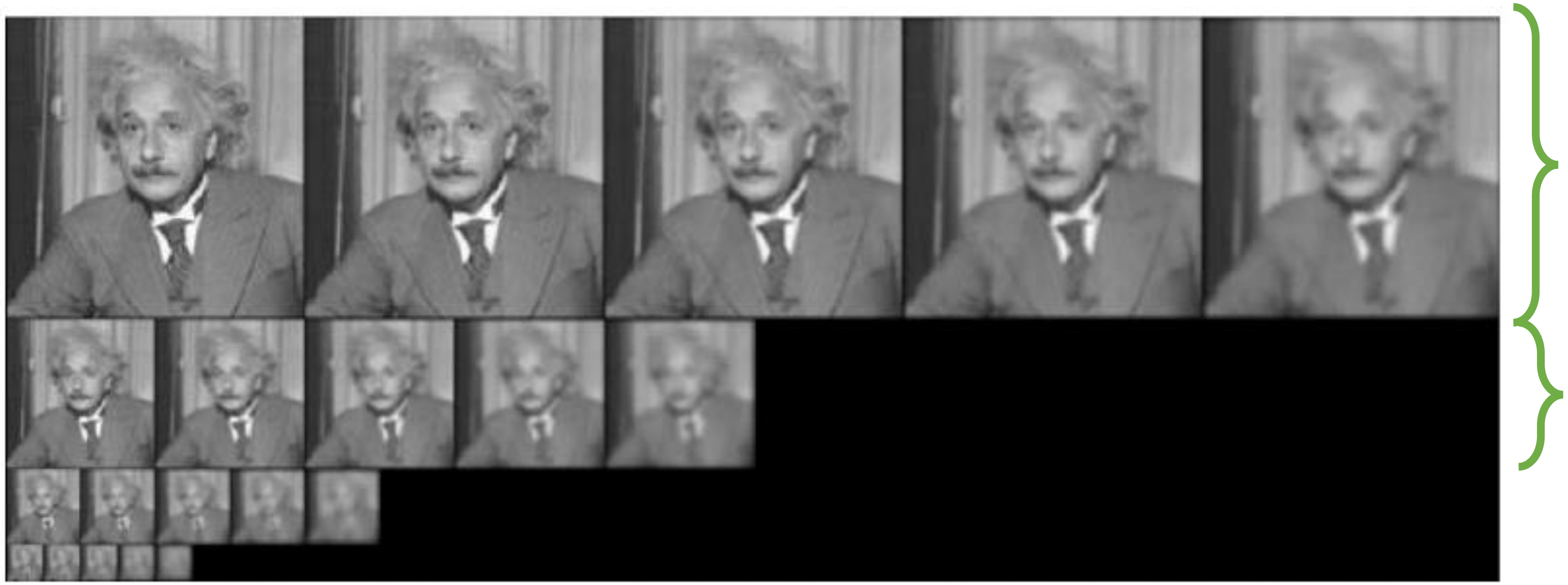
SIFT – Scale Invariant Feature Transform

- First step: build a scale-space representation
  - Filter the image with a Gaussian kernel with increasing σ
  - Downscale the image, and repeat

Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004).

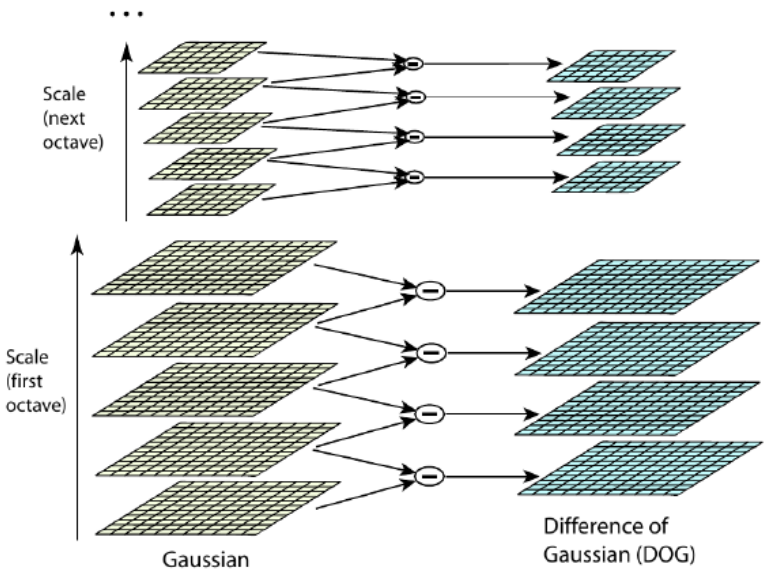
SIFT – Scale Invariant Feature Transform

- First step: build a scale-space representation
  - Filter the image with a Gaussian kernel with increasing σ
  - Downscale the image, and repeat



Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004).

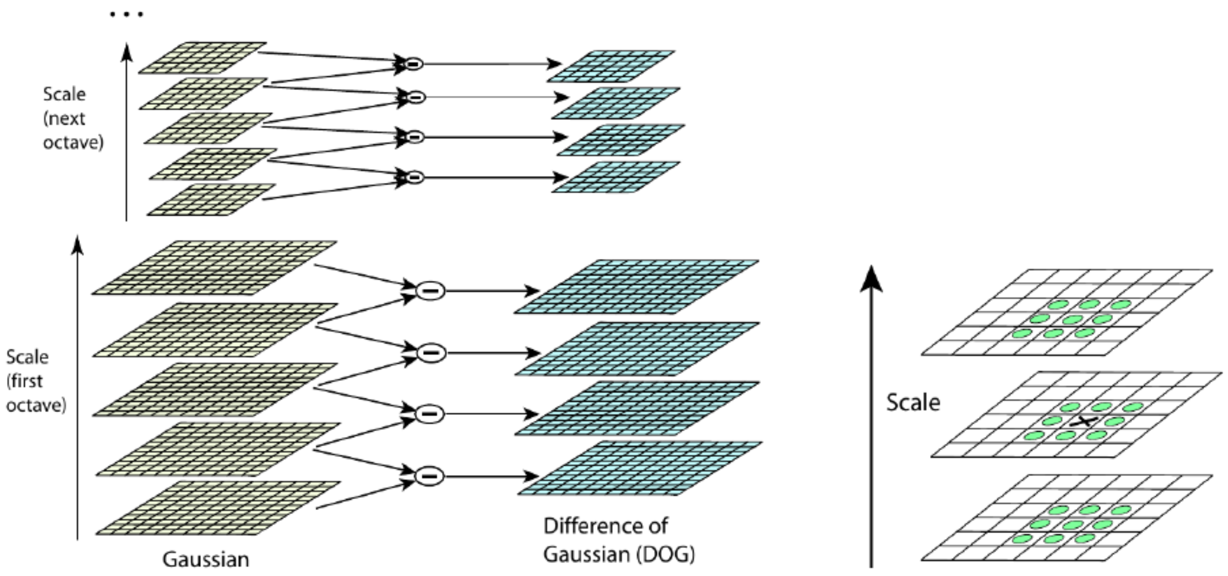
SIFT – Scale Invariant Feature Transform

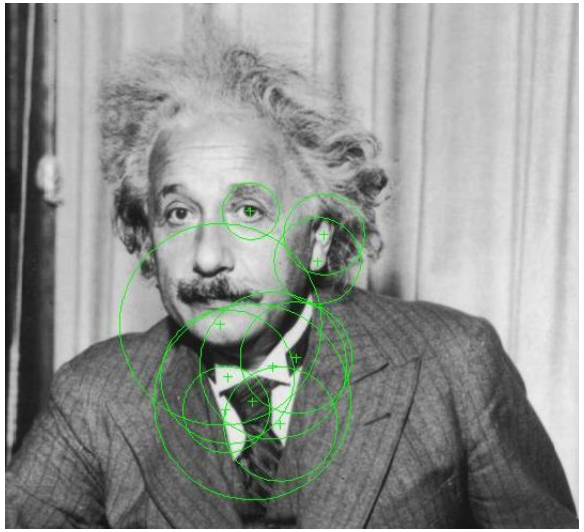


- Adjacent scale are subtracted to obtain a **Difference of Gaussian (DoG)**

Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004).

SIFT – Scale Invariant Feature Transform

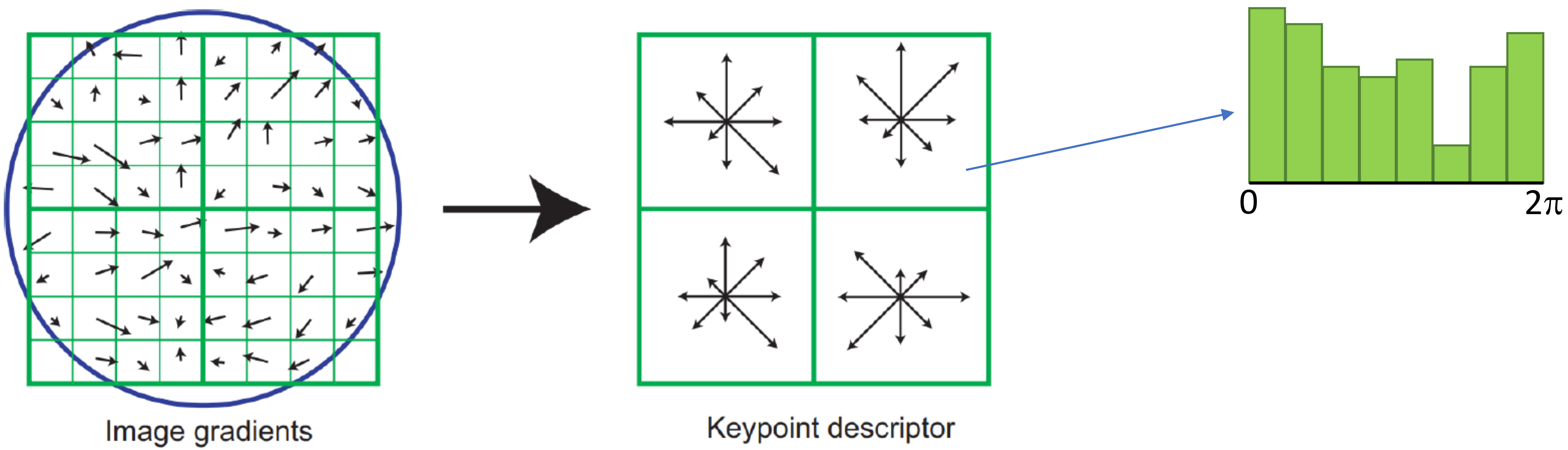




- Adjacent scale are subtracted to obtain a **Difference of Gaussian (DoG)**
- DoG **extrema points** are selected as keypoints (blobs)
- Blobs correspond to areas of high intensity change making them ideal for feature extraction tasks.

Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004).

SIFT – Scale Invariant Feature Transform



- To obtain a keypoint descriptor, **gradients** are computed in the surrounding pixels
- For each sub-area an **histogram of the gradient orientations** is obtained
- The SIFT descriptor is the concatenation of all the histograms

Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004).

Point matching

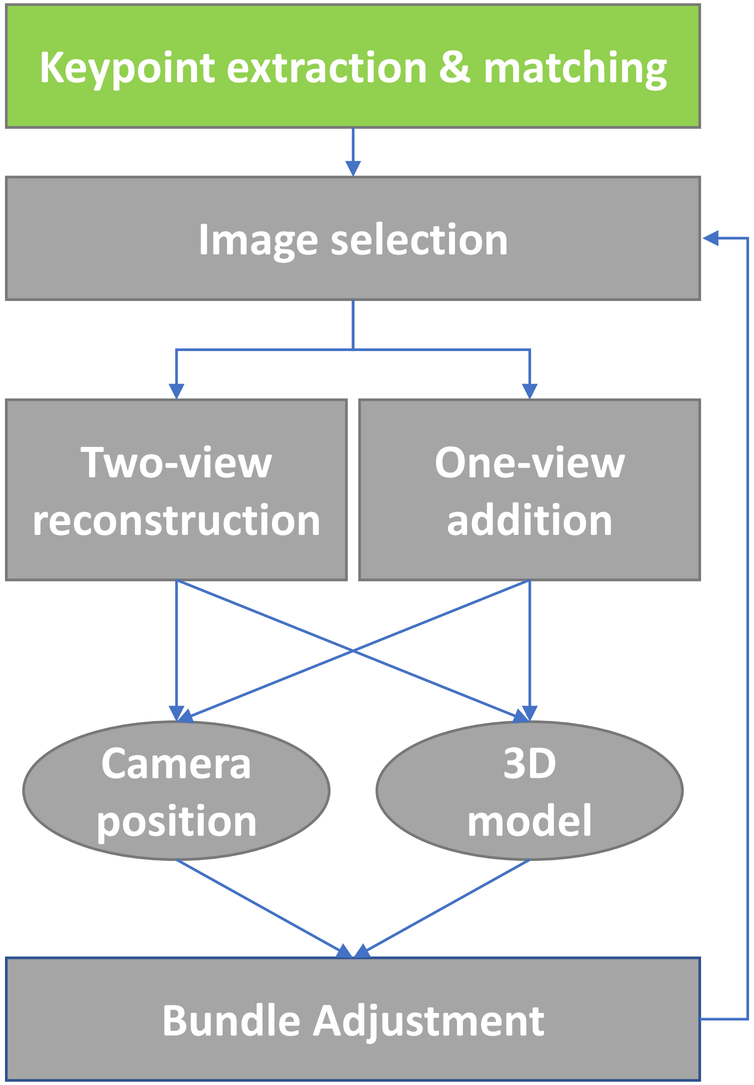
- Other than SIFT, there is a plethora of keypoint descriptors/detectors:
  - Harris Corner
  - SURF
  - FAST
  - BRIEF
  - ORB
  - …
- Also, deep-learning based solutions are available
  - Superpoint
  - D2-Net
  - LF-Net
  - …

Point matching

- Each point set $\{\mathbf{x}_i, desc(\mathbf{x}_i)\}$ is compared against all the sets extracted

- Matches are evaluated by measuring the distance (e.g. L1 or L2) between the descriptor vectors. A match is a pair of points $(\mathbf{x}_i, \mathbf{x}_j)$ from different images with minimum descriptor distance.

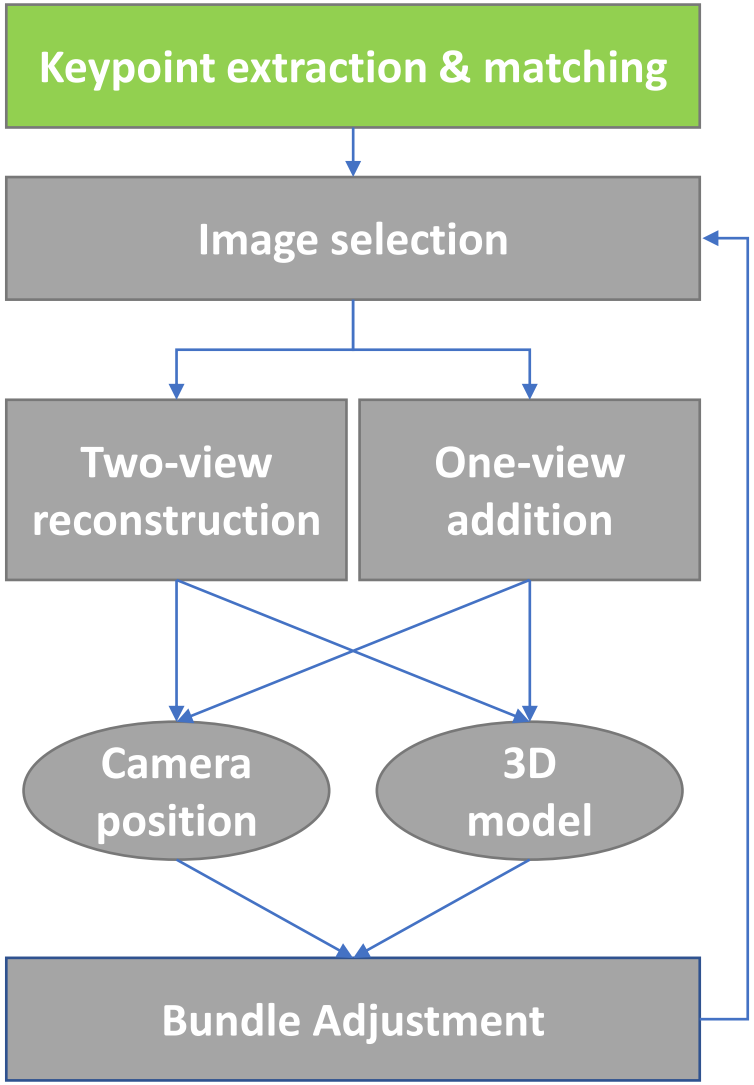
- By concatenating the matches among different images, we will obtain a **track**, i.e., all the projections of a single 3D point



Point matching

- Matches must be validated with **robust estimation of epipolar geometry** (i.e., the fundamental matrix, $\mathbf{x}_2^T F \mathbf{x}_1 = 0$) to discard outliers.

- The RANdom SAmple Consensus (RANSAC) algorithm can be used



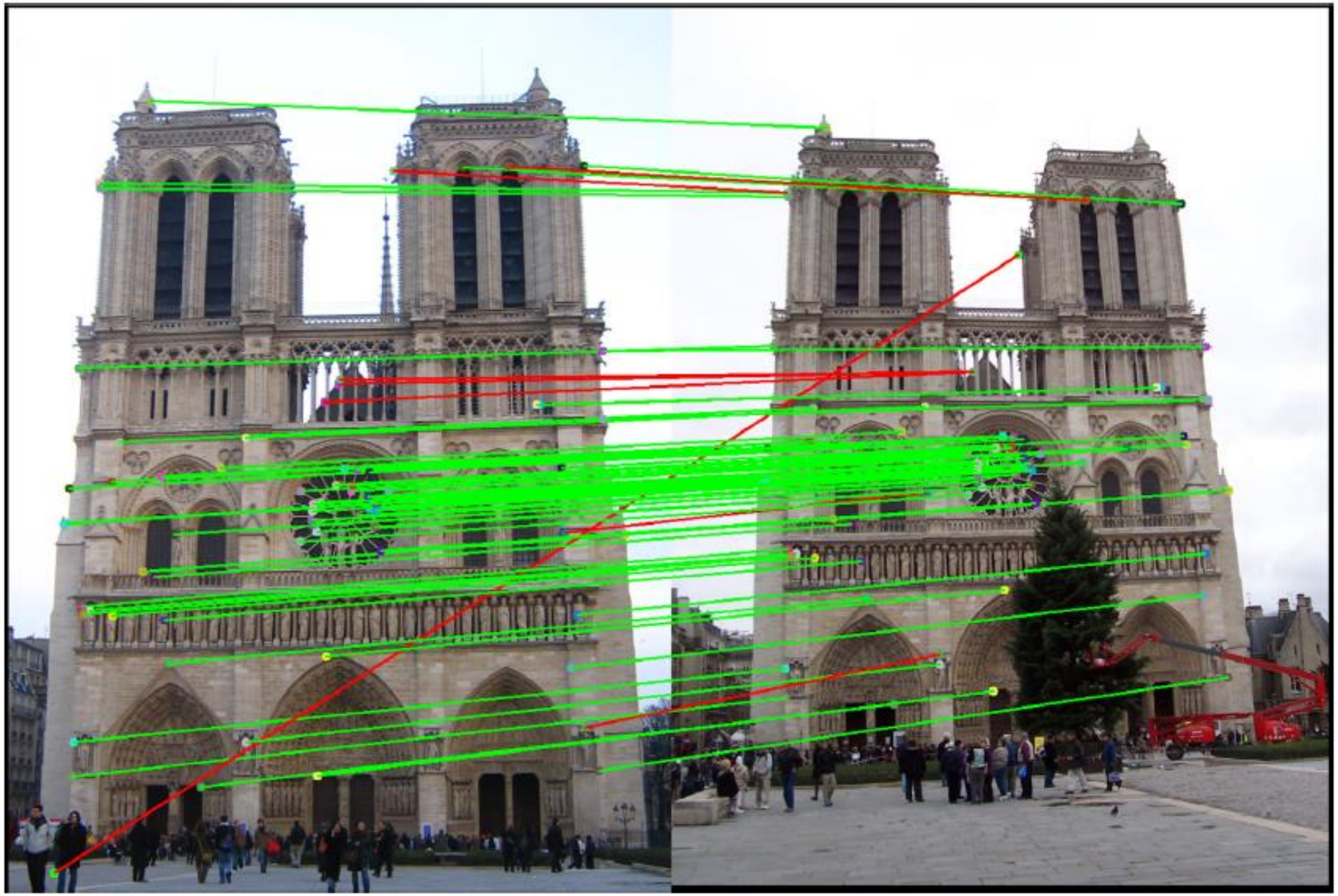
Point matching

Robust 8-point algorithm

Let $\{x_i\}$ and $\{x_j\}$ be the set of matched keypoints between images I and J

1. Randomly select 8 matches
2. Compute F with the 8-point algorithm using the selected matches
3. Validate the obtained F, using, for example, the distance $d(\mathbf{x}_j, \mathbf{l}_j)$ where $\mathbf{l}_j = \mathrm{F}\mathbf{x}_i$ for all the matches
4. Define the inlier and the outlier sets according to a threshold τ
   I. if $d(\mathbf{x}_j, \mathbf{l}_j) \leq \tau$, $\mathbf{x}_j$ is an inlier
   II. if $d(\mathbf{x}_j, \mathbf{l}_j) > \tau$, $\mathbf{x}_j$ is an outlier
5. Repeat 1 – 4 for k iterations
6. Retrieve the maximum inlier set
7. Using all the inliers compute the final F

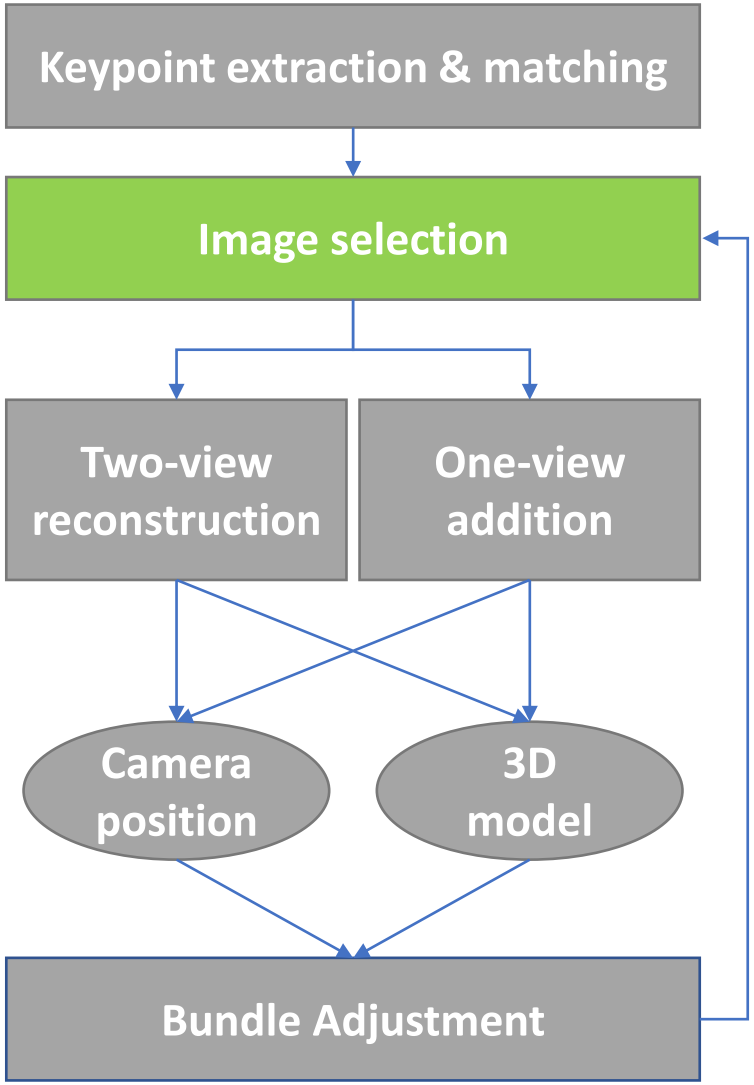
Point matching



Inlier
Outlier

Image selection

- To avoid bad conditioning, we have to carefully search for the best image pair from which start the reconstruction, i.e., images that have:
    - High number of matches
    - Sufficient baseline
- After the initialization, the same heuristics are used to select the successive image to be included in the process
- How to check the baseline?
    - Match flow measurement
    - Low percentage of homography inliers
    - Geometric Robust Information Criterion[1] (GRIC)



Initialization – two view reconstruction

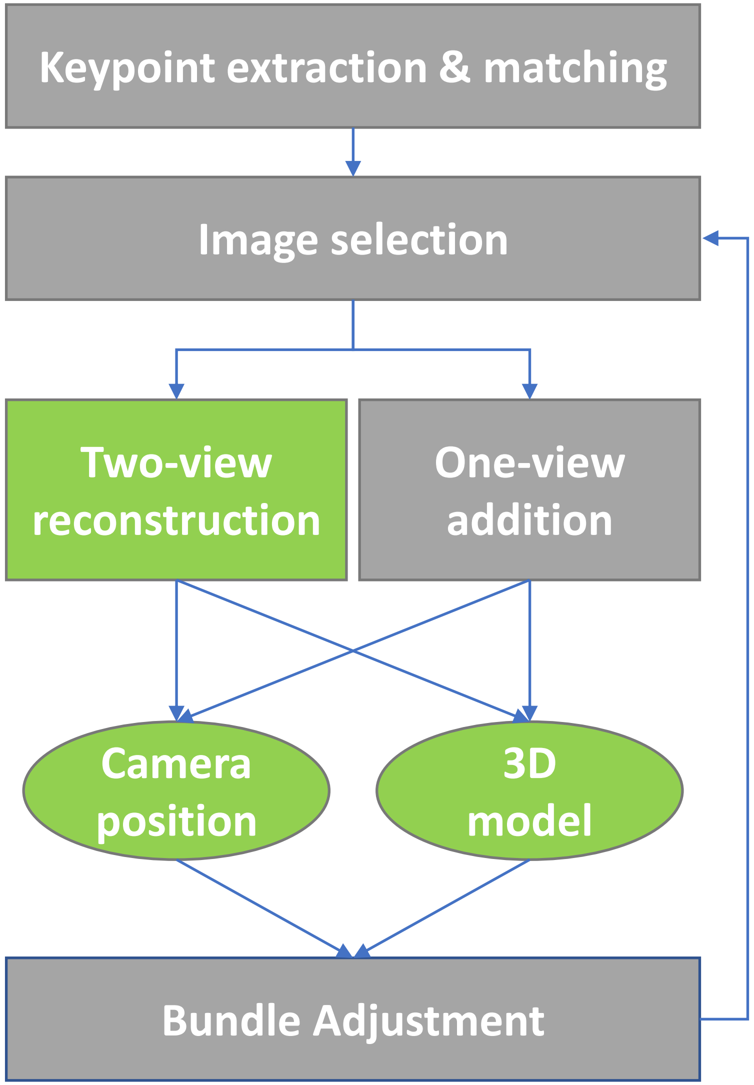
- The **essential matrix** is computed between the first image pair

$$E = K_2^T F K_1 = [\mathbf{t}]_\times R$$

- Then, by decomposing E, the **first two camera matrices** can be defined as

$$P_1 = K_1[I \quad \mathbf{0}] \text{ and } P_2 = K_2[R \quad \mathbf{t}/||\mathbf{t}||]$$

- Finally, the **initial 3D structure** is computed by triangulating the matching points

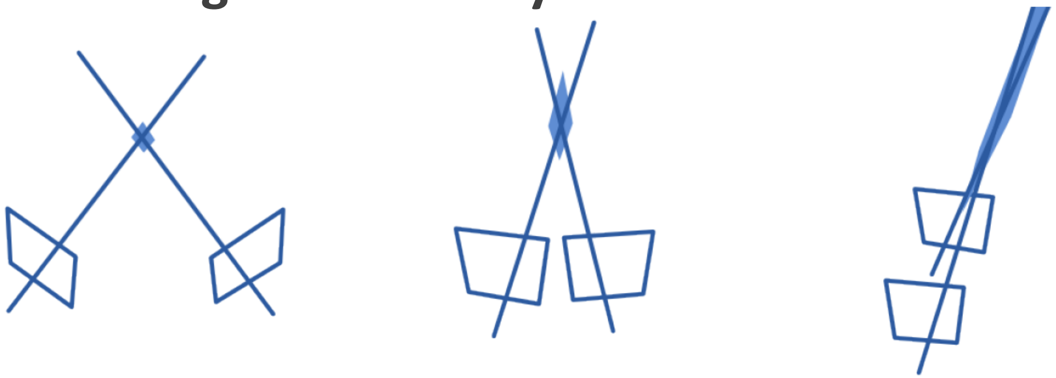


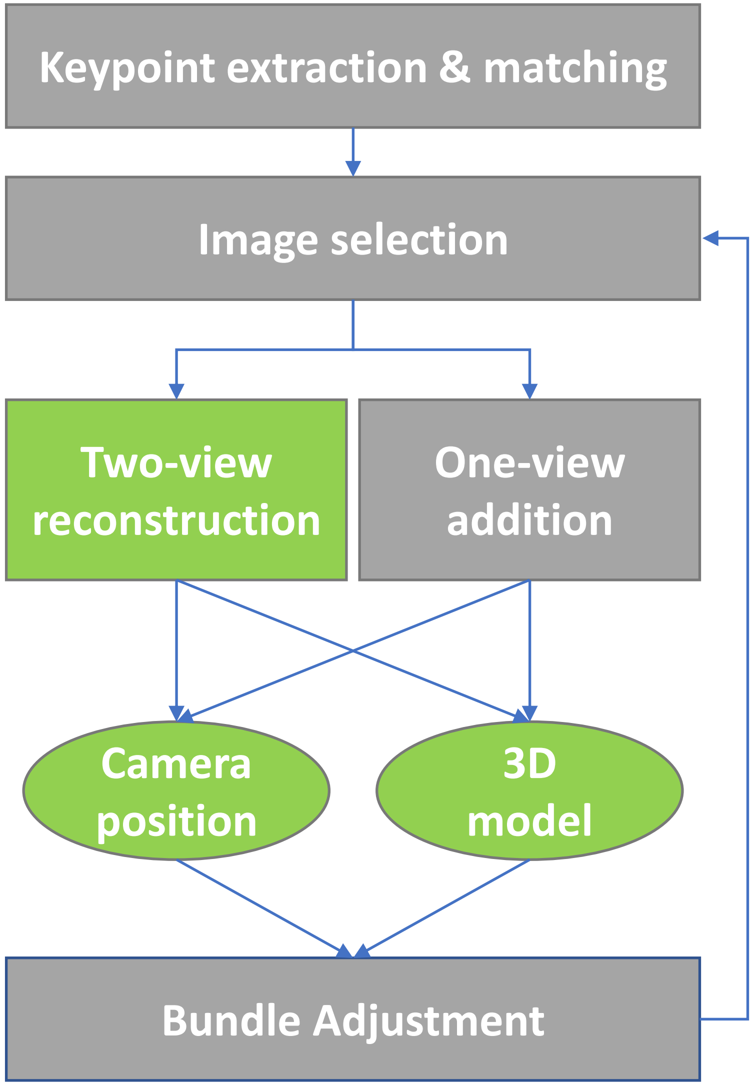
Triangulation

- Given a match ($\mathbf{x}_0$,$\mathbf{x}_1$) the relative 3D point $\mathbf{X}$ is obtained solving $A\mathbf{X} = 0$ where

$$A = \begin{bmatrix} x_0\mathbf{p}_0^3 - \mathbf{p}_0^1 \\ y_0\mathbf{p}_0^3 - \mathbf{p}_0^2 \\ x_1\mathbf{p}_1^3 - \mathbf{p}_1^1 \\ y_1\mathbf{p}_1^3 - \mathbf{p}_1^2 \end{bmatrix}$$

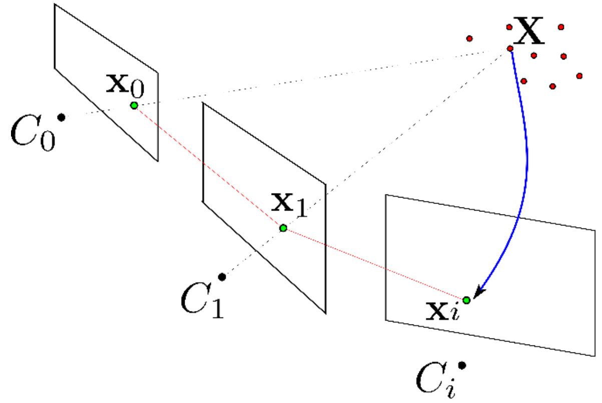
- Note that low disparity matches (e.g., from images with low baseline, points at infinity, etc.) can produce **3D points with high uncertainty**





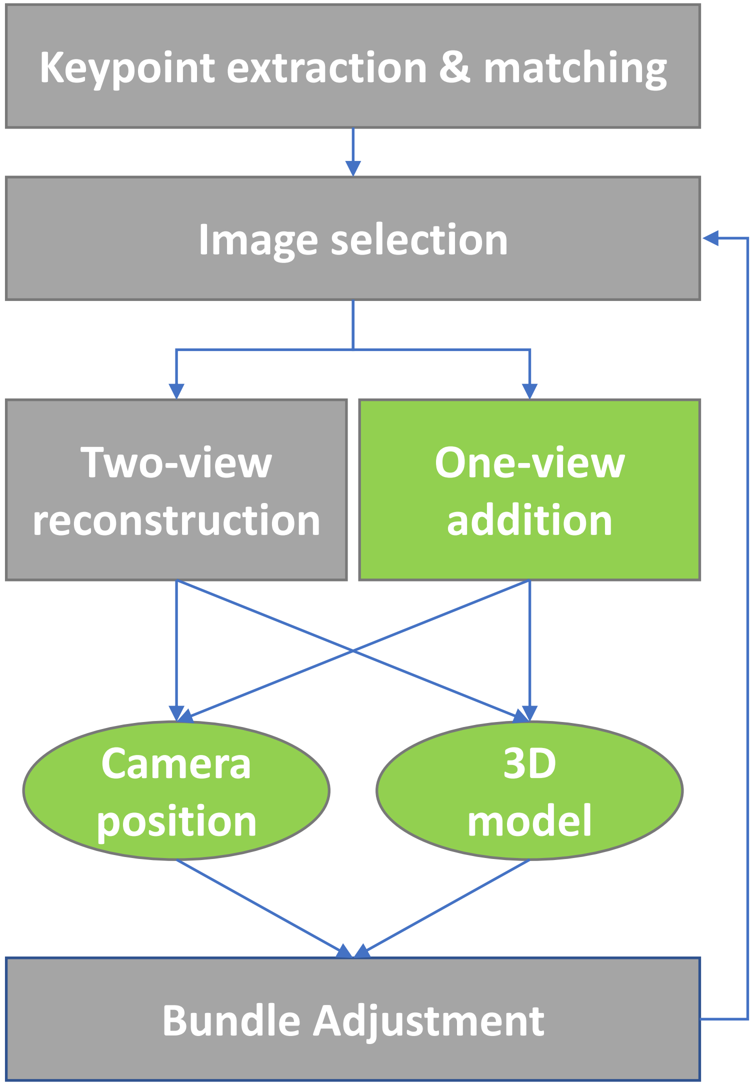
One-view addition

- Given the 3D model and the 2D tracks is possible to recover the 2D/3D matches



- The new camera matrix is estimated by solving an over-constrained linear system

$$\begin{bmatrix} \mathbf{0}^\top & -w_0\mathbf{X}_0^\top & y_0\mathbf{X}_0^\top \\ w_0\mathbf{X}_0^\top & \mathbf{0}^\top & -x_0\mathbf{X}_0^\top \\ & \vdots & \\ \mathbf{0}^\top & -w_n\mathbf{X}_n^\top & y_n\mathbf{X}_n^\top \\ w_n\mathbf{X}_n^\top & \mathbf{0}^\top & -x_n\mathbf{X}_n^\top \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \end{bmatrix} = \mathbf{0}$$



One-view addition

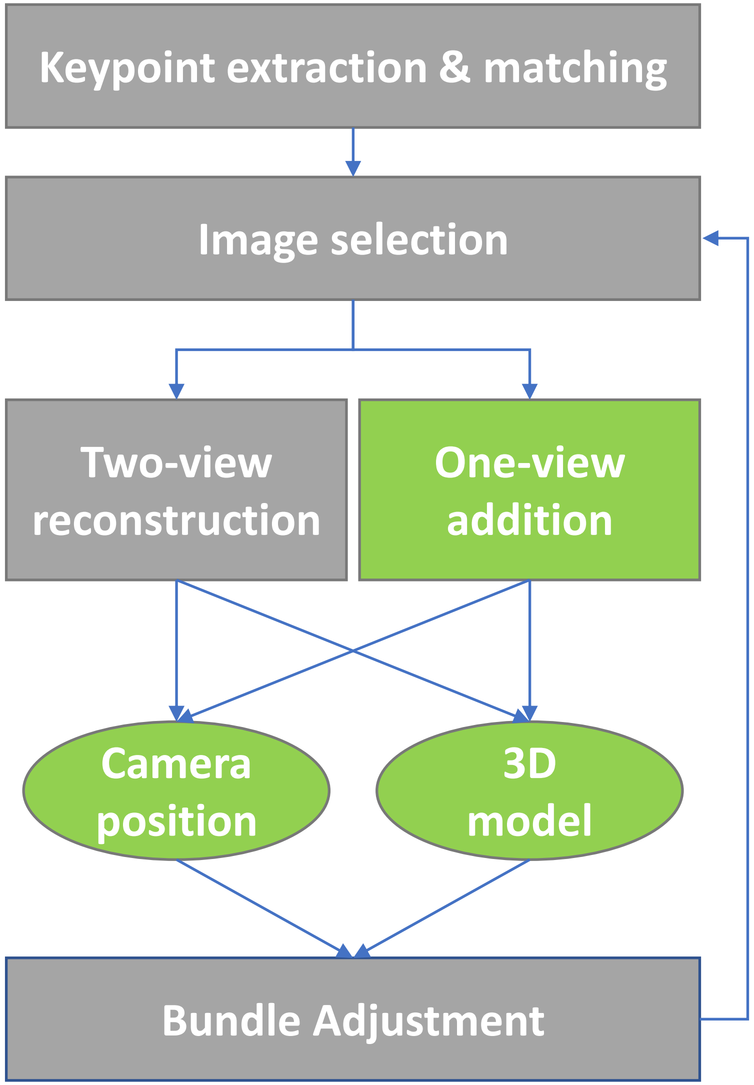
- If K is known, we can solve an **exterior orientation problem** to find *R* and **t**
- We have to minimize

$$\sum_{i=1}^{N} ||\mathbf{x}_i - \pi(\mathbf{X}_i; \mathbf{r}, \mathbf{t})||^2$$

where

$$\pi(\mathbf{X}_i; \mathbf{r}, \mathbf{t}) = K[R(\mathbf{r})|\mathbf{t}]\mathbf{X}_i$$

- This is a non-linear problem that can be solved by iterative minimization

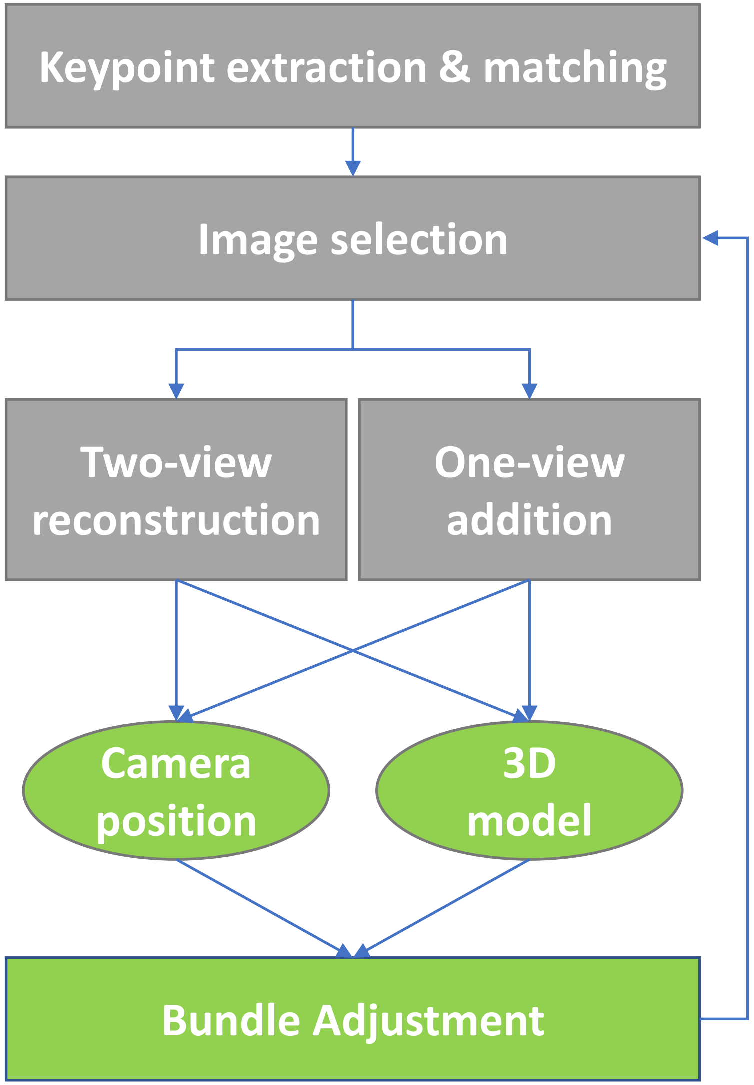


Optimization – bundle adjustment

- Bundle Adjustment is an iterative algorithm used to **minimize the local/global reprojection error**, by minimizing

$$\min_{P^i, \mathbf{X}_j} \sum_{i,j} ||P^i \mathbf{X}_j - \mathbf{x}_j^i||^2$$

- Computationally expensive:
  - m cameras P with 11 DoF
  - n 3D points X with 3 DoF
  - BA has to deal with **factorization** and **inversion** of matrices (3n+11m)x(3n+11m)
- To ease computation **interleaving techniques** can be used, as well as, exploiting the **sparsity of the matrix**
- Note: **BA can't deal with outliers!**



Triggs et al., "Bundle adjustment – a modern synthesis", 2000

SfM Results

SfM Results

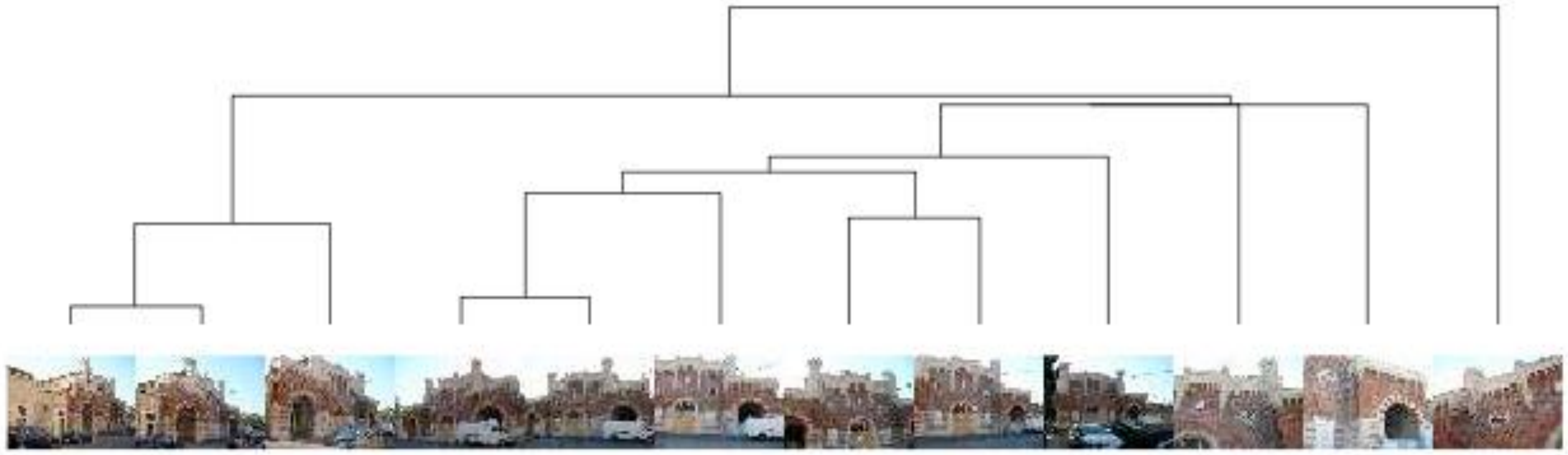
Multi-view stereo can be used to obtain a **denser** reconstruction

SfM Softwares

- The **OpenCV** library (C/C++, Python) includes functions to build a SfM pipeline

- There are also software **ready to use**, that do not require particular knowledge:
  - PhotoTurism: http://phototour.cs.washington.edu/
  - VisualSfM: http://ccwu.me/vsfm/
  - Colmap: https://colmap.github.io/
  - AliceVision: https://alicevision.org/
  - …

Hierarchical SfM



Hierarchical SfM is based on the identification and fusion of **clusters of images**

[1] A. M.Farenzena, A.Fusiello, R. Gherardi. "Structure-and-Motion Pipeline on a Hierarchical Cluster Tree." Workshop on 3-D Digital Imaging and Modeling, 2009.
[2] R. Gherardi, M. Farenzena, A. Fusiello. "Improving the efficiency of hierarchical structure-and-motion." CVPR , 2010.

Hierarchical SfM

1. At first keypoint matches are found among all the images
2. Each image forms a cluster, and cluster distance is measured as

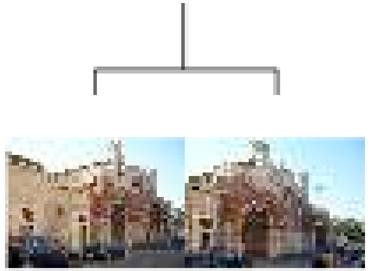
$$1 - a_{i,j} = 1 - \left[\frac{1}{2} \frac{|S_i \cap S_j|}{|S_i \cup S_j|} + \frac{1}{2} \frac{CH(S_i) + CH(S_j)}{A_i + A_j} \right]$$

   where S_* are the keypoint sets, *CH()* is the convex-hull, and A_* is the image area
3. Reconstruction starts from the leaves of the constructed dendrogram, and progressively climbs the tree until all images are included in a single model/cluster

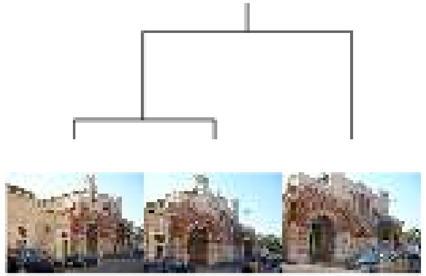
Hierarchical SfM

To merge two clusters A and B, we will face three different problems

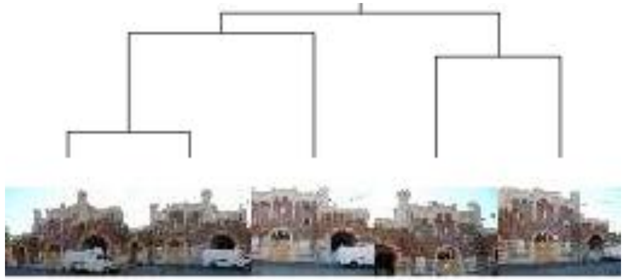
I. If both A and B include a single image, we can use the **decomposition of the essential matrix** to obtains camera poses, and then a local 3D map



II. If A include multiple images, and B only a single image (or vice-versa) we can solve an **exterior orientation problem** to add the image of B in the model of A



III. If both A and B include multiple images and we have already built a local model for A and B, thighs get a little bit trickier



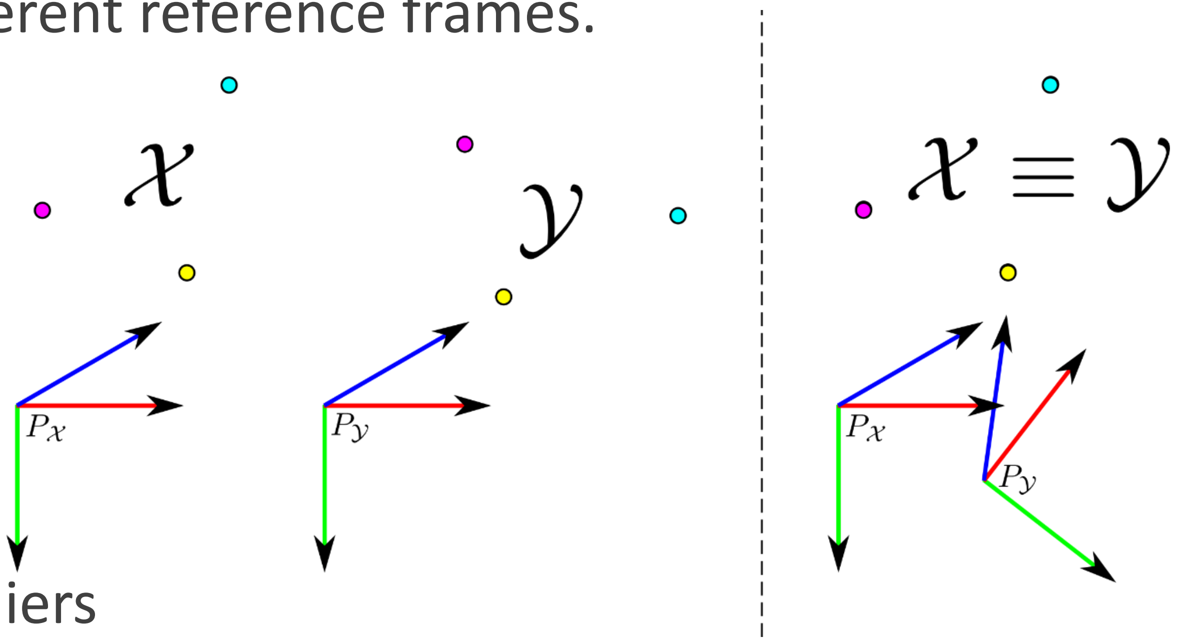
Hierarchical SfM

Cluster A and B have their model expressed in **different coordinate systems** and **different scale**. We can exploit the 3D points to register the two models:

- Let $\mathcal{X} = \{\mathbf{X}_i\}_0^n$ and $\mathcal{Y} = \{\mathbf{Y}_i\}_0^n$ be to set of n 3D points, with known correspondences, expressed into two different reference frames.
- To estimate the similarity transform to map *Y* onto *X* we can minimize

  $$\sum_{i=0}^{n} ||\mathbf{X}_i - (sR\mathbf{Y}_i + \mathbf{t})||^2$$

  in order to find *s*, *R* and **t**.
- Wrap the minimization into a RANSAC routine could help to discard possible outliers



3D Reconstruction

Multi-view stereo

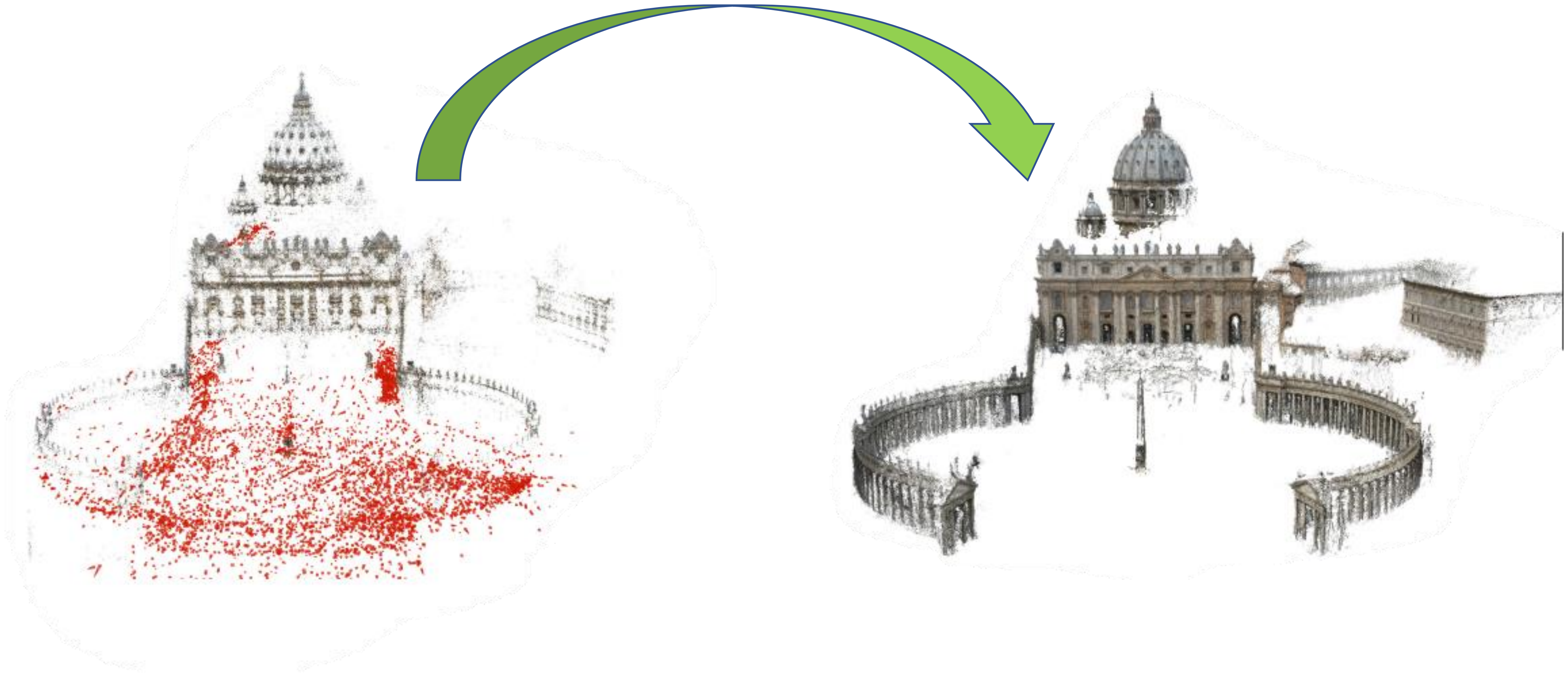
Multi-view stereo

- **Goal**: given a set of images with known camera poses obtain the **depth maps** for all the images

- As we have seen, camera poses can be obtained using **Structure from Motion** algorithms

- In order to obtain a dense 3D reconstruction, the **plane-sweeping algorithm** can be used

Multi-view stereo



Multi-view stereo

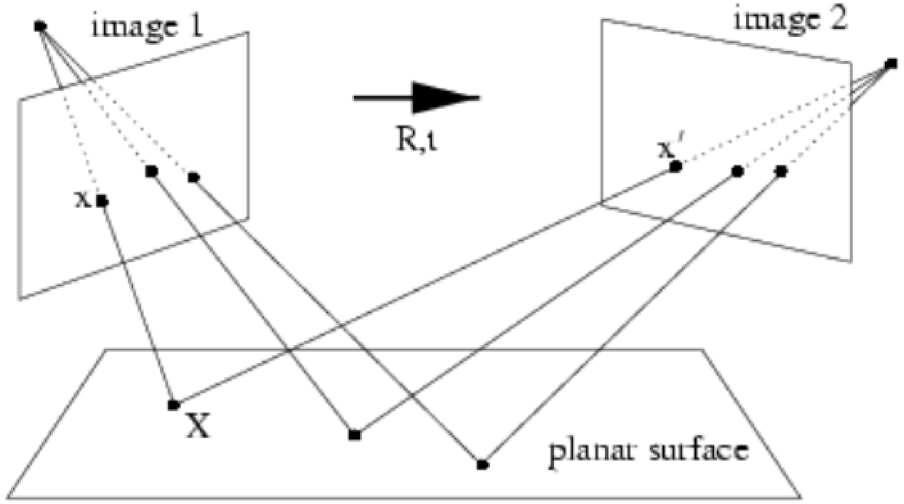
- We will consider
  - N camera views with $\mathrm{P}_n = \mathrm{K}_n[\mathrm{R}_n \ \mathbf{t}_n]$ with $\mathrm{P}_0 = \mathrm{K}_0[\mathrm{I} \ \ \mathbf{0}]$
  - M depth planes $\boldsymbol{\pi}_m = [\mathbf{n}_m \ \ -d_m]^\top$

- In case of front-to-parallel plane sweeping
  - $\mathbf{n}_m = [0\ 0\ 1]^\top$ and
  - $d_m = \{d_{near}, \dots, d_{far}\}$

Plane Sweeping

- Using the defined planes, we can use planar homographies to obtain additional correspondences

$$\mathrm{H}_{\pi_m,\mathrm{P}_n} = \mathrm{K}_n \left(\mathrm{R}_n + \frac{\mathbf{t}_n \mathbf{n}_m^\top}{d_m} \right) \mathrm{K}_0^{-1} ,$$



Plane Sweeping

- Using the defined planes, we can use planar homographies to obtain additional correspondences

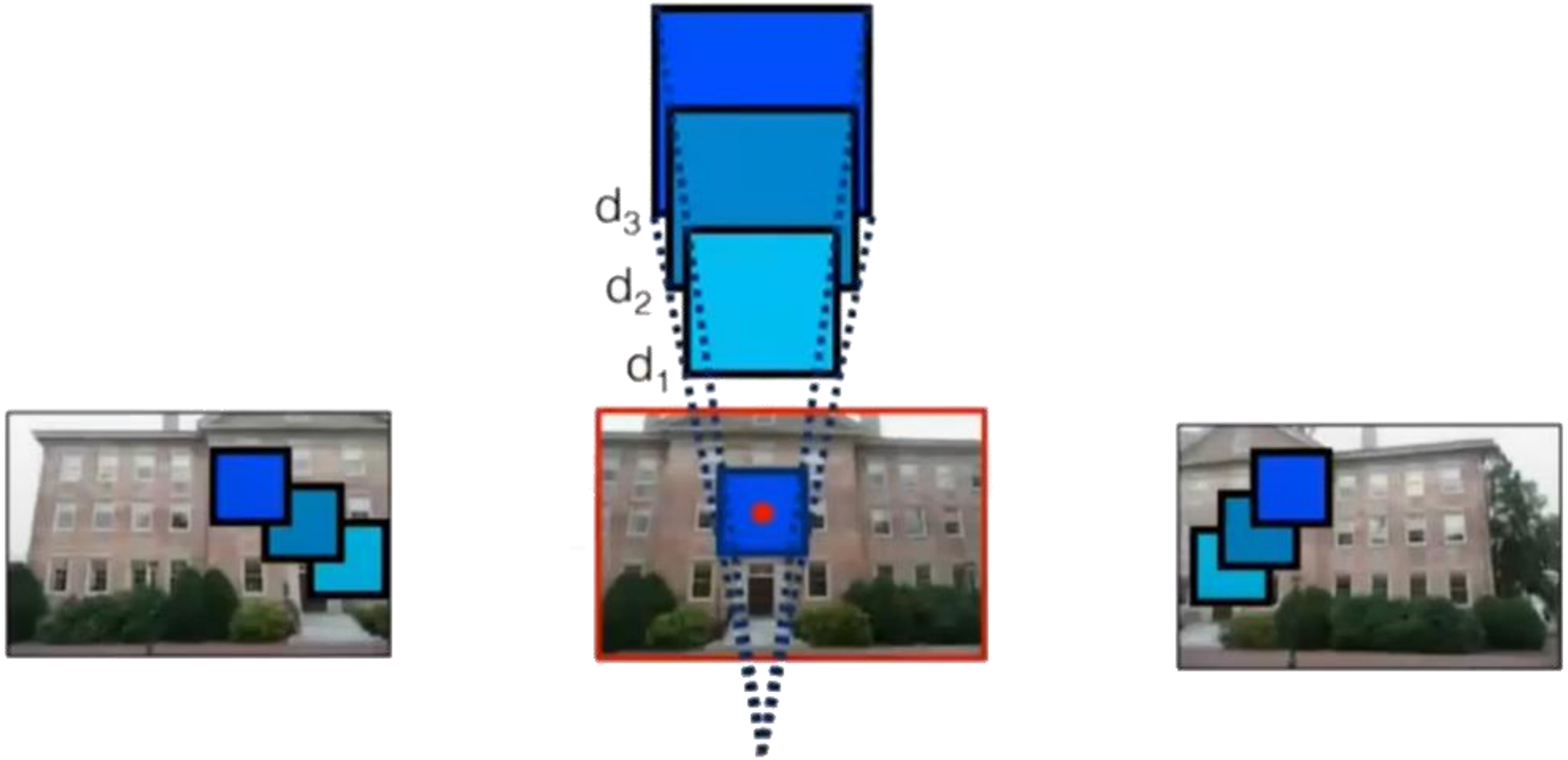
$$\mathrm{H}_{\pi_m,\mathrm{P}_n} = \mathrm{K}_n \left(\mathrm{R}_n + \frac{\mathbf{t}_n \mathbf{n}_m^\top}{d_m} \right) \mathrm{K}_0^{-1} \,,$$

- So, a point on the first image can be matched with a point on the n image using

$$[\tilde{x}, \tilde{y}, \tilde{w}]^\top = \mathrm{H}_{\pi_m,\mathrm{P}_n} [x, y, 1]^\top$$

Plane Sweeping

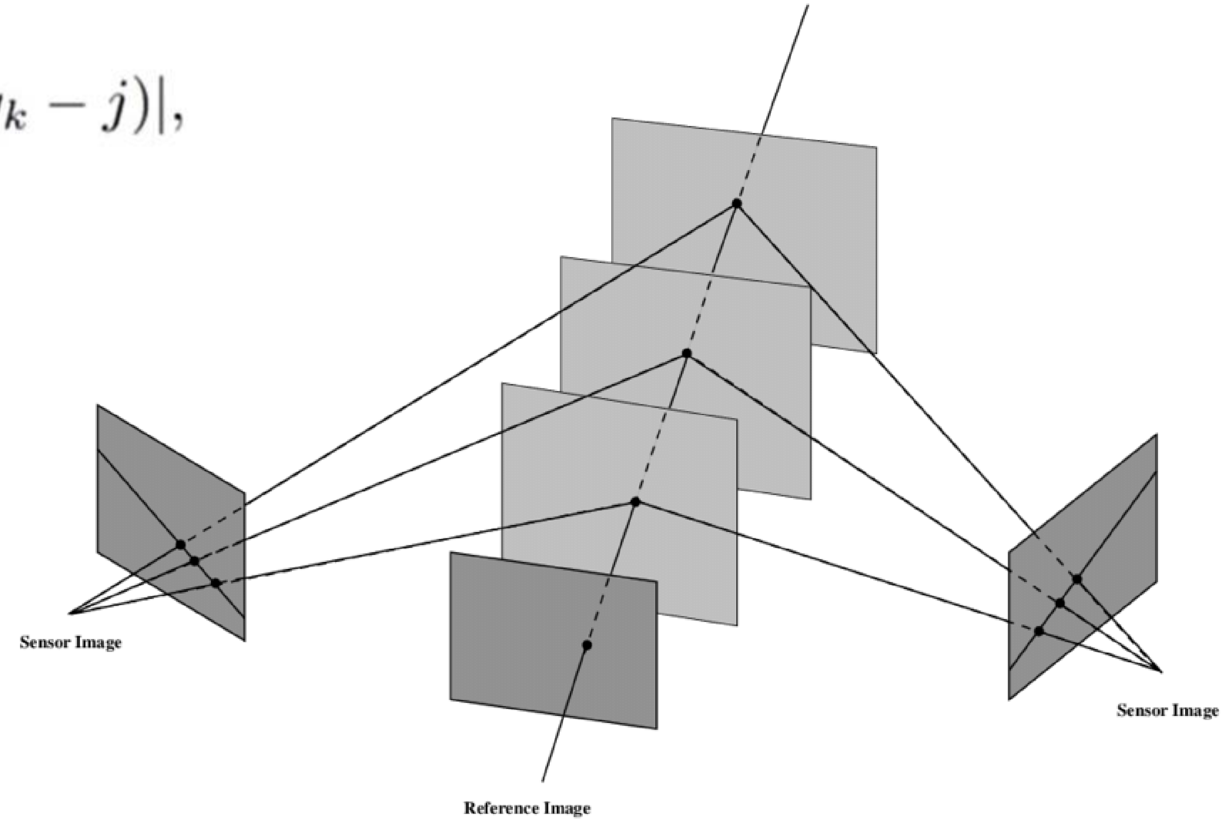
- Obtained putative correspondences must be validated
- More exactly, we have to find which plane $\boldsymbol{\pi}_m = [\mathbf{n}_m \quad -d_m]^\mathsf{T}$ is the one that really maps a point on the reference image on the other images



Plane Sweeping

- We can define a cost function such as

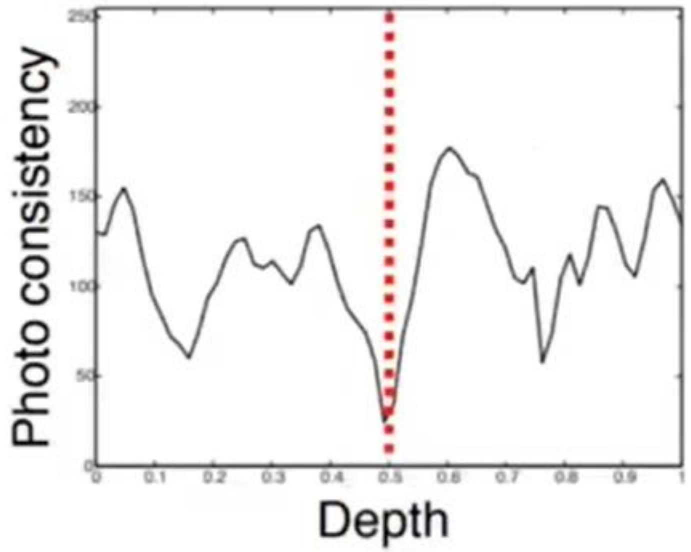
$$C(x, y, \Pi_k) = \sum_{k=0}^{N-1} \sum_{(i,j)\in W} |I_{ref}(x-i, y-j) - \beta_k^{ref} I_k(x_k - i, y_k - j)|,$$



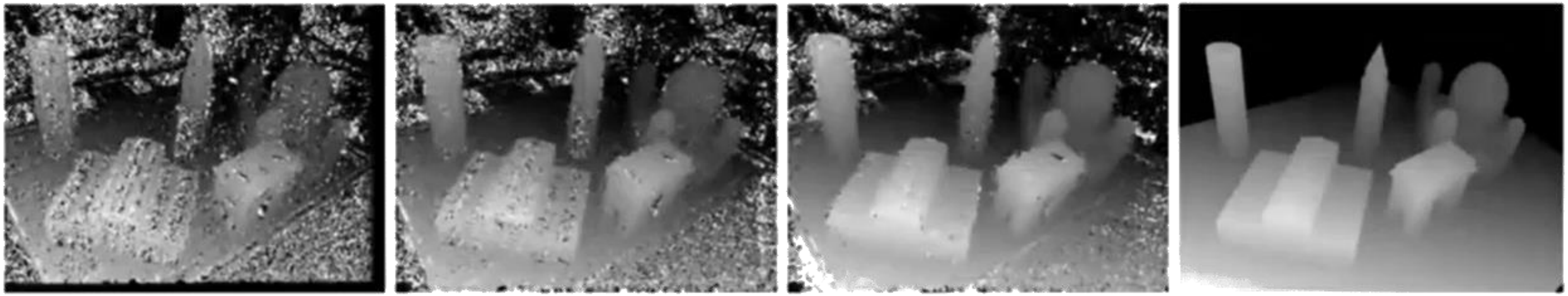
Plane Sweeping

- By evaluating the function on the different planes, we can find which one minimize the cost function

$$\tilde{\Pi}(x, y) = \underset{\Pi_m}{\operatorname{argmin}}\, C(x, y, \Pi_m)$$



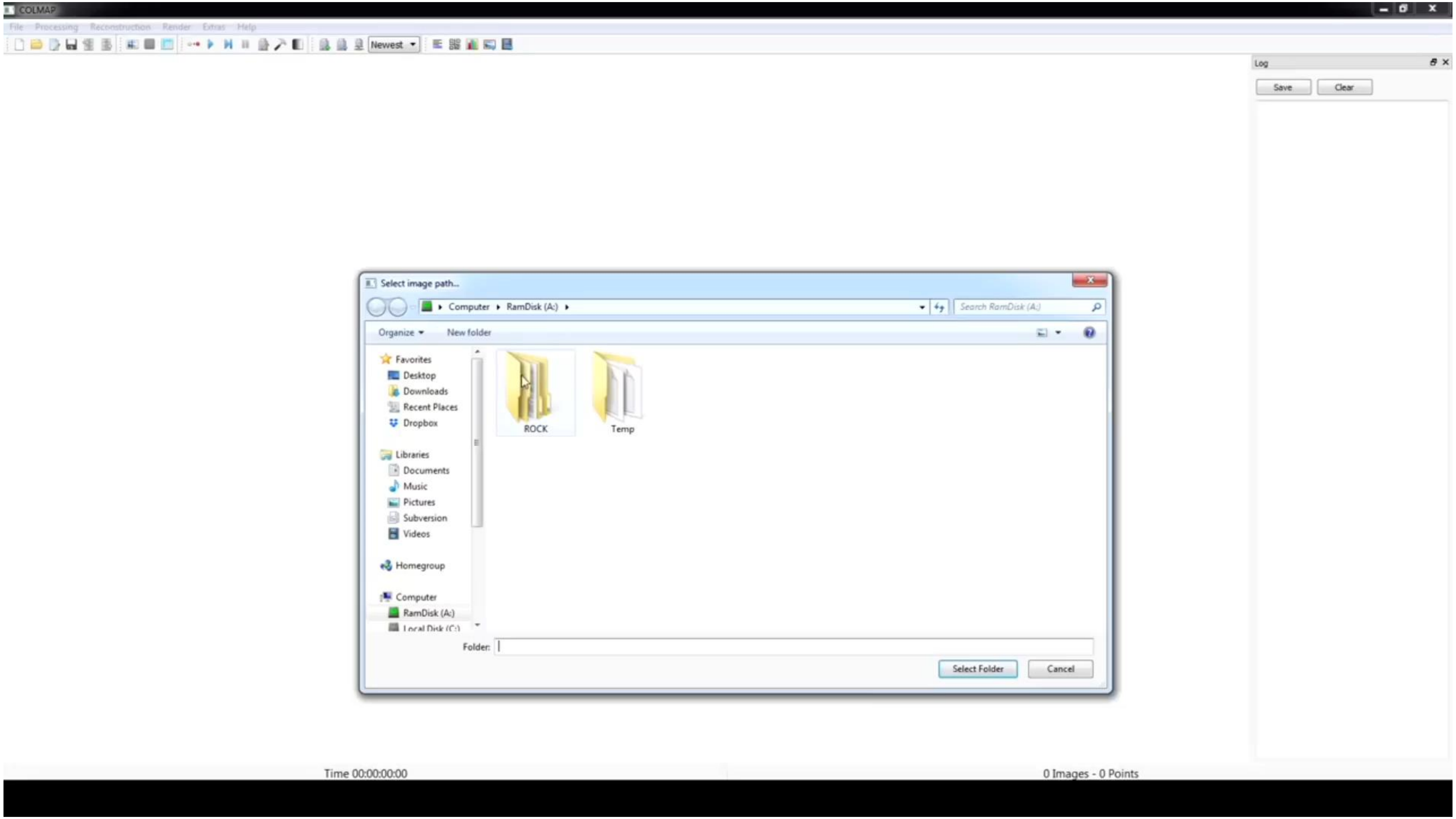
Plane Sweeping



(a) 2 views (b) 5 views (c) 20 views (d) Ground Truth

Newcombe, 2013

Colmap + Meshlab



https://www.youtube.com/watch?v=bDHJM6nAKtc

3D Reconstruction

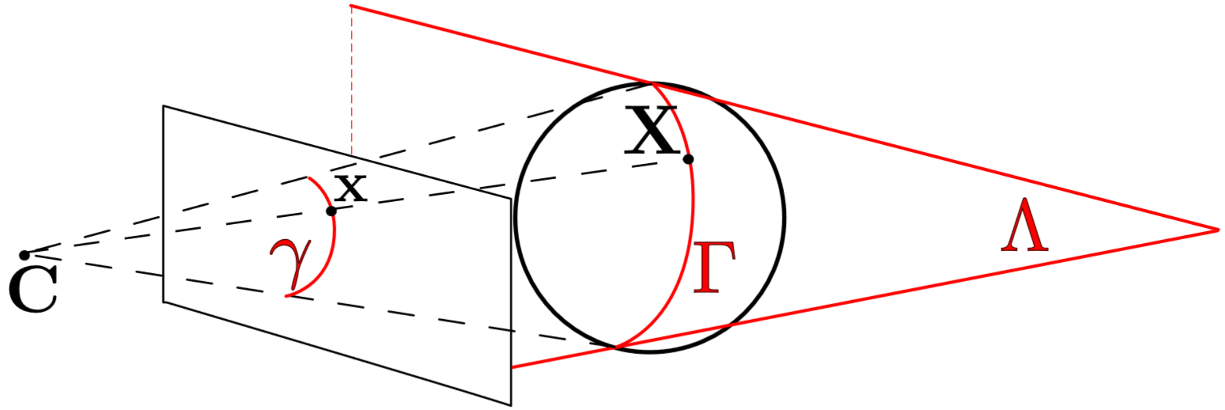
Structured Light

Structured Light

- 3D reconstruction can be achieved by using a **camera** in combination with an **active element**, such as a laser projector

Structured Light

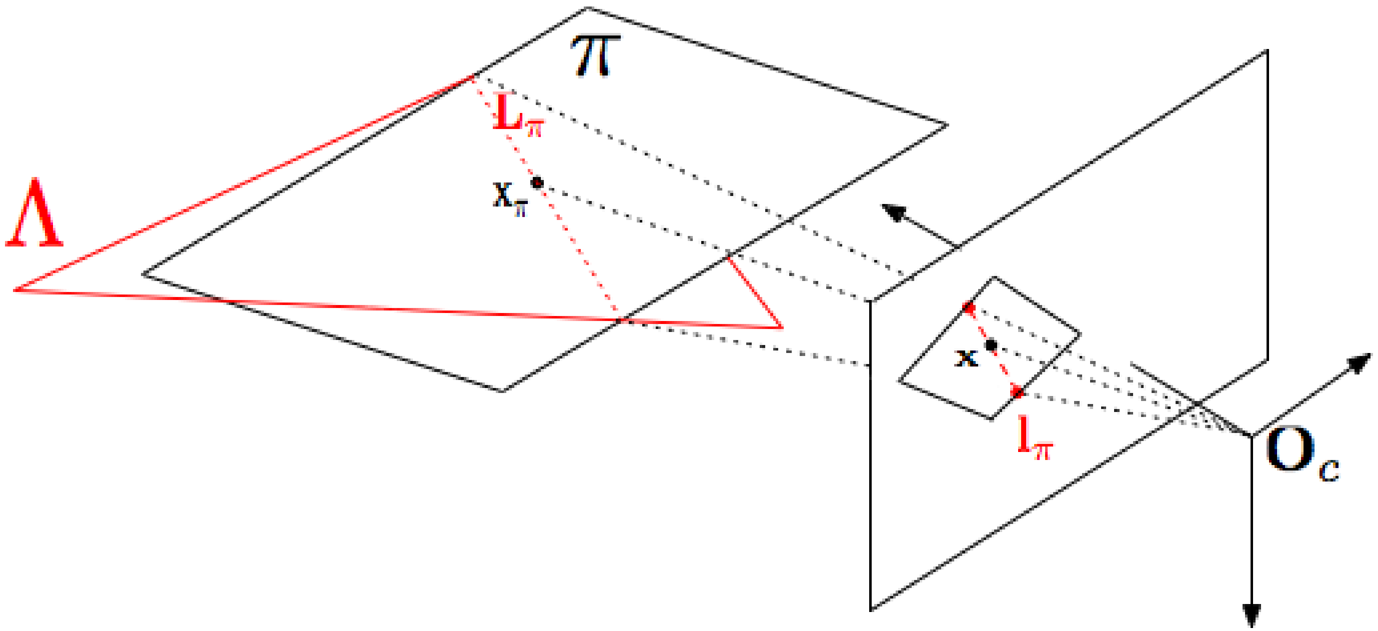
- 3D reconstruction can be achieved by using a **camera** in combination with an **active element**, such as a laser projector



- Intuitively, we exploit the knowledge of the **structured light pattern** projected onto the scene to estimate the depth of the pixels by observing the pattern deformation

Structured Light

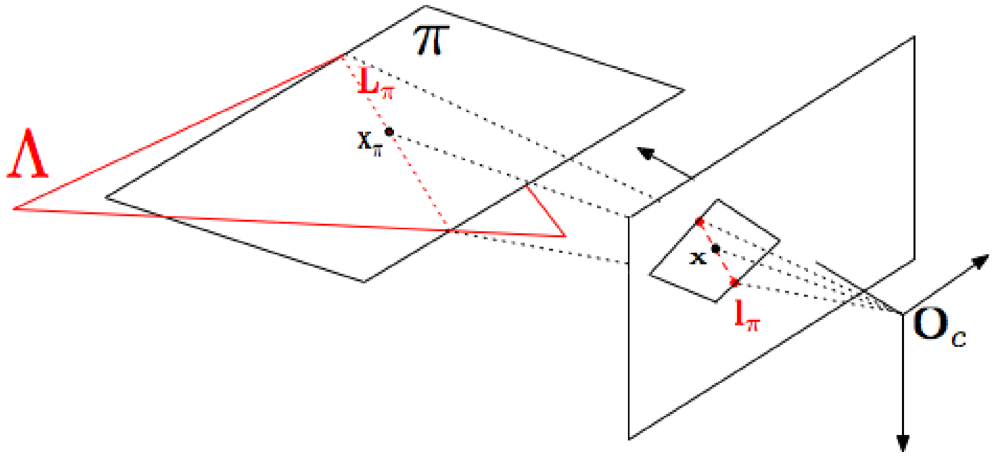
Laser plane calibration



- We use the camera-laser device to scan an object with known geometry, for example a plane π

Structured Light

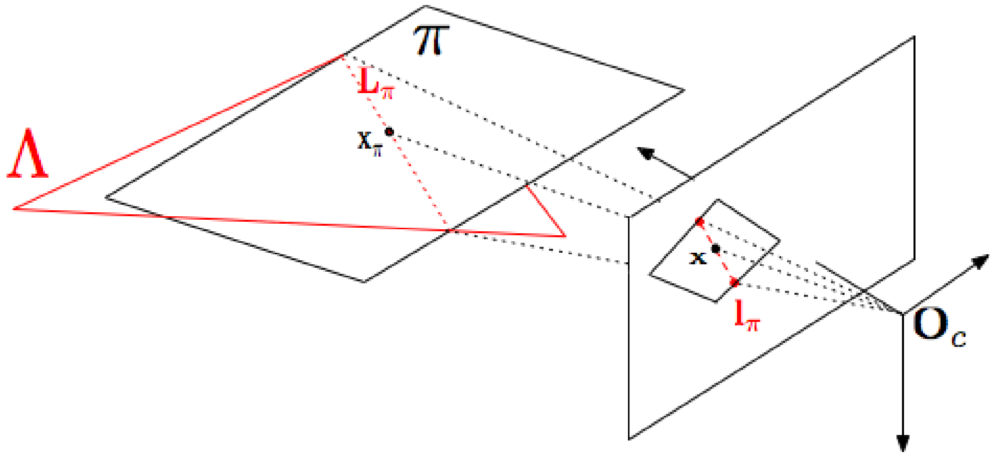
Laser plane calibration



1. Firstly, we must estimate the planar homography H_π between the 3D plane π and the image

Structured Light

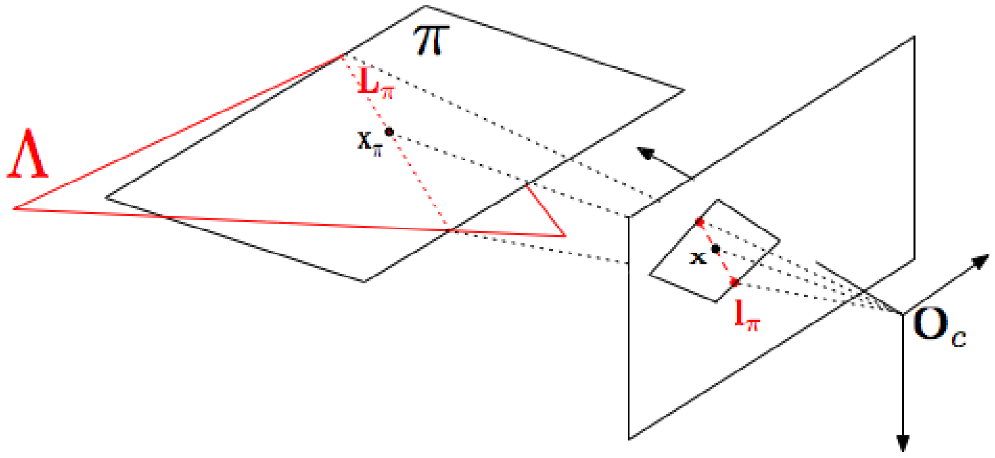
Laser plane calibration



1. Firstly, we must estimate the planar homography H_π between the 3D plane π and the image
2. Then, the laser line $\mathbf{l}_\pi$ on the image must detected

Structured Light

Laser plane calibration

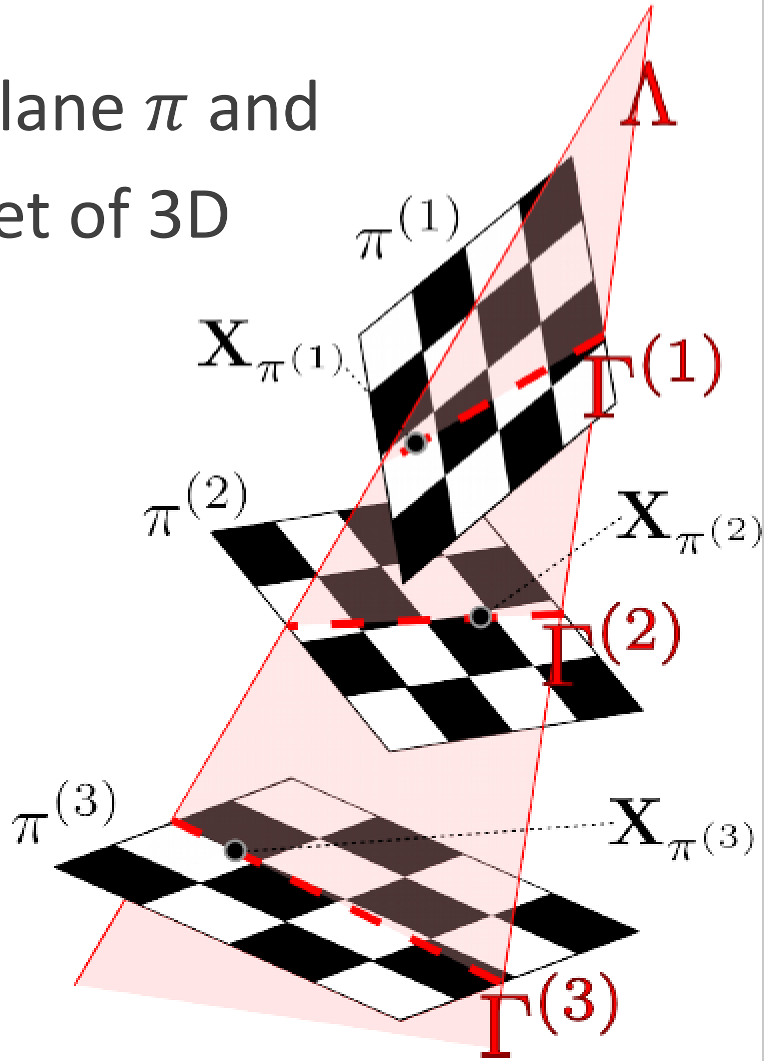


1. Firstly, we must estimate the planar homography H_π between the 3D plane π and the image
2. Then, the laser line $\mathbf{l}_\pi$ on the image must detected
3. All the points $\mathbf{x} \in \mathbf{l}_\pi$ can be reprojected on the points $\mathbf{X}_\pi \in \mathbf{L}_\pi$ using H_π

Structured Light

Laser plane calibration

4. By repeating the steps 1-3 after moving the 3D plane π and leaving fixed the camera-laser, we can collect a set of 3D points $\{\mathbf{X}_\pi\}$

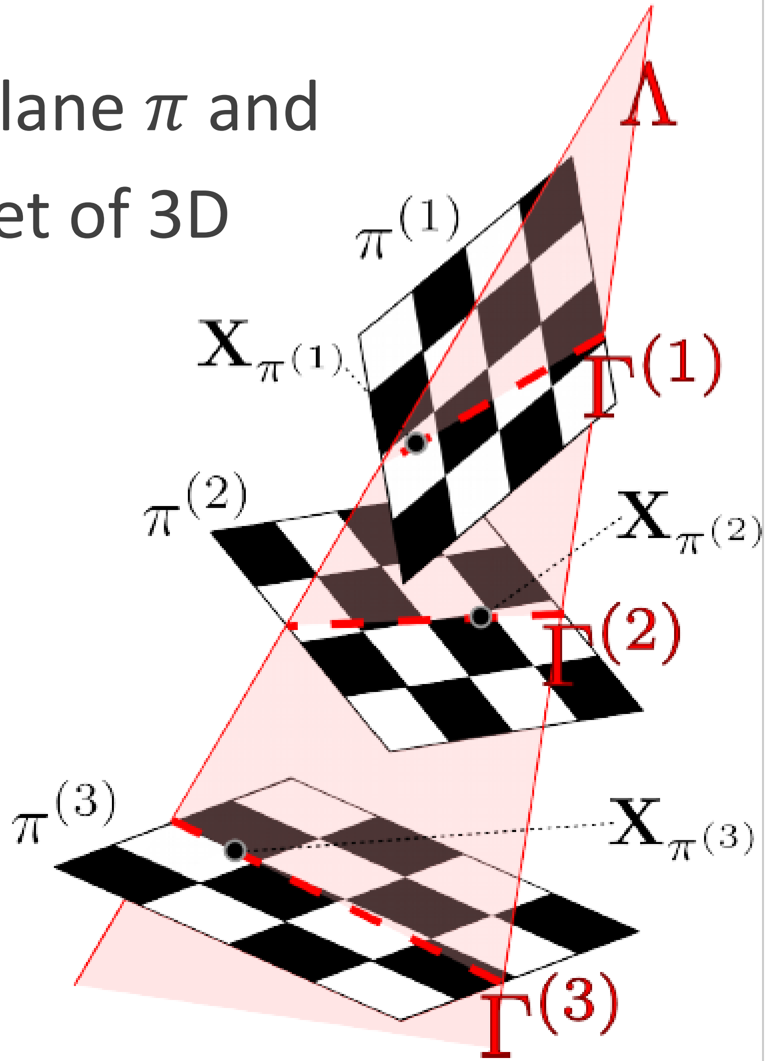


Structured Light

Laser plane calibration

4. By repeating the steps 1-3 after moving the 3D plane π and leaving fixed the camera-laser, we can collect a set of 3D points $\{\mathbf{X}_\pi\}$
5. Since if a 3D point X belong to a plane Λ with equation $[\mathbf{n}_\Lambda \ \mathrm{d}_\Lambda]^\top$ then

$$[\mathbf{n}_\Lambda \ \mathrm{d}_\Lambda]^\top \mathbf{X} = 0$$



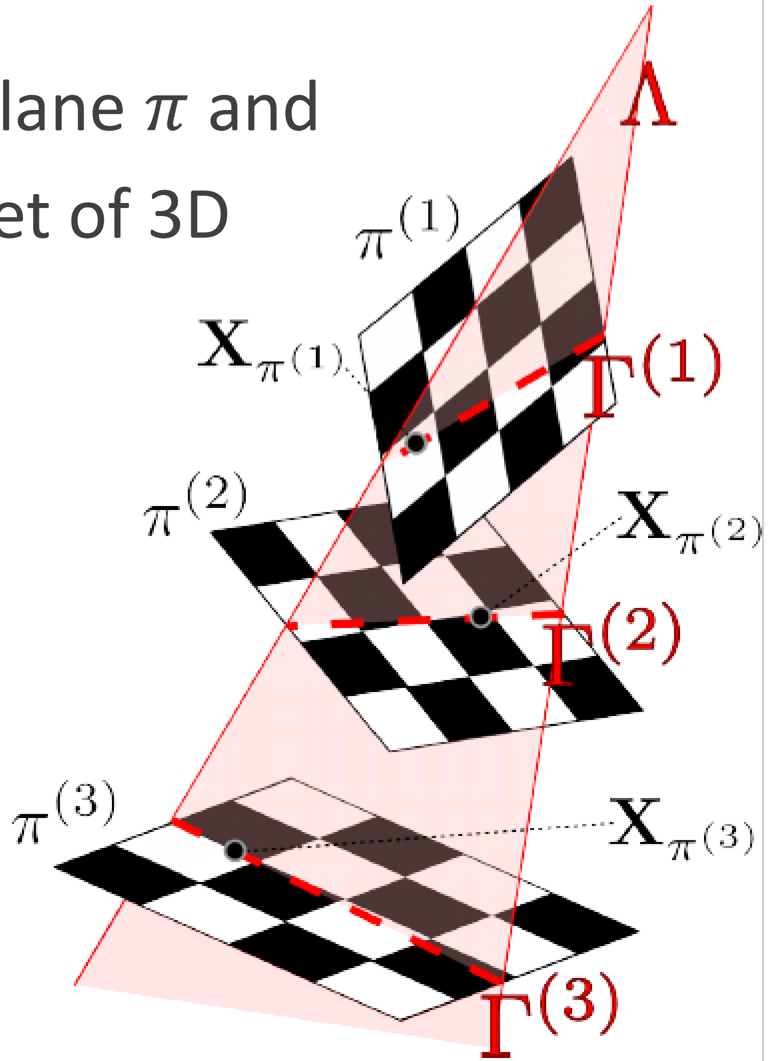
Structured Light

Laser plane calibration

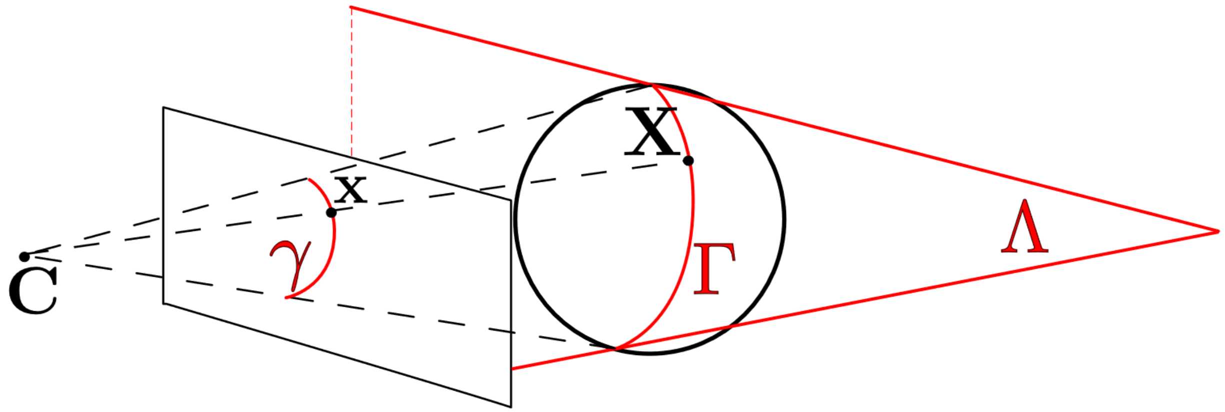
4. By repeating the steps 1-3 after moving the 3D plane π and leaving fixed the camera-laser, we can collect a set of 3D points $\{\mathbf{X}_\pi\}$
5. Since if a 3D point X belong to a plane Λ with equation $[\mathbf{n}_\Lambda \ \mathrm{d}_\Lambda]^\top$ then

$$[\mathbf{n}_\Lambda \ \mathrm{d}_\Lambda]^\top \mathbf{X} = 0$$

With at least three non-collinear points we can estimate the equation of the laser plane Λ



Structured Light



- Now, we know the laser plane Λ equation in the camera coordinate plane Λ: $[\mathbf{n}_\Lambda^\top \;\; d_\Lambda]$, such that for each point $\mathbf{X} \in \Lambda$ we have $\mathbf{n}_\Lambda^\top \mathbf{X} + d_\Lambda = 0$
- We know that $\mathbf{X} = \mu \mathrm{K}^{-1}\mathbf{x}$, where $\mu \in \mathbb{R}$ is the depth of $\mathbf{X}$. Then

$$\mathbf{n}_\Lambda^\top \mathbf{X} + d_\Lambda = \mathbf{n}_\Lambda^\top(\mu \mathrm{K}^{-1}\mathbf{x}) + d_\Lambda = 0 \;\Leftrightarrow\; \mu = \frac{-d_\Lambda}{\mathbf{n}_\Lambda^\top \mathrm{K}^{-1}\mathbf{x}}$$

and, finally

$$\mathbf{X} = \left(\frac{-d_\Lambda}{\mathbf{n}_\Lambda^\top \mathrm{K}^{-1}\mathbf{x}}\right)\mathrm{K}^{-1}\mathbf{x}$$

Structured Light

- From each image we can extract the 3D positions of the points highlighted by the laser plane

- In order to build the model, the camera-laser device must be moved in front of the object so to accumulate different laser stripes and the related 3D points

- To obtain a full 3D model the camera motion must be estimated, by using
  - SfM-like solution
  - Homography decomposition

Structured Light

- Indeed, if we have the homography H_π and the camera is calibrated, the position of the camera can be obtained as

$$\frac{1}{\mu}\mathbf{x} = \mathtt{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X_1\\X_2\\0\\1\end{bmatrix} = \mathtt{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X_1\\X_2\\1\end{bmatrix}, \quad \mu \in \mathbb{R}^+$$

$$\Downarrow$$

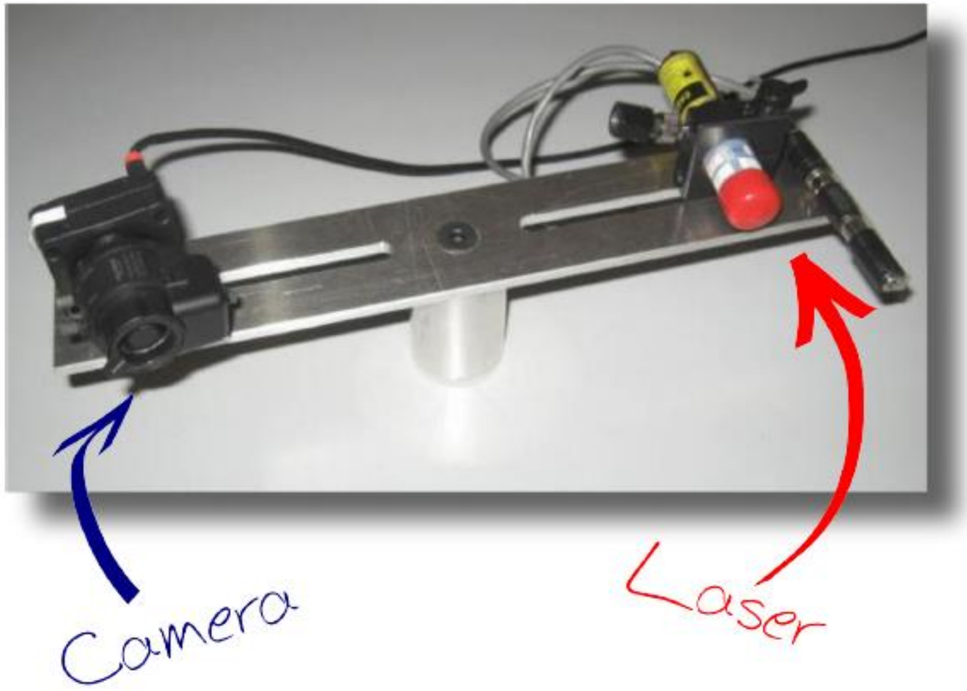
$$\mathtt{H} = \mu\mathtt{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}$$

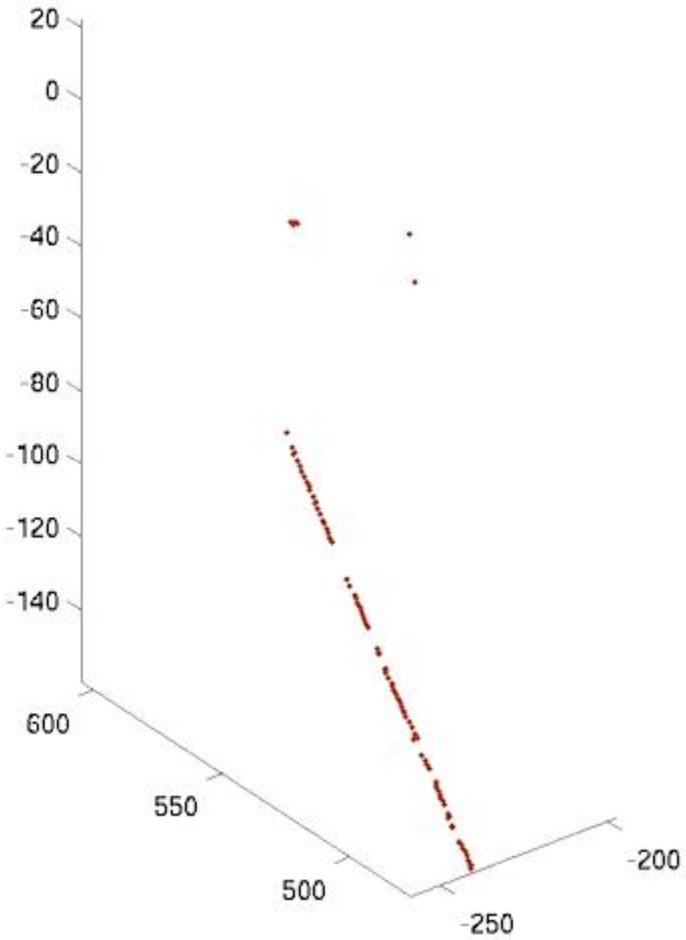
- So given $\mathrm{H}_\pi = [h_1 h_2 h_3]$,

$$\mathbf{r}_j = \frac{1}{\mu}\mathtt{K}^{-1}\mathbf{h}_j, \quad j = 1,2 \qquad \mathbf{t} = \frac{1}{\mu}\mathtt{K}^{-1}\mathbf{h}_3$$

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \qquad \mu = ||\mathtt{K}^{-1}\mathbf{h}_1||$$

Structured Light





http://cvg.dsi.unifi.it/FRATINO_BUILD.mp4

Structured Light

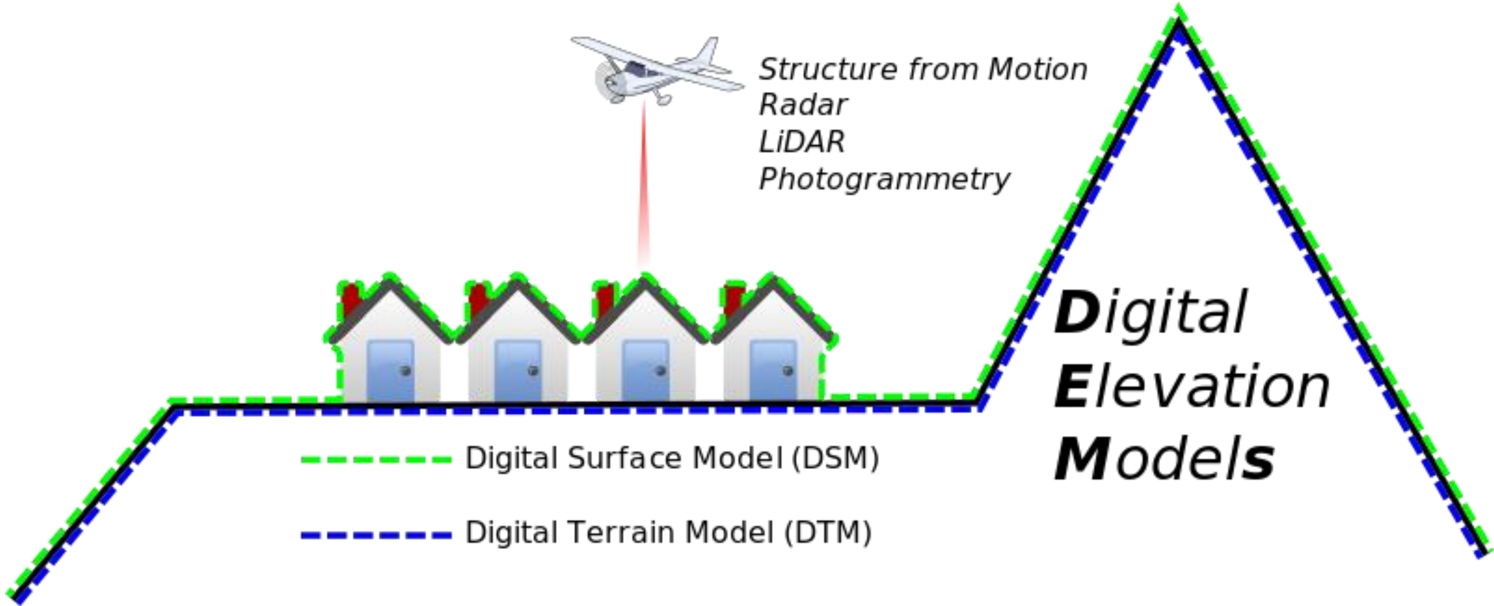
https://www.youtube.com/watch?v=DWc9AzbrJDk&t=23s

3D Reconstruction

Digital Elevation Model

DEM, DTM, DSM

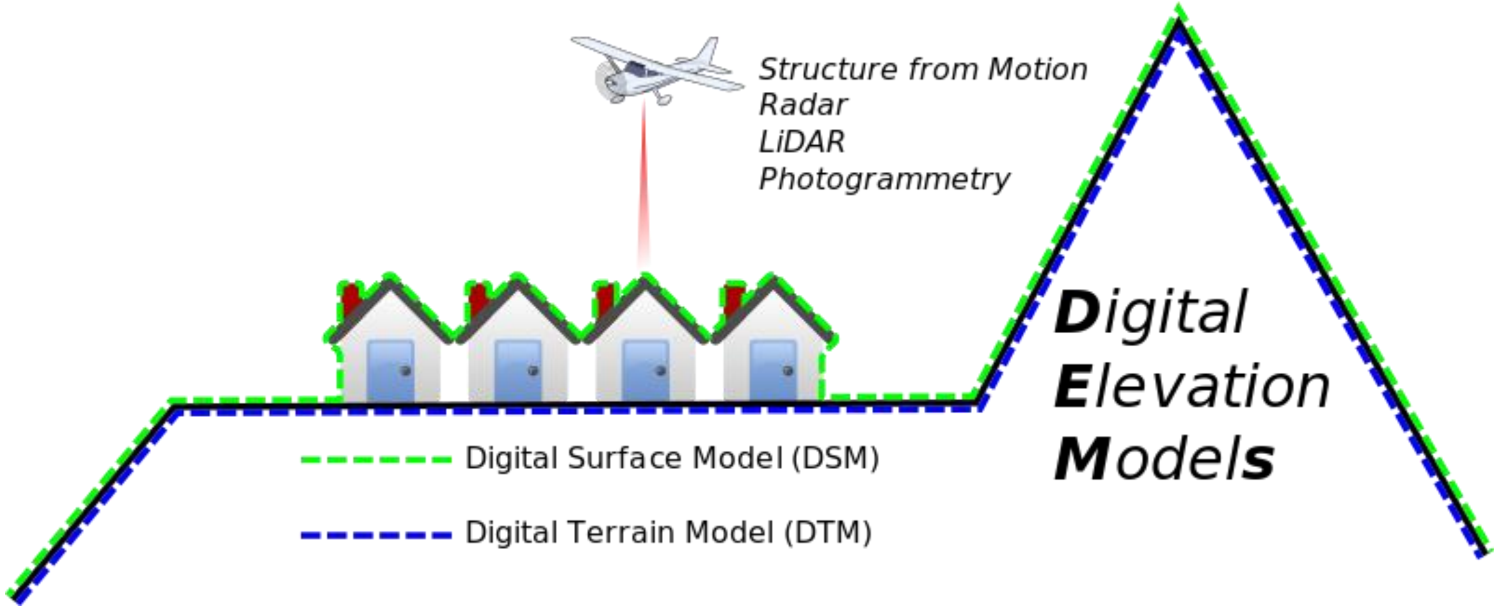
- DEM is obtained by aerial acquisition using
  - Lidar
  - Radar
  - Photogrammetry



DEM, DTM, DSM

- DEM is obtained by aerial acquisition using
  - Lidar
  - Radar
  - Photogrammetry

- DEM is composed by
  - DTM: Digital **Terrain** Model
  - DSM: Digital **Surface** Model



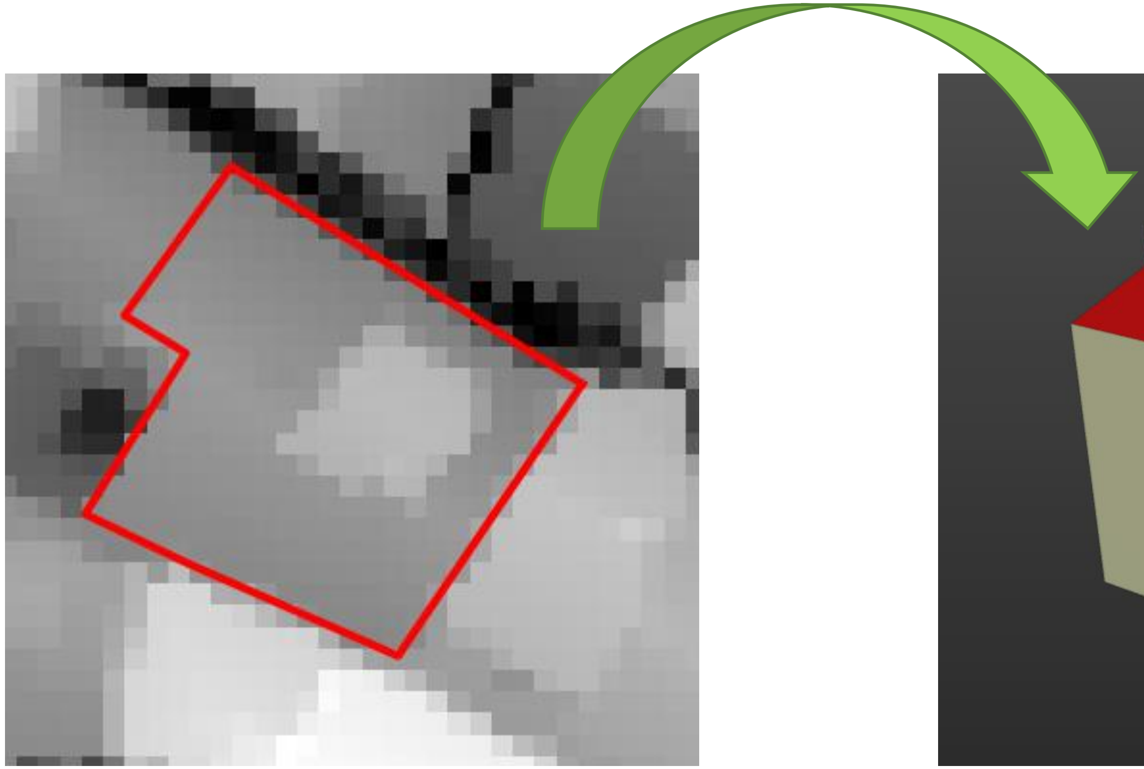
DTM

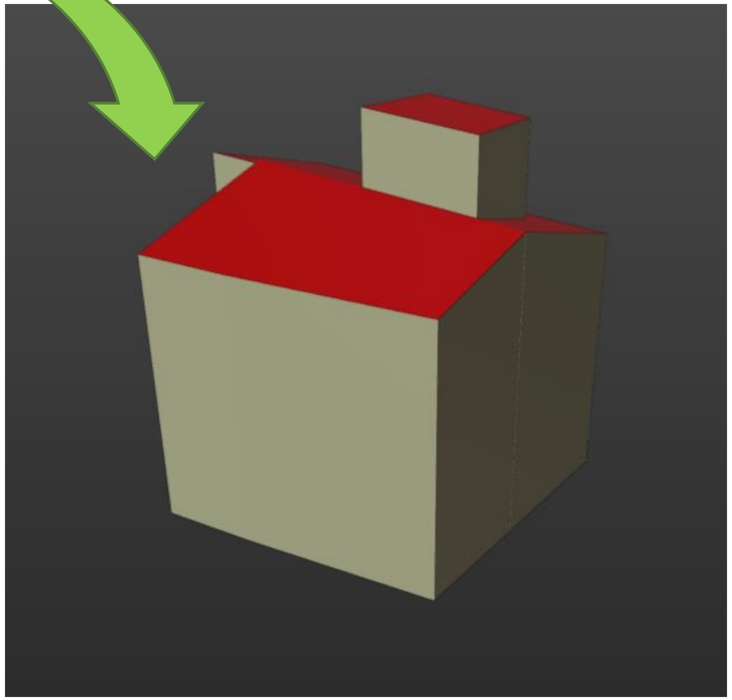
Via Francesco Puccinotti
Viale Giovanni Milton
Fratelli Rosselli
The Student Hotel Firenze Belfiore
Monumento a Garibaldi
Fiume Arno
Ponte Amerigo Vespucci
Via dei Banchi
Via del Corso
Borgo degli Albizi
Via Pietrapiana
Ponte al Pino
Piazza D'Azeglio

DSM

DSM Modelling

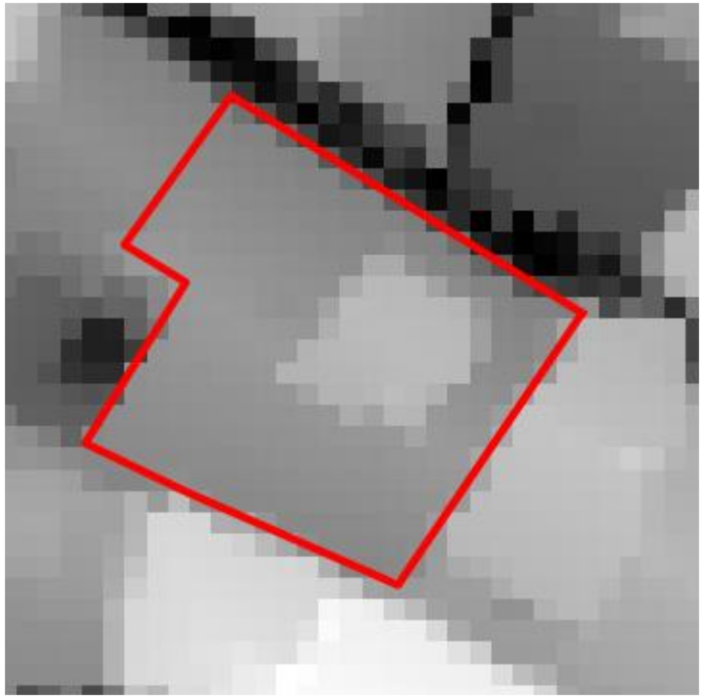
- From the DSM is possible to obtain the 3D building shape





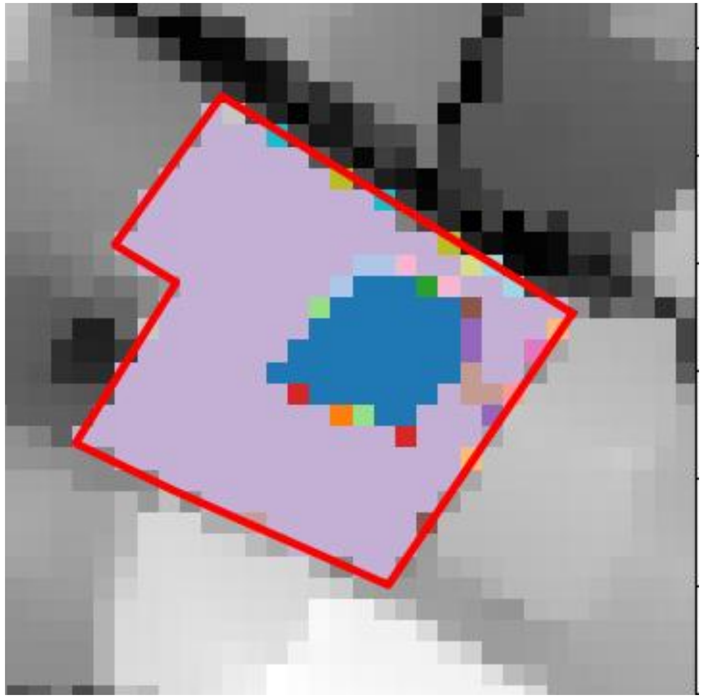
DSM Modelling

- From the DSM is possible to obtain the 3D building shape
  1. Buildings cadastral maps used to segment the DSM



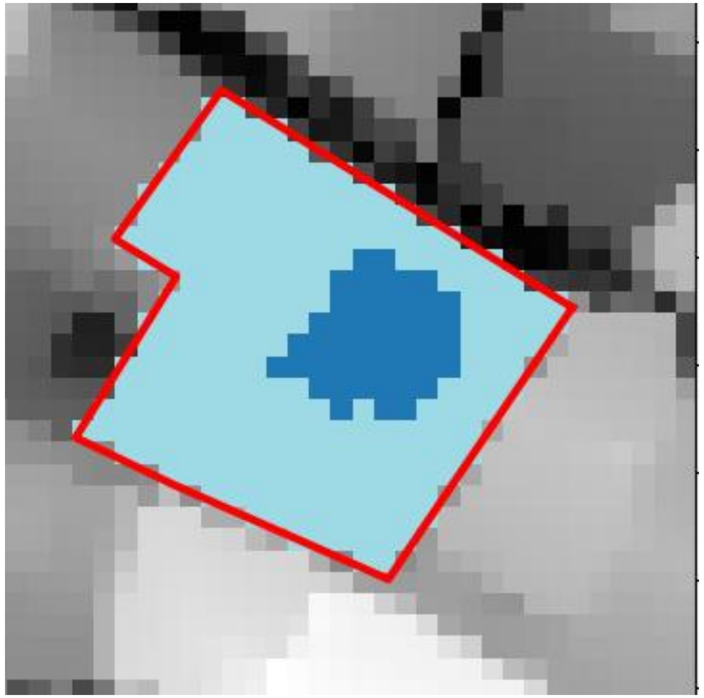
DSM Modelling

- From the DSM is possible to obtain the 3D building shape
  1. Buildings cadastral maps used to segment the DSM
  2. Region-Growing algorithm to cluster different building elevation



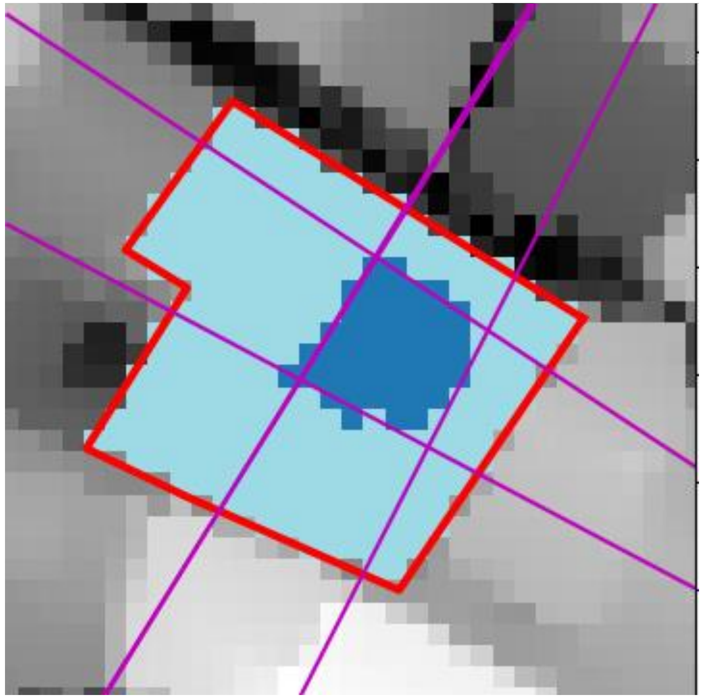
DSM Modelling

- From the DSM is possible to obtain the 3D building shape
  1. Buildings cadastral maps used to segment the DSM
  2. Region-Growing algorithm to cluster different building elevation
  3. Small cluster aggregation



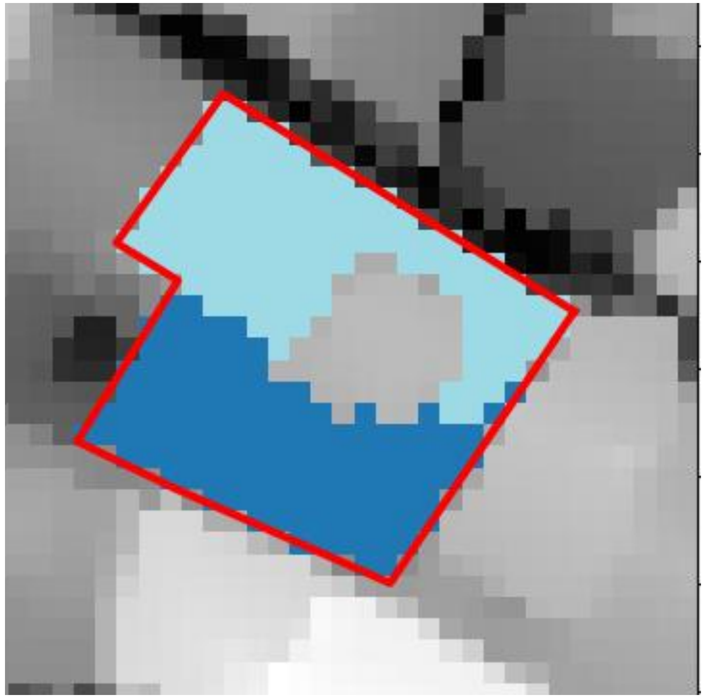
DSM Modelling

- From the DSM is possible to obtain the 3D building shape
  1. Buildings cadastral maps used to segment the DSM
  2. Region-Growing algorithm to cluster different building elevation
  3. Small cluster aggregation
  4. Step-line regression



DSM Modelling

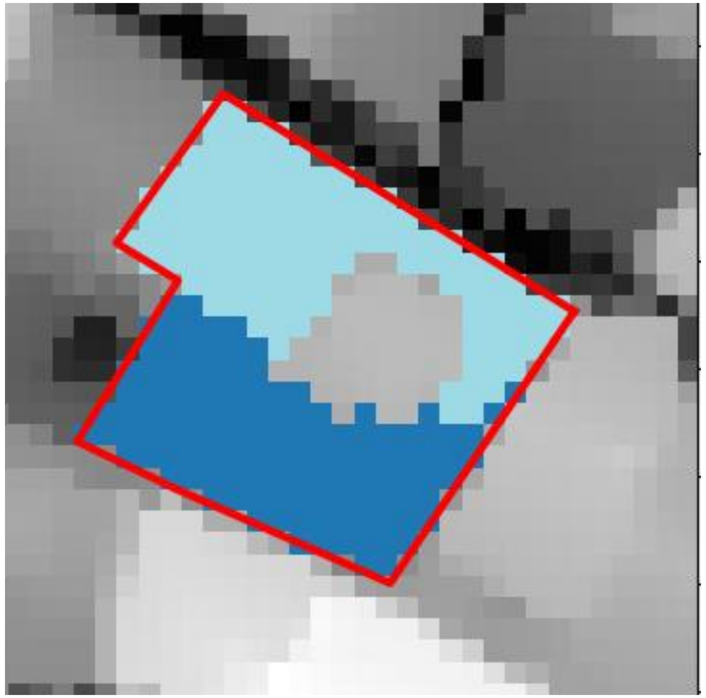
- From the DSM is possible to obtain the 3D building shape
  1. Buildings cadastral maps used to segment the DSM
  2. Region-Growing algorithm to cluster different building elevation
  3. Small cluster aggregation
  4. Step-line regression
  5. HDBSCAN algorithm based on a custom weight matrix W= {w_ij }



$$w_{ij} = \begin{cases} \infty, & if\ L_{ij} > 6 \\ w_N N_{ij} + w_M M_{ij} + w_D D_{ij} + w_L L_{ij}, & otherwise \end{cases}$$

DSM Modelling

- From the DSM is possible to obtain the 3D building shape
  1. Buildings cadastral maps used to segment the DSM
  2. Region-Growing algorithm to cluster different building elevation
  3. Small cluster aggregation
  4. Step-line regression
  5. HDBSCAN algorithm based on a custom weight matrix W= {w_ij }

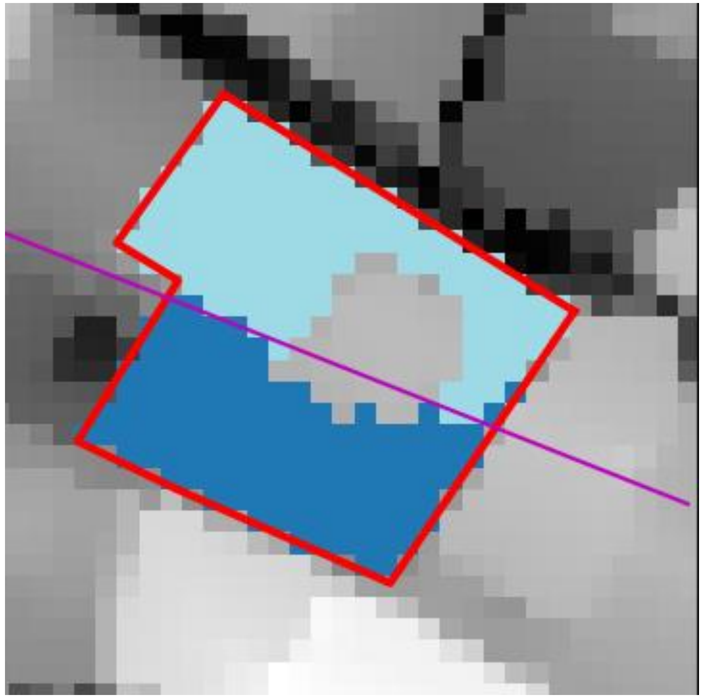


normal angular difference — *grad magnitude difference* — *grad direction difference* — *L1 distance*

$$w_{ij} = \begin{cases} \infty, & if\ L_{ij} > 6 \\ w_N N_{ij} + w_M M_{ij} + w_D D_{ij} + w_L L_{ij}, & otherwise \end{cases}$$

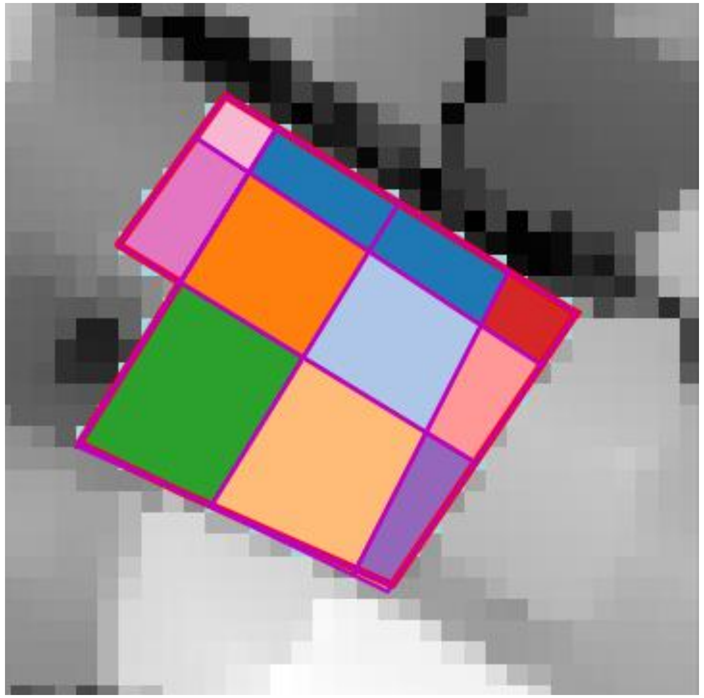
DSM Modelling

- From the DSM is possible to obtain the 3D building shape
   6. Hip-line regression



DSM Modelling

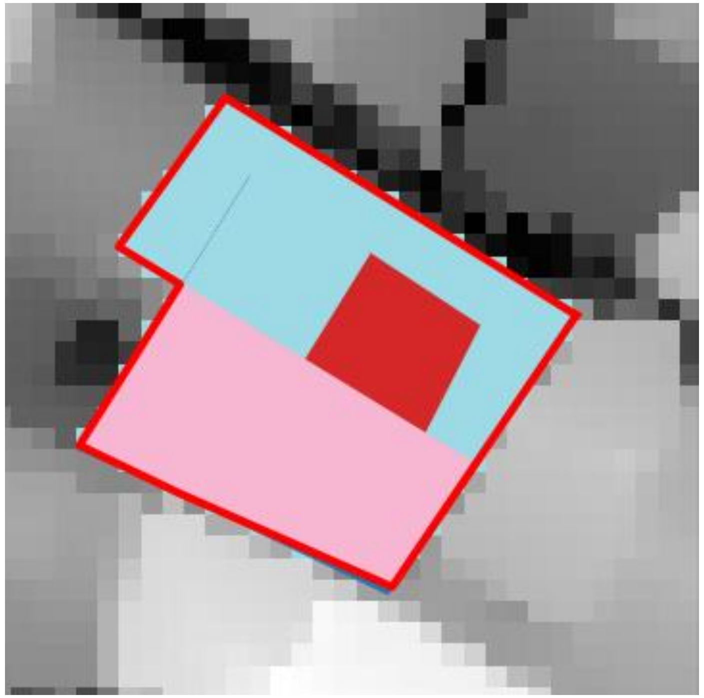
- From the DSM is possible to obtain the 3D building shape
  6. Hip-line regression
  7. Planar patch definition



DSM Modelling

- From the DSM is possible to obtain the 3D building shape

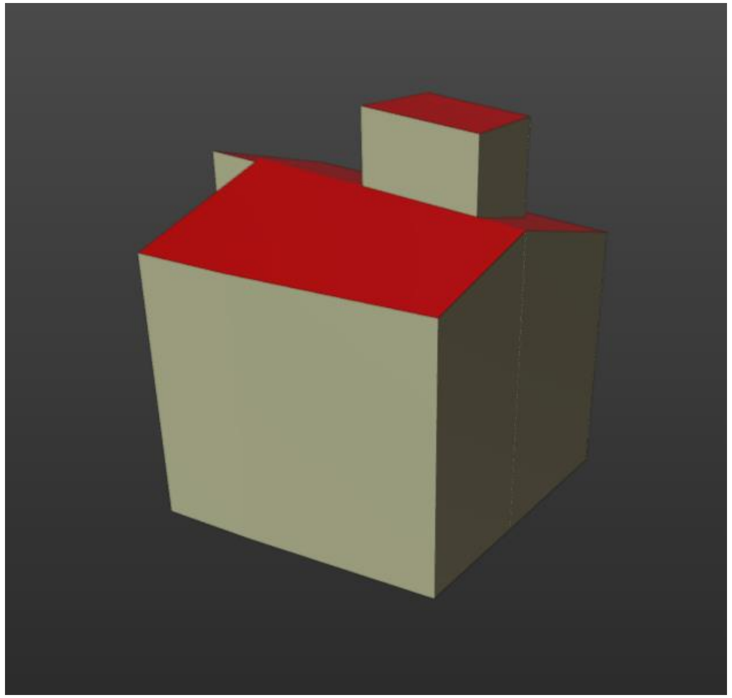
6. Hip-line regression
7. Planar patch definition
8. Planar patch labeling and grouping



DSM Modelling

- From the DSM is possible to obtain the 3D building shape

6. Hip-line regression
7. Planar patch definition
8. Planar patch labeling and grouping
9. Compute the 3D roof planes by robust regression for each planar patch



DSM Modelling

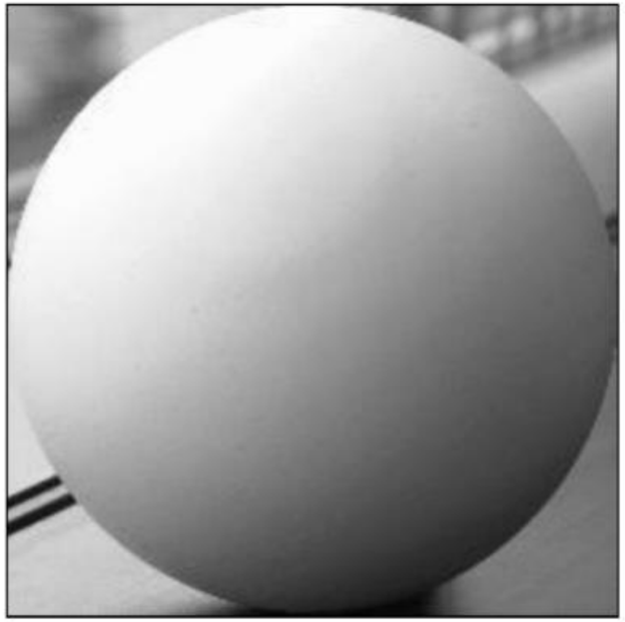
- Repeating the process for all the city buildings, a complete 3D map can be obtained

3D Reconstruction

Shape from Shading

Shape from Shading

- Is it possible to estimate the 3D shape of an object exploiting the shading?
- That is: from the image intensity recover the 3D shape



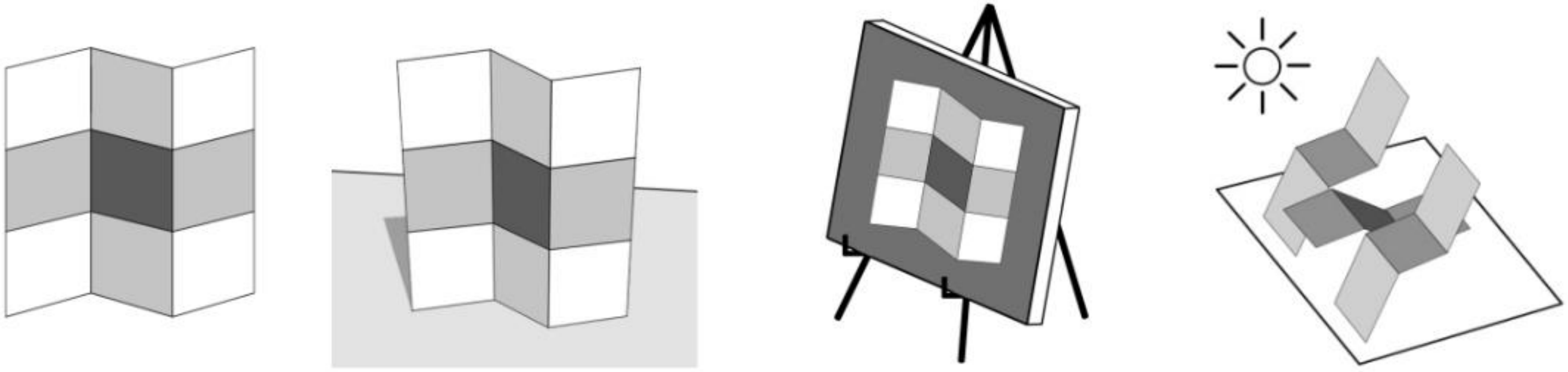
Slide from A. Geiger

Shape from Shading

- Observing the shading could give rise to different interpretations

Shape from Shading

- Observing the shading could give rise to different interpretations

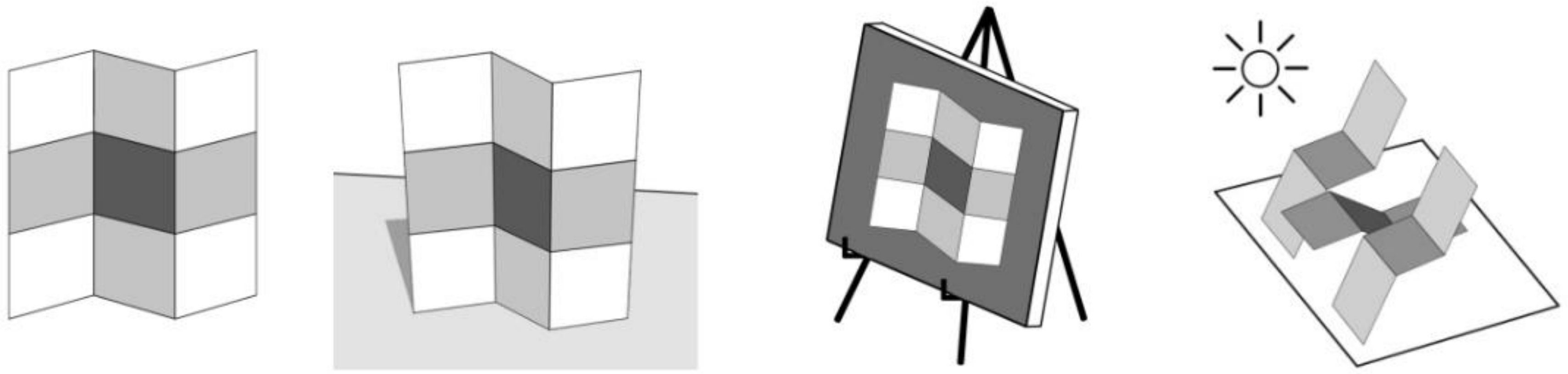


(a) an image (b) a likely explanation (c) painter's explanation (d) sculptor's explanation

- For example, the 2D image (a) could be obtained by
  - (b) a 3D structure with planes in different orientations producing such shading
  - (c) but it could also be a picture of a painting
  - (d) or a particular 3D structure with specific illumination

Shape from Shading

- Observing the shading could give rise to different interpretations



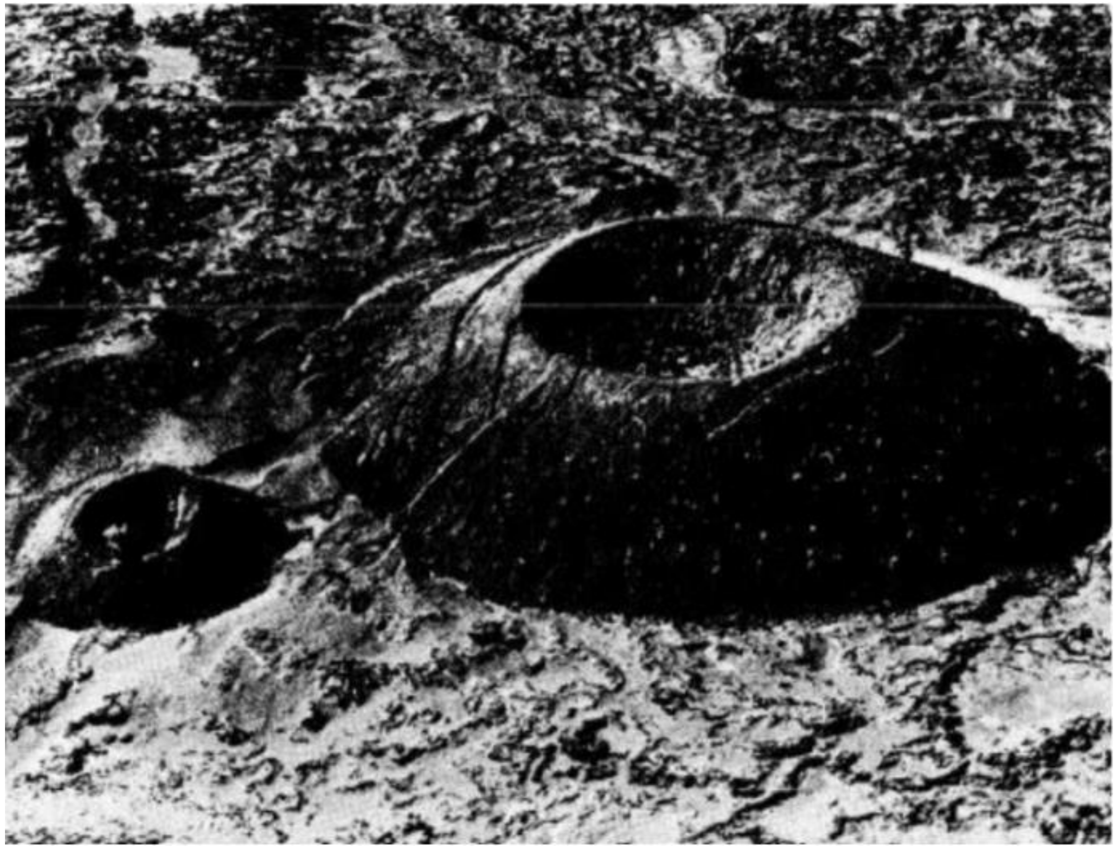
(a) an image (b) a likely explanation (c) painter's explanation (d) sculptor's explanation

- We must use some **prior knowledge** to solve this problem!

Slide from A. Geiger

Shape from Shading

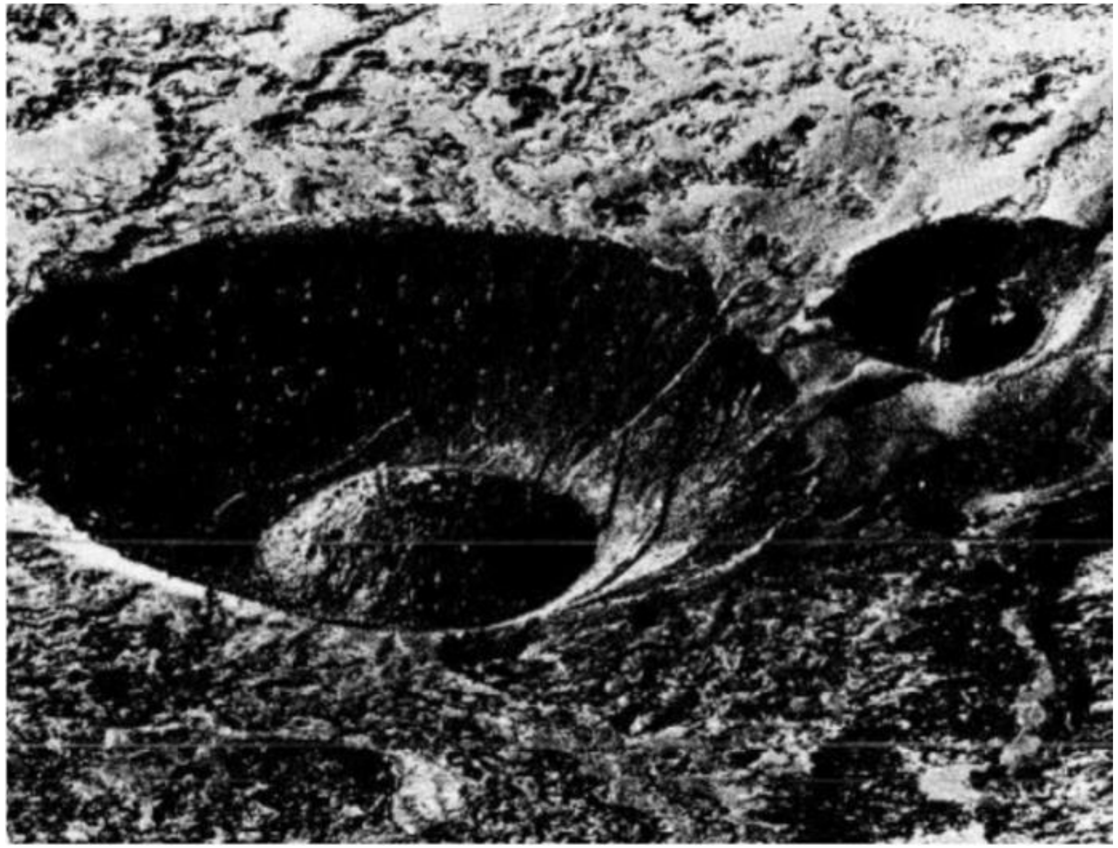
- Human perception



Slide from A. Geiger

Shape from Shading

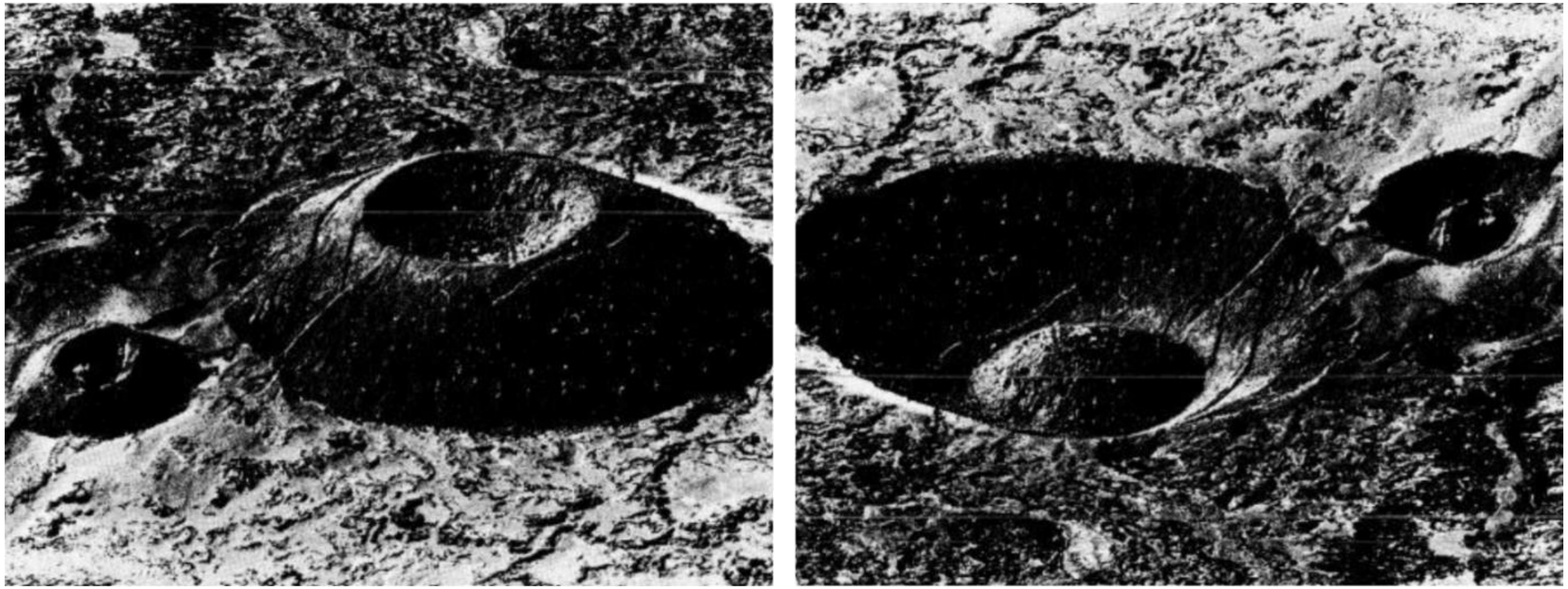
- Human perception



Slide from A. Geiger

Shape from Shading

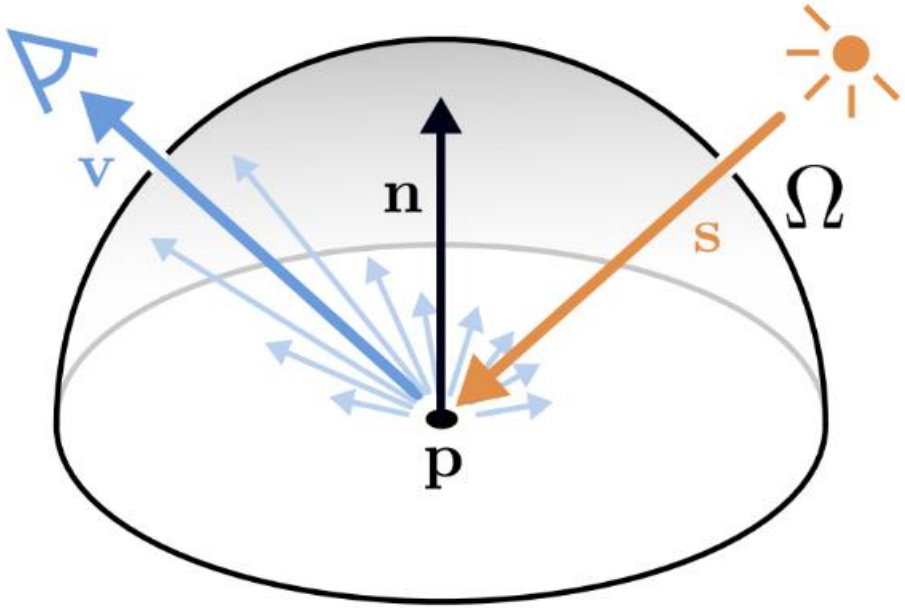
- Human perception



- We also **make assumption based on our experience**, for instance we assume that the light always came from the upward direction

Slide from A. Geiger

Rendering equations

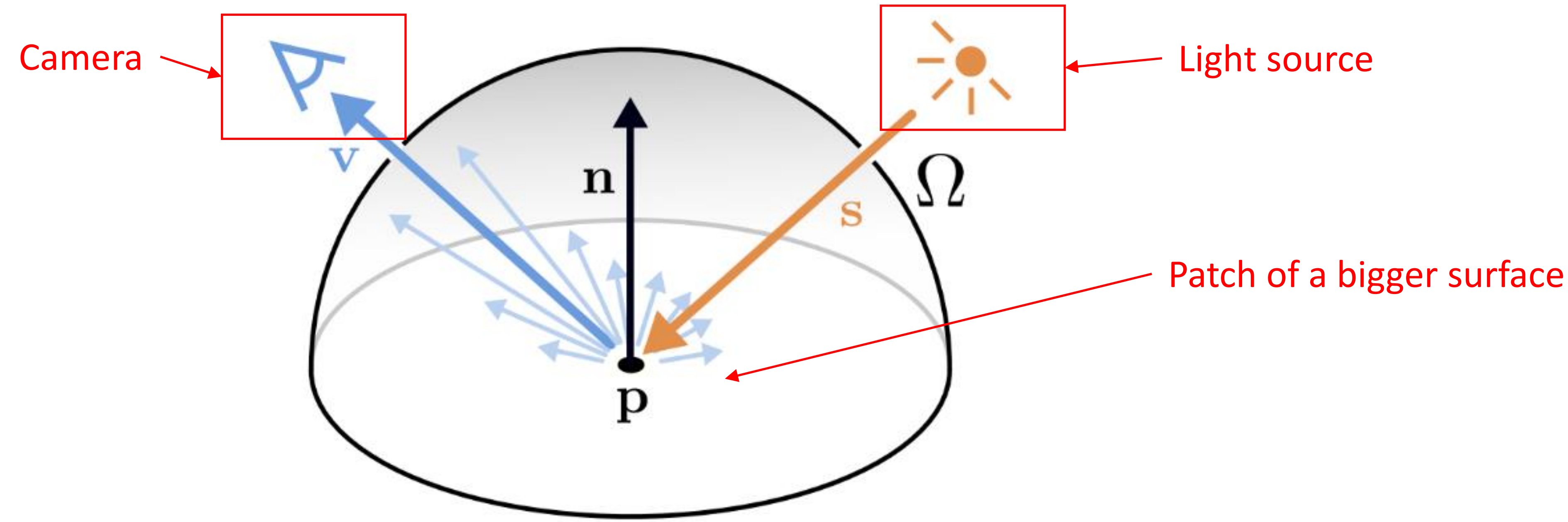


- Let $\mathbf{p} \in \mathbb{R}^3$ denote a 3D surface point, $\mathbf{v} \in \mathbb{R}^3$ the viewing direction and $\mathbf{s} \in \mathbb{R}^3$ the incoming light direction. The rendering equation describes how much of the light L_{in} with wavelength λ arriving at $\mathbf{p}$ is reflected into the viewing direction $\mathbf{v}$:

$$L_{\text{out}}(\mathbf{p}, \mathbf{v}, \lambda) = L_{\text{emit}}(\mathbf{p}, \mathbf{v}, \lambda) + \int_{\Omega} \text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda) \cdot L_{\text{in}}(\mathbf{p}, \mathbf{s}, \lambda) \cdot (-\mathbf{n}^\top \mathbf{s}) \, d\mathbf{s}$$

Slide from A. Geiger

Rendering equations

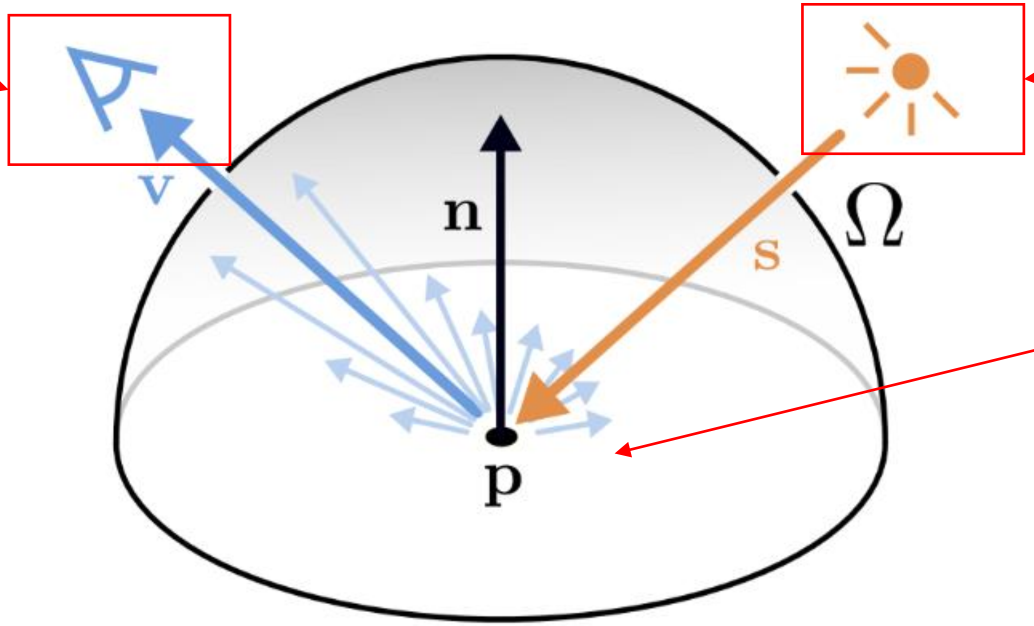


- Let $\mathbf{p} \in \mathbb{R}^3$ denote a 3D surface point, $\mathbf{v} \in \mathbb{R}^3$ the viewing direction and $\mathbf{s} \in \mathbb{R}^3$ the incoming light direction. The rendering equation describes how much of the light L_{in} with wavelength λ arriving at $\mathbf{p}$ is reflected into the viewing direction $\mathbf{v}$:

$$L_{\text{out}}(\mathbf{p}, \mathbf{v}, \lambda) = L_{\text{emit}}(\mathbf{p}, \mathbf{v}, \lambda) + \int_{\Omega} \text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda) \cdot L_{\text{in}}(\mathbf{p}, \mathbf{s}, \lambda) \cdot (-\mathbf{n}^\top \mathbf{s}) \, d\mathbf{s}$$

Slide from A. Geiger

Rendering equations



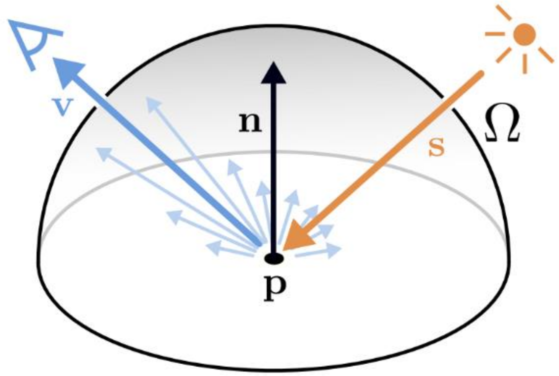
- Let $\mathbf{p} \in \mathbb{R}^3$ denote a 3D surface point, $\mathbf{v} \in \mathbb{R}^3$ the viewing direction and $\mathbf{s} \in \mathbb{R}^3$ the incoming light direction. The rendering equation describes how much of the light L_{in} with wavelength λ arriving at $\mathbf{p}$ is reflected into the viewing direction $\mathbf{v}$:

$$\underbrace{L_{\text{out}}(\mathbf{p}, \mathbf{v}, \lambda)}_{\text{Received light}} = \underbrace{L_{\text{emit}}(\mathbf{p}, \mathbf{v}, \lambda)}_{\text{Light emitted by the point}} + \int_{\Omega} \underbrace{\text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda)}_{\text{Bidirectional Reflectance Distribution Function}} \cdot \underbrace{L_{\text{in}}(\mathbf{p}, \mathbf{s}, \lambda)}_{\text{Strength of the incoming light}} \cdot \underbrace{(-\mathbf{n}^\top \mathbf{s})}_{\text{Incidence of the light}} \, d\mathbf{s}$$

Slide from A. Geiger

Rendering equations

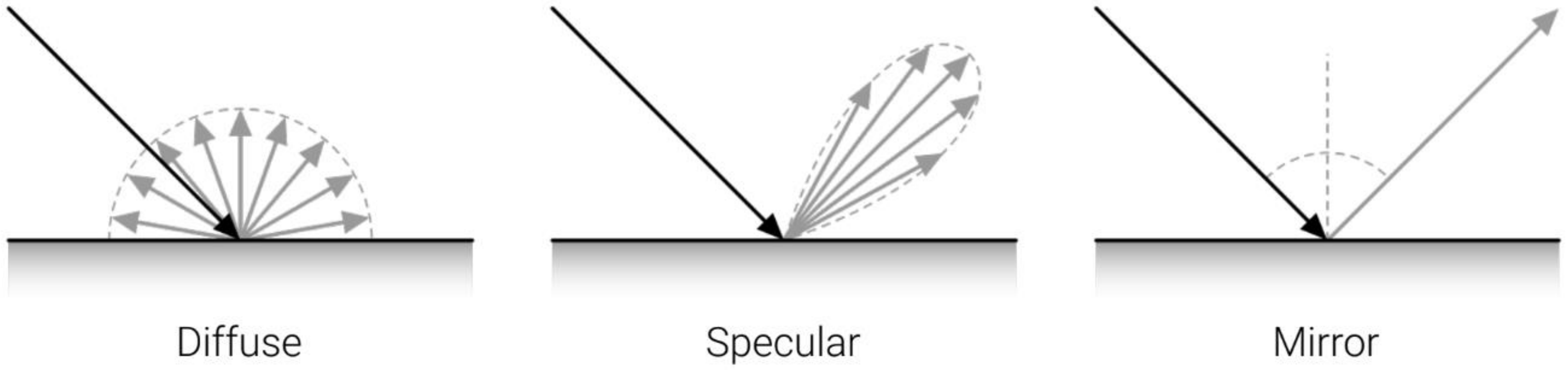
$$\underbrace{L_{\text{out}}(\mathbf{p},\mathbf{v},\lambda)}_{\text{Received light}} = \underbrace{L_{\text{emit}}(\mathbf{p},\mathbf{v},\lambda)}_{\text{Light emitted by the point}} + \int_{\Omega} \underbrace{\text{BRDF}(\mathbf{p},\mathbf{s},\mathbf{v},\lambda)}_{\text{Bidirectional Reflectance Distribution Function}} \cdot \underbrace{L_{\text{in}}(\mathbf{p},\mathbf{s},\lambda)}_{\text{Strength of the incoming light}} \cdot \underbrace{(-\mathbf{n}^{\top}\mathbf{s})}_{\text{Incidence of the light}} \, d\mathbf{s}$$



- The BRDF gives the reflectance of a target as a function of illumination geometry and viewing geometry. It defines how light is reflected at an opaque surface.
- $L_{emit} > 0$ only for light emitting surfaces (typically can be neglected)
- $(-\mathbf{n}^{\top}\mathbf{s})$ is related to the angle of incidence of the light w.r.t. the surface. If the surface normal $\mathbf{n}$ and the light direction $\mathbf{s}$ are parallel, the light intensity is maximized
- We evaluate the integral on the hemisphere Ω of all possible light directions because we can have multiple light sources

Slide from A. Geiger

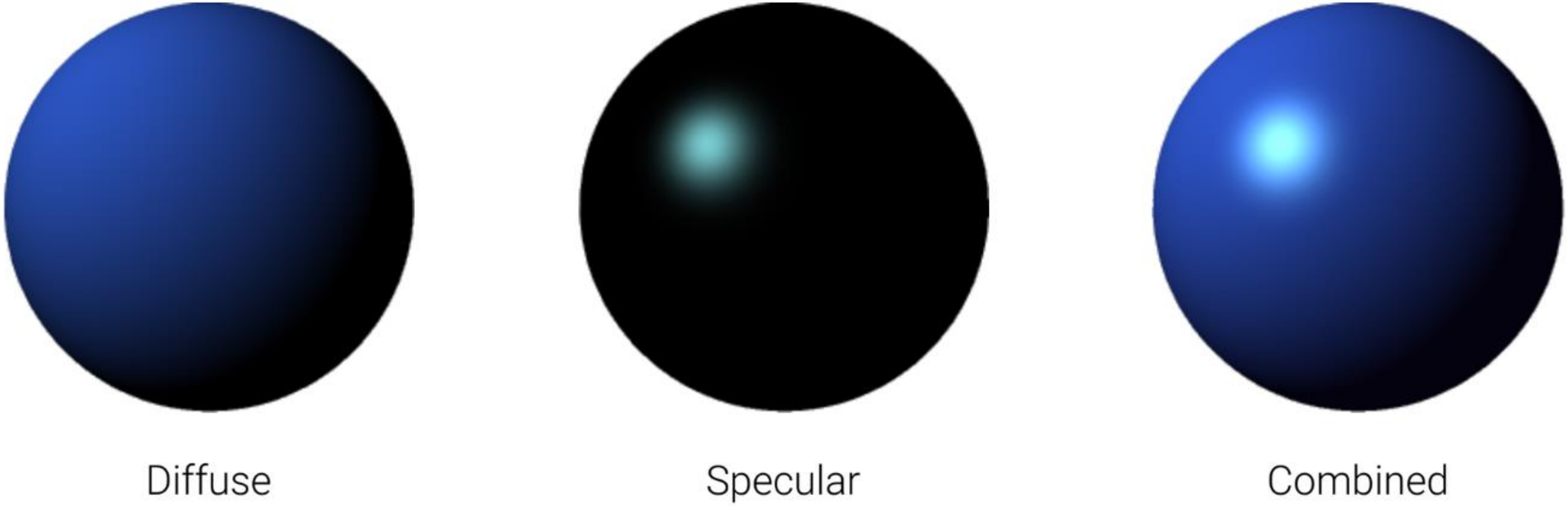
Diffuse and specular reflections



- Typical BRDFs have a diffuse and a specular component
- The **diffuse** (=constant) component **scatters light uniformly in all directions**
  - This leads to **shading**, i.e., smooth variation of intensity w.r.t. surface normal
- The **specular** component depends strongly on the outgoing light direction

Slide from A. Geiger

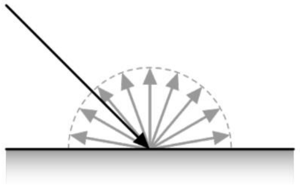
Diffuse and specular reflections



- Usually, materials are a **combinations** of diffuse and specular reflections

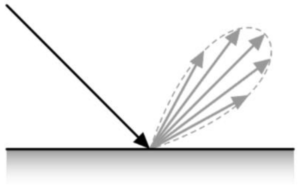
Slide from A. Geiger

Diffuse and specular reflections



Clay Pot

Cloud Gate (Kapoor, 2006)



- Usually, materials are a **combinations** of diffuse and specular reflections
- However, exist also (almost) purely diffuse and (almost) purely specular surfaces
- A purely diffuse surface is also known as Lambertian surface

Slide from A. Geiger

Rendering equations

- Let makes some simplification on the general rendering equation

$$L_{\text{out}}(\mathbf{p}, \mathbf{v}, \lambda) = L_{\text{emit}}(\mathbf{p}, \mathbf{v}, \lambda) + \int_{\Omega} \text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda) \cdot L_{\text{in}}(\mathbf{p}, \mathbf{s}, \lambda) \cdot (-\mathbf{n}^\top \mathbf{s}) \, d\mathbf{s}$$

Rendering equations

- Let makes some simplification on the general rendering equation

$$L_{\text{out}}(\mathbf{p}, \mathbf{v}, \lambda) = L_{\text{emit}}(\mathbf{p}, \mathbf{v}, \lambda) + \int_{\Omega} \text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda) \cdot L_{\text{in}}(\mathbf{p}, \mathbf{s}, \lambda) \cdot (-\mathbf{n}^{\top}\mathbf{s}) \, d\mathbf{s}$$

  - We can **drop the wavelength $\boldsymbol{\lambda}$** because the camera sensor already performs integrations on some wavelength

Slide from A. Geiger

Rendering equations

- Let makes some simplification on the general rendering equation

$$L_{\text{out}}(\mathbf{p}, \mathbf{v}, \lambda) = L_{\text{emit}}(\mathbf{p}, \mathbf{v}, \lambda) + \int_{\Omega} \text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda) \cdot L_{\text{in}}(\mathbf{p}, \mathbf{s}, \lambda) \cdot (-\mathbf{n}^{\top}\mathbf{s})\, d\mathbf{s}$$

  - We can **drop the wavelength** λ because the camera sensor already performs integrations on some wavelength
  - We can **drop the point p** for simplicity, and put $L_{\text{emit}} = 0$

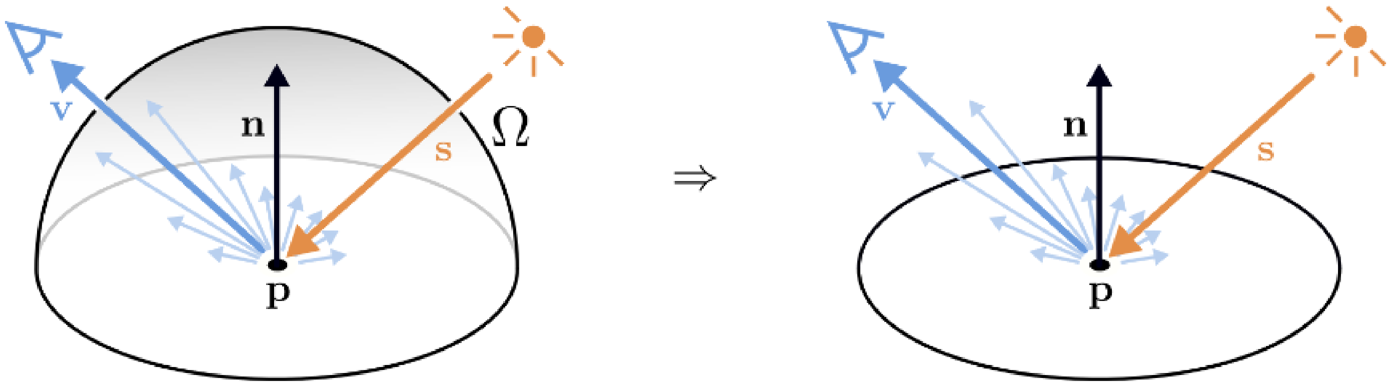
Slide from A. Geiger

Rendering equations

- Let makes some simplification on the general rendering equation

$$L_{\text{out}}(\mathbf{p}, \mathbf{v}, \lambda) = L_{\text{emit}}(\mathbf{p}, \mathbf{v}, \lambda) + \int_{\Omega} \text{BRDF}(\mathbf{p}, \mathbf{s}, \mathbf{v}, \lambda) \cdot L_{\text{in}}(\mathbf{p}, \mathbf{s}, \lambda) \cdot (-\mathbf{n}^{\top}\mathbf{s})\, d\mathbf{s}$$

  - We can **drop the wavelength** λ because the camera sensor already performs integrations on some wavelength
  - We can **drop the point p** for simplicity, and put $L_{\text{emit}} = 0$
  - More importantly, we can assume to have a **single light source**, and so **avoiding to compute the integral over Ω**

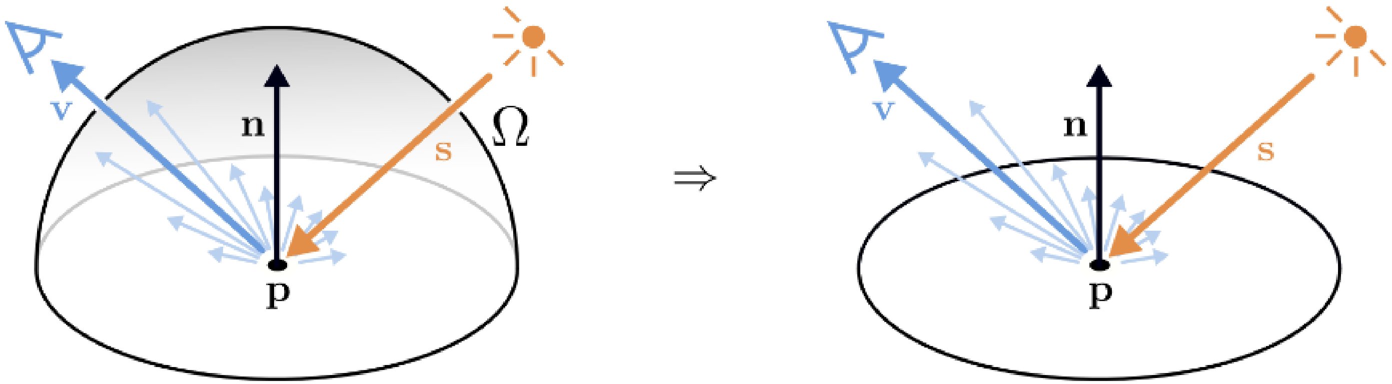


Slide from A. Geiger

Rendering equations

- Let makes some simplification on the general rendering equation

$$L_{\text{out}}(\mathbf{p},\mathbf{v},\lambda) = L_{\text{emit}}(\mathbf{p},\mathbf{v},\lambda) + \int_{\Omega} \text{BRDF}(\mathbf{p},\mathbf{s},\mathbf{v},\lambda) \cdot L_{\text{in}}(\mathbf{p},\mathbf{s},\lambda) \cdot (-\mathbf{n}^{\top}\mathbf{s})\, d\mathbf{s}$$



- In the end we obtain

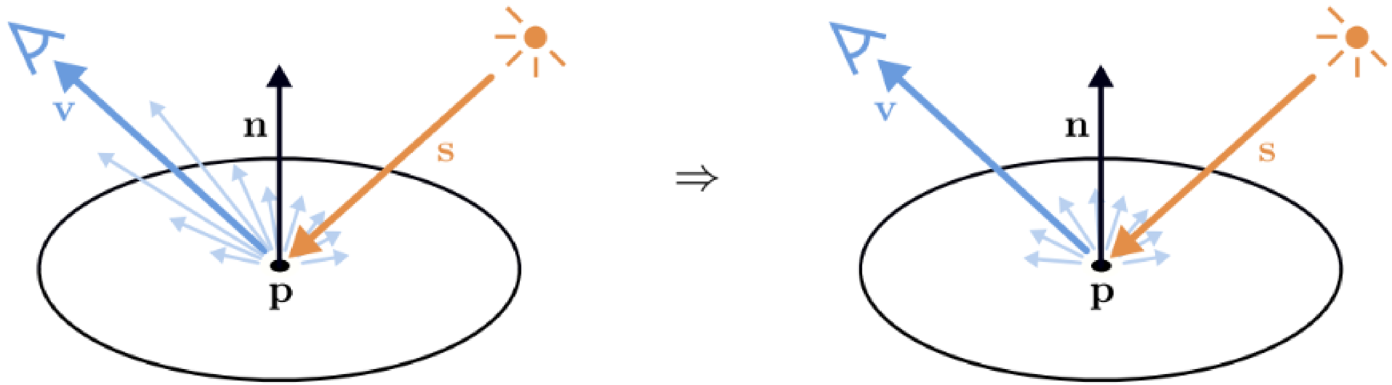
$$L_{\text{out}}(\mathbf{v}) = \text{BDRF}(\mathbf{s},\mathbf{v}) L_{\text{in}}(\mathbf{s})(-\mathbf{n}^{\top}\mathbf{s})$$

Slide from A. Geiger

Rendering equations

- We can also assume to look at purely diffuse material. In this case the BRDF function become a constant ρ (i.e., albedo) that does not depend anymore on $\mathbf{s}$ and $\mathbf{v}$

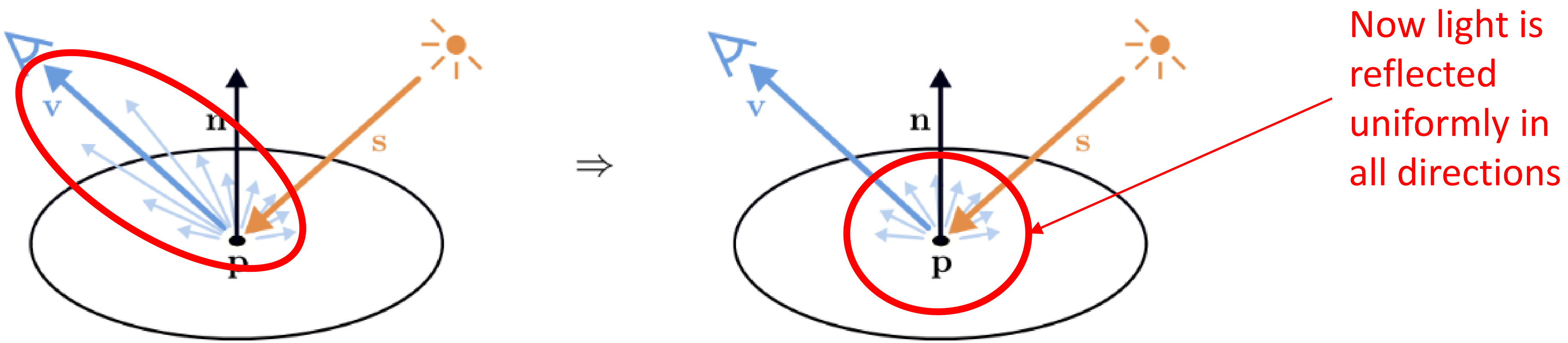
$$L_{\text{out}} = \rho L_{\text{in}}(-\mathbf{n}^\top \mathbf{s})$$



Rendering equations

- We can also assume to look at purely diffuse material. In this case the BRDF function become a constant ρ (i.e., albedo) that does not depend anymore on $\mathbf{s}$ and $\mathbf{v}$

$$L_{\text{out}} = \rho L_{\text{in}}(-\mathbf{n}^{\top}\mathbf{s})$$

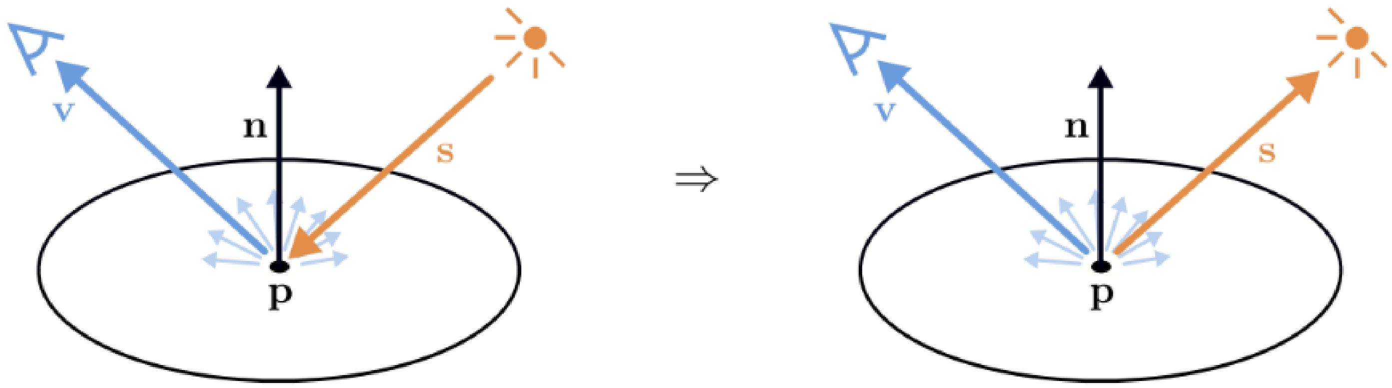


Slide from A. Geiger

Rendering equations

- Finally, we can remove the minus sign from $(-\mathbf{n}^\top \mathbf{s})$ by simply considering the reverse $\mathbf{s}$ vector

$$L_{\text{out}} = \rho L_{\text{in}} \mathbf{n}^\top \mathbf{s}$$



Slide from A. Geiger

Rendering equations

- Finally, we can remove the minus sign from $(-\mathbf{n}^\top\mathbf{s})$ by simply considering the reverse $\mathbf{s}$ vector

$$L_{\text{out}} = \rho L_{\text{in}} \mathbf{n}^\top \mathbf{s}$$

- Having assumed a fixed light source (e.g., we calibrated its position), the rendering equation depends only on the normal orientation of the surface

$$L_{\text{out}} = \rho L_{\text{in}} \mathbf{n}^\top \mathbf{s} = R(\mathbf{n})$$

- The $R(\mathbf{n})$ function is known as **reflectance map**

Shape from Shading

- The $R(\mathbf{n})$ function is known as **reflectance map**

$$L_{\text{out}} = \rho L_{\text{in}} \mathbf{n}^\top \mathbf{s} = R(\mathbf{n})$$

- So the idea is to **exploit the image intensities in order to obtain the surface normal vectors** for all the object points

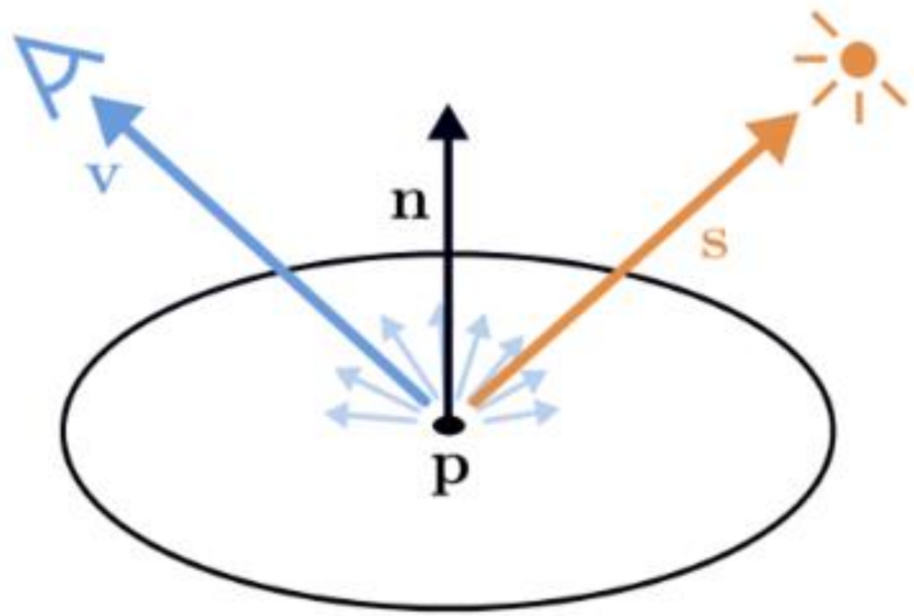
- From the normal vectors, the 3D shape can finally be retrieved

- This is known as **Shape-from-Shading** (Horn, 1970)

Slide from A. Geiger

Shape from Shading

- To solve Shape from Shading we assume:
  - Lambertian (diffuse) material with constant albedo
    - $\text{BDRF} = \rho$
    - $L_{\text{emit}} = 0$
  - Known point light source at infinity
    - Light direction $\mathbf{s}$ is constant for all the points
    - A single light source to avoid the integral over $\boldsymbol{\Omega}$
  - Known camera at infinity
    - Viewing direction $\mathbf{v}$ is constant for all the points



Slide from A. Geiger

Shape from Shading

- We have to find the $\mathbf{n}$ that satisfy the simplified reflectance function $R(\mathbf{n}) = \rho L_{\text{in}} \mathbf{n}^\top \mathbf{s}$

- ρ and L_{in} are constant number
  - ρ is constant if the object is composed by a single material only

- ρ and L_{in} can be assumed to be absorbed into $R(\mathbf{n})$, so

$$R(\mathbf{n}) = \mathbf{n}^\top \mathbf{s}$$

- Question: how to model $\mathbf{n}$?

Slide from A. Geiger

Shape from Shading

- Question: how to model $\mathbf{n}$?
- Being a 3D normal vector, $\mathbf{n}$ has 2 DoF
- Instead of $\mathbf{n}$, we can represent the same information by using the **negative gradients of the depth-map**

$$(p, q) = \left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}\right)$$

Shape from Shading

- Question: how to model $\mathbf{n}$?
- Being a 3D normal vector, $\mathbf{n}$ has 2 DoF
- Instead of $\mathbf{n}$, we can represent the same information by using the **negative gradients of the depth-map**

$$(p, q) = \left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}\right)$$

and then, $\mathbf{n}$ can be obtained as

$$\mathbf{n} = \frac{(p, q, 1)^\top}{\sqrt{p^2 + q^2 + 1}}$$

Slide from A. Geiger

Shape from Shading

- Question: how to model $\mathbf{n}$?
- Being a 3D normal vector, $\mathbf{n}$ has 2 DoF
- Instead of $\mathbf{n}$, we can represent the same information by using the **negative gradients of the depth-map**

$$(p, q) = \left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}\right)$$

and then, $\mathbf{n}$ can be obtained as

$$\mathbf{n} = \frac{(p, q, 1)^\top}{\sqrt{p^2 + q^2 + 1}}$$

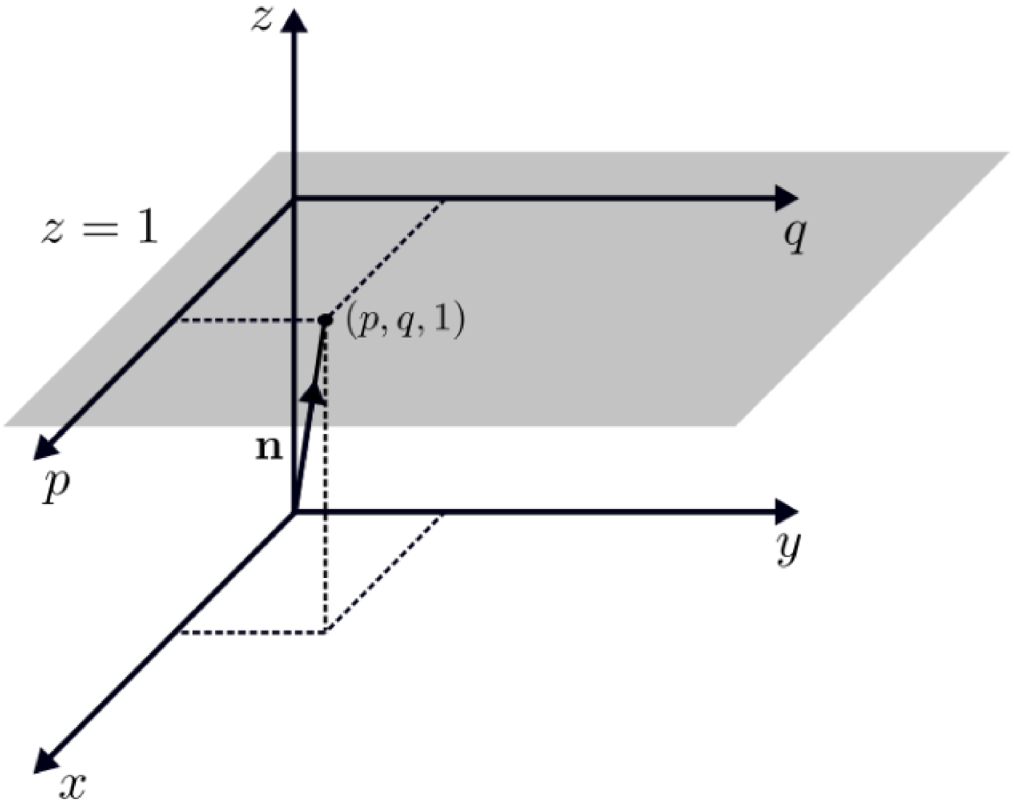
- Finally, we obtain

$$R(\mathbf{n}) = \frac{(ps_x, qs_y, s_z)}{\sqrt{p^2+q^2+1}} = R(p, q)$$

Slide from A. Geiger

Shape from Shading

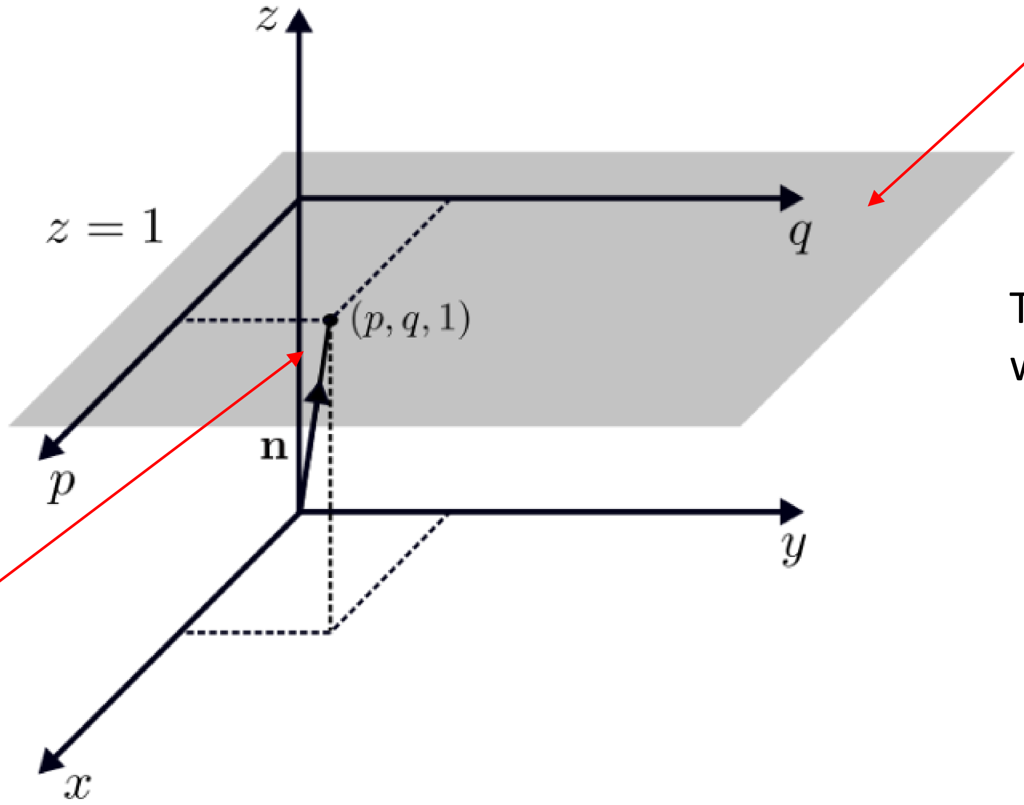
- Visualization of the gradient space



Slide from A. Geiger

Shape from Shading

- Visualization of the gradient space



Gradient space (plane Z = 1)

The normal vector intersect the gradient space in (p,q,1)

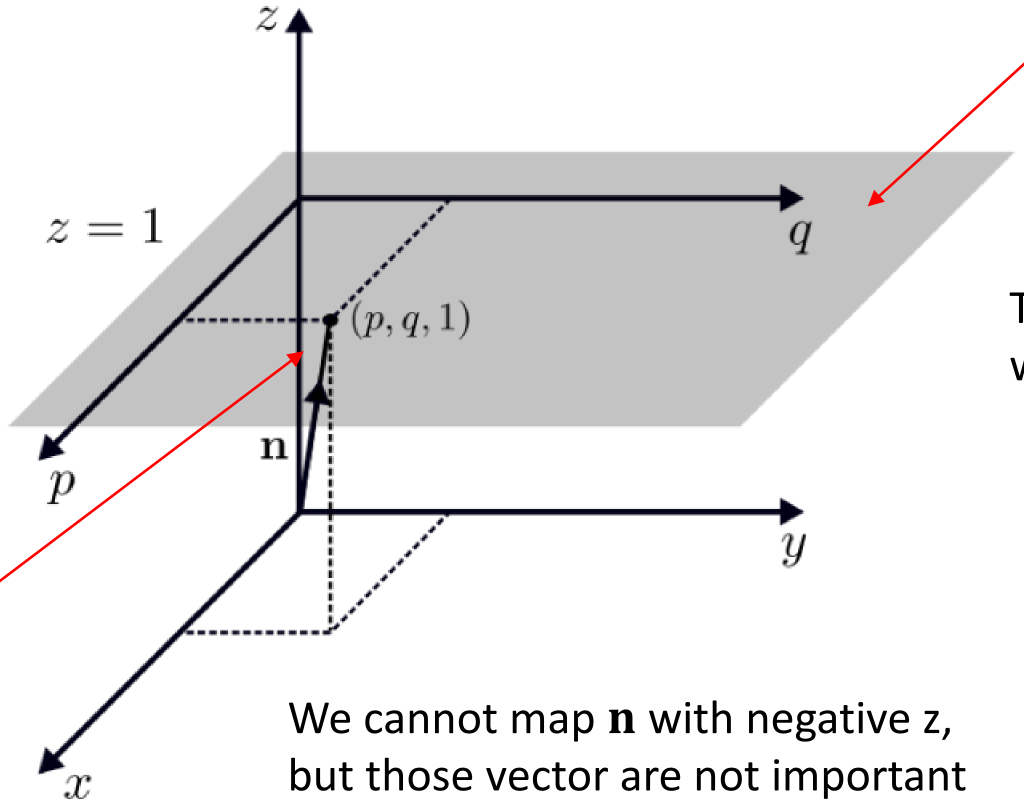
To obtain **n** from the (p,q,1) vector, we have to normalize it, i.e.

$$\mathbf{n} = \frac{(p, q, 1)^\top}{\sqrt{p^2 + q^2 + 1}}$$

Shape from Shading

- Visualization of the gradient space

Gradient space (plane Z = 1)



To obtain **n** from the (p,q,1) vector, we have to normalize it, i.e.

$$\mathbf{n} = \frac{(p, q, 1)^\top}{\sqrt{p^2 + q^2 + 1}}$$

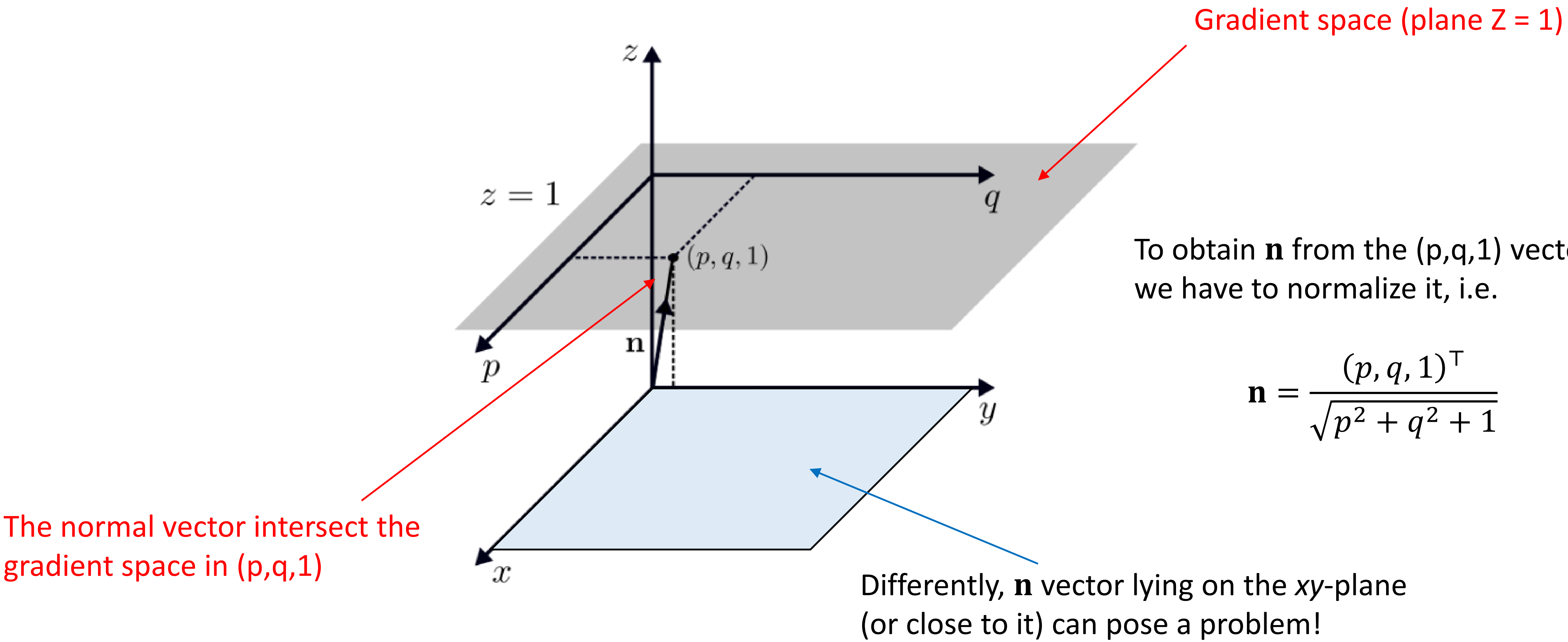
The normal vector intersect the gradient space in (p,q,1)

We cannot map **n** with negative z, but those vector are not important to this problem, since they do not reflect light

Slide from A. Geiger

Shape from Shading

- Visualization of the gradient space



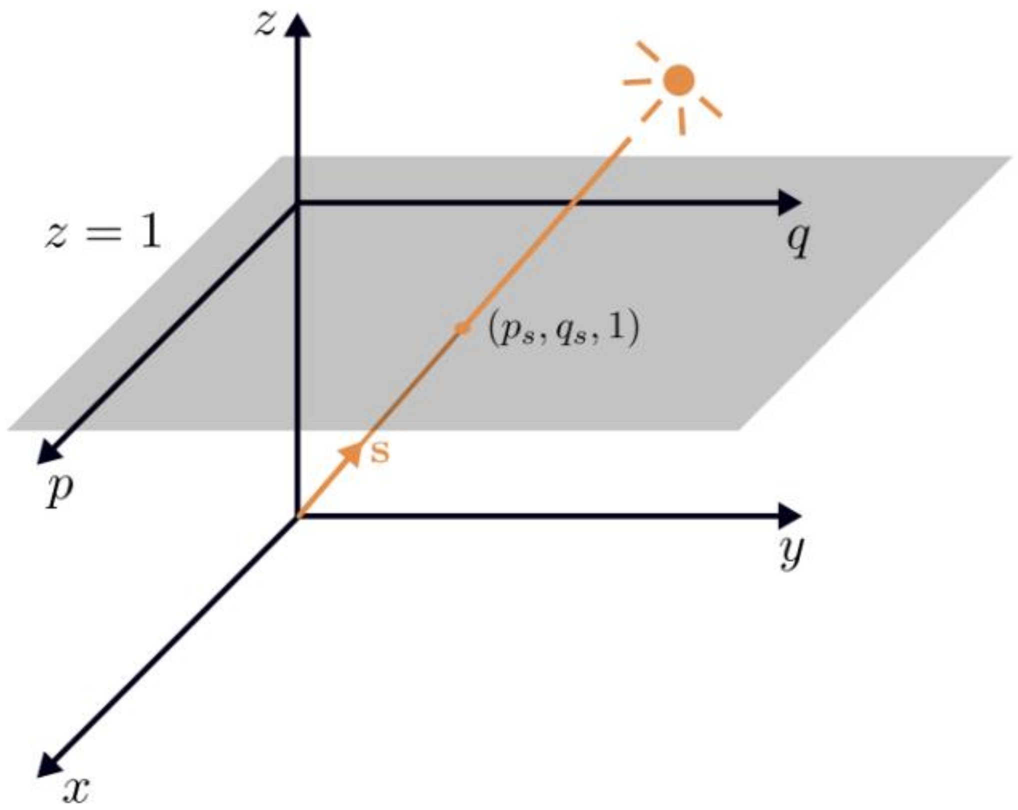
To obtain **n** from the (p,q,1) vector, we have to normalize it, i.e.

$$\mathbf{n} = \frac{(p, q, 1)^\mathsf{T}}{\sqrt{p^2 + q^2 + 1}}$$

Slide from A. Geiger

Shape from Shading

- Also the light source vector **s** can be represented in the gradient space

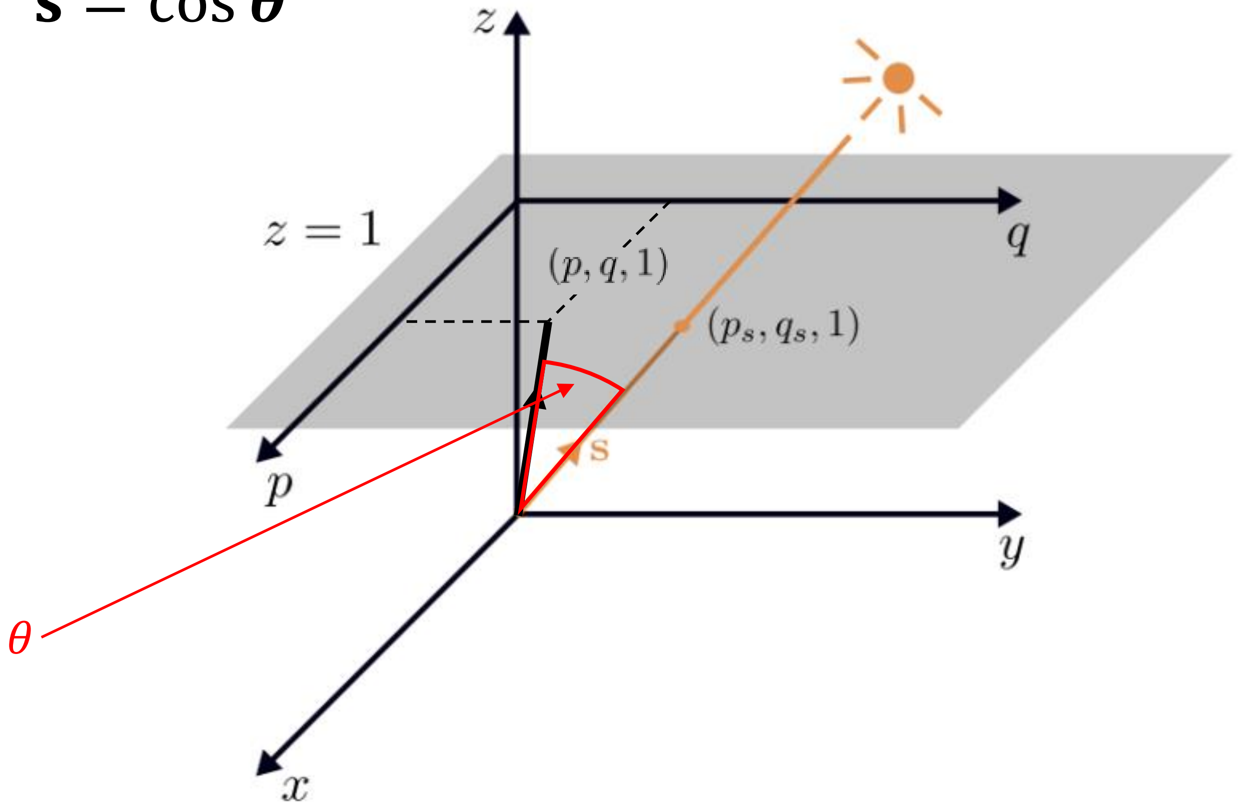


Slide from A. Geiger

Shape from Shading

- Since both $\mathbf{n}$ and $\mathbf{s}$ are unit vectors

$$R(\mathbf{n}) = \mathbf{n}^\top \mathbf{s} = \cos \boldsymbol{\theta}$$



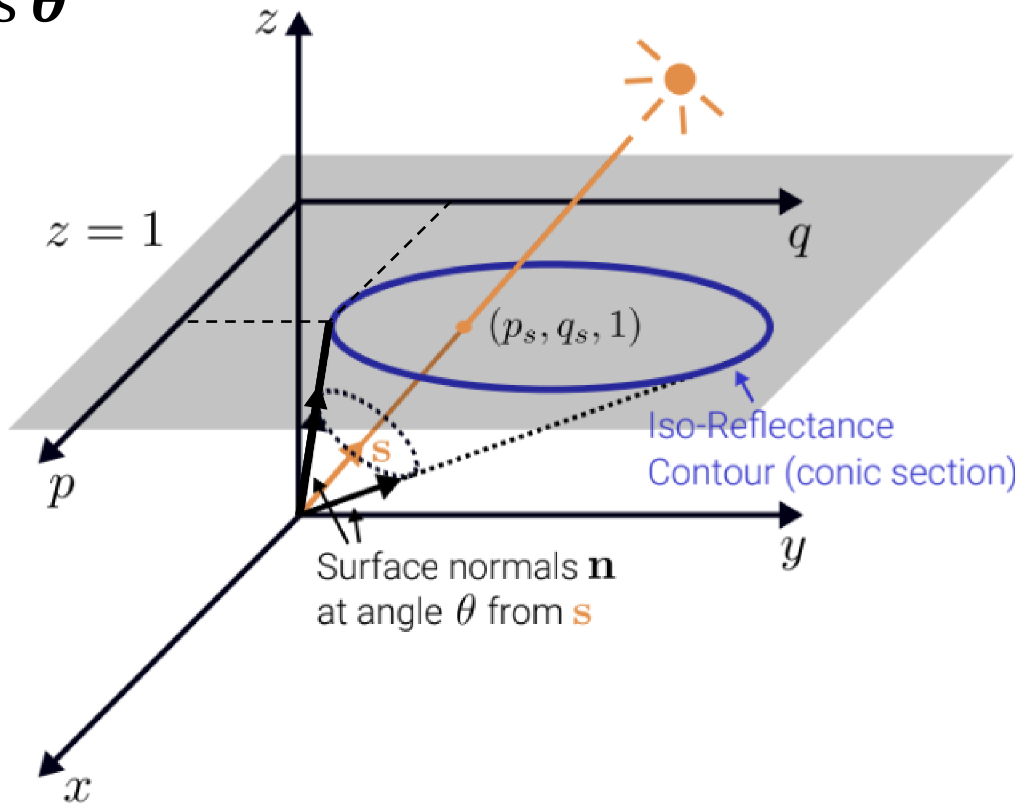
Shape from Shading

- Since both $\mathbf{n}$ and $\mathbf{s}$ are unit vectors

$$R(\mathbf{n}) = \mathbf{n}^\top \mathbf{s} = \cos \boldsymbol{\theta}$$

- All $\mathbf{n}$ that form an angle equal to $\boldsymbol{\theta}$ with $\mathbf{s}$ are valid solutions for $R(\mathbf{n})$

- The set of $\{\mathbf{n}\}$ solutions are a circle around $\mathbf{s}$



- Projecting the circle on the Z=1 plane yield a conic section

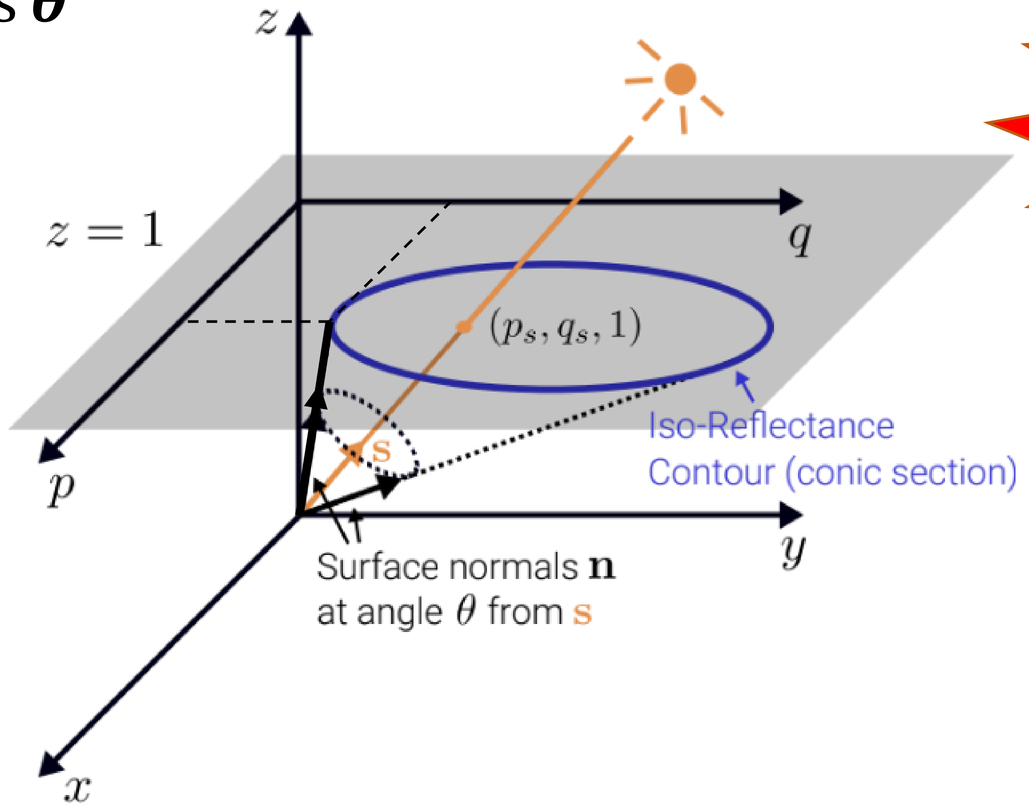
Shape from Shading

- Since both $\mathbf{n}$ and $\mathbf{s}$ are unit vectors

$$R(\mathbf{n}) = \mathbf{n}^\top \mathbf{s} = \cos \boldsymbol{\theta}$$

- All $\mathbf{n}$ that form an angle equal to $\boldsymbol{\theta}$ with $\mathbf{s}$ are valid solutions for $R(\mathbf{n})$

- The set of $\{\mathbf{n}\}$ solutions are a circle around $\mathbf{s}$



Solution is not unique!

- Projecting the circle on the Z=1 plane yield a conic section

Slide from A. Geiger

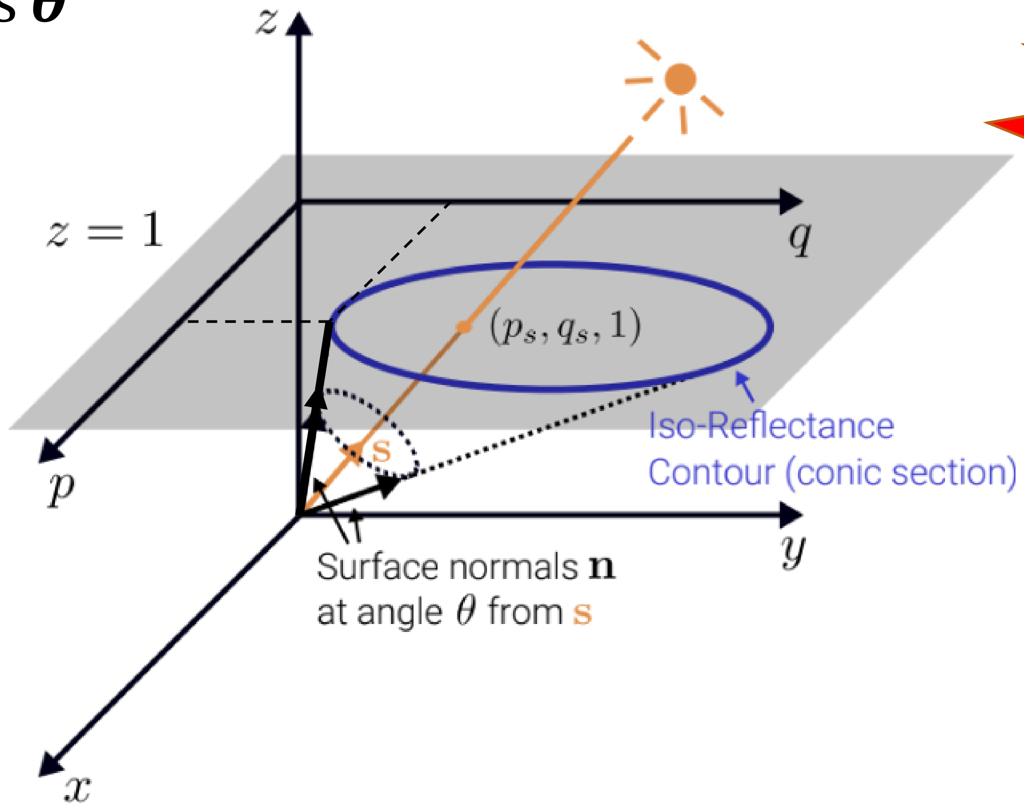
Shape from Shading

- Since both $\mathbf{n}$ and $\mathbf{s}$ are unit vectors

$$R(\mathbf{n}) = \mathbf{n}^\top \mathbf{s} = \cos \boldsymbol{\theta}$$

- All $\mathbf{n}$ that form an angle equal to $\boldsymbol{\theta}$ with $\mathbf{s}$ are valid solutions for $R(\mathbf{n})$

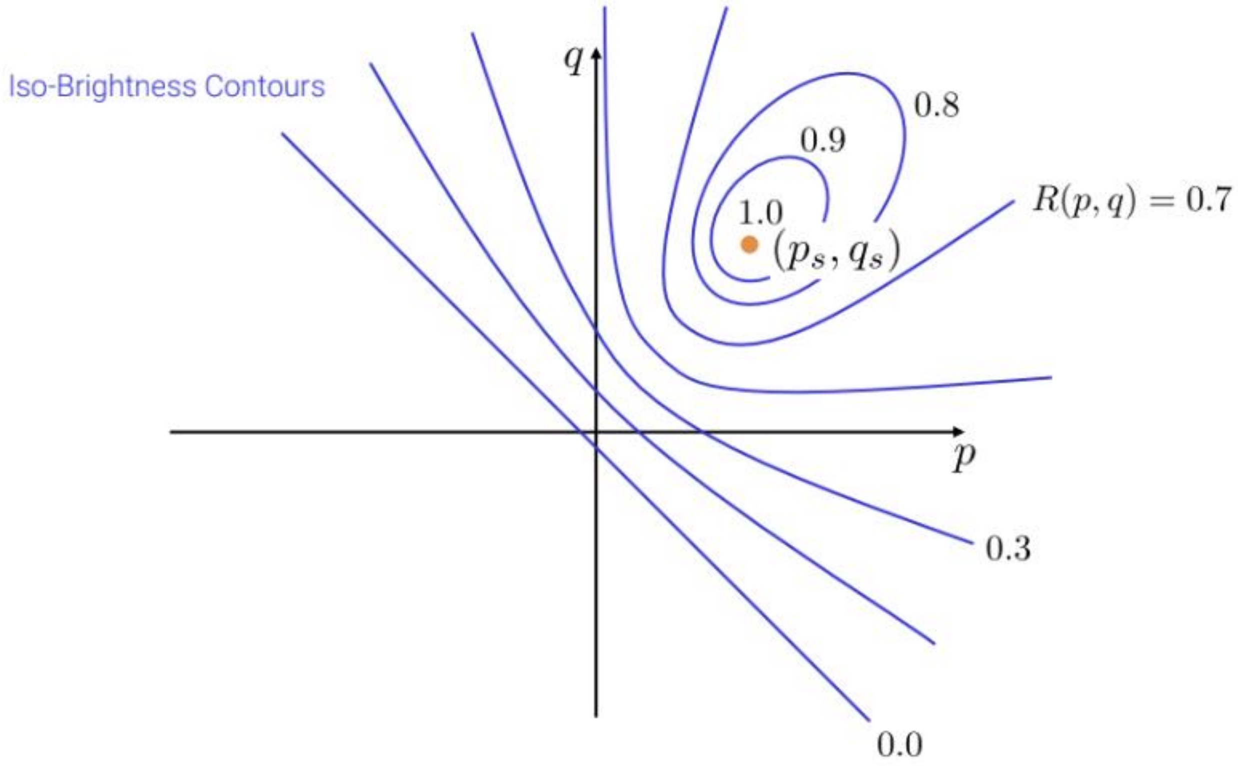
- The set of $\{\mathbf{n}\}$ solutions are a circle around $\mathbf{s}$



Indeed, we want to estimate 2 DoF with a single equation: the problem is **ill posed**!

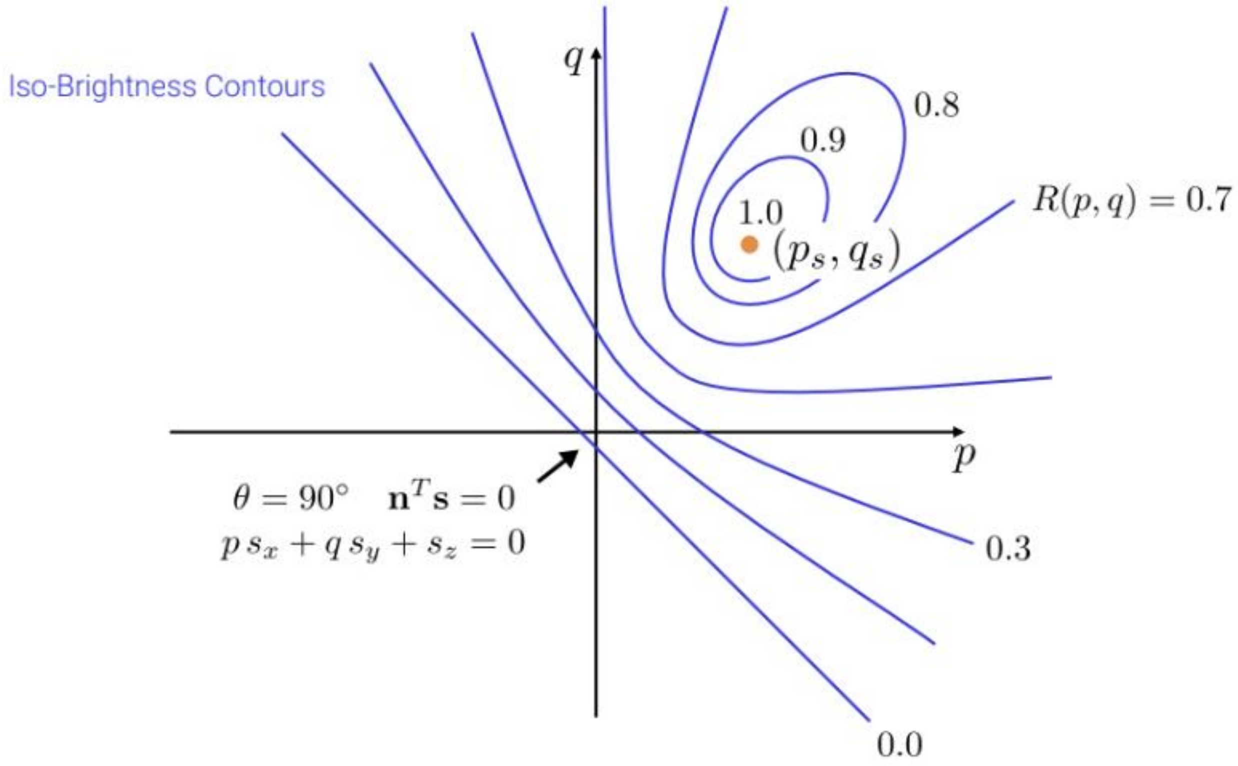
- Projecting the circle on the Z=1 plane yield a conic section

Shape from Shading



- There is only one exception: when the **light source direction is coincident with the normal direction**
- In this case the conic section collapses to a **point**

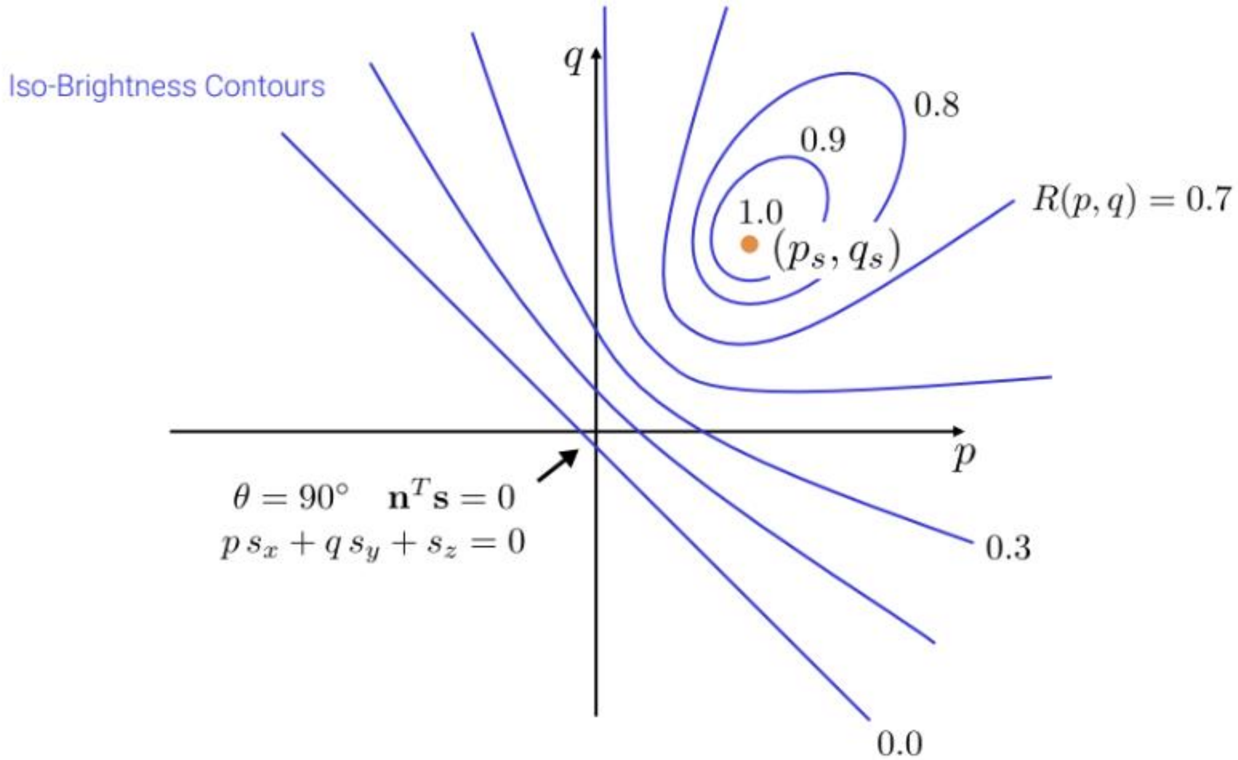
Shape from Shading



- Another particular case is when the light source direction and the normal direction are **orthogonal**
- In this case the conic section become a **line**

Slide from A. Geiger

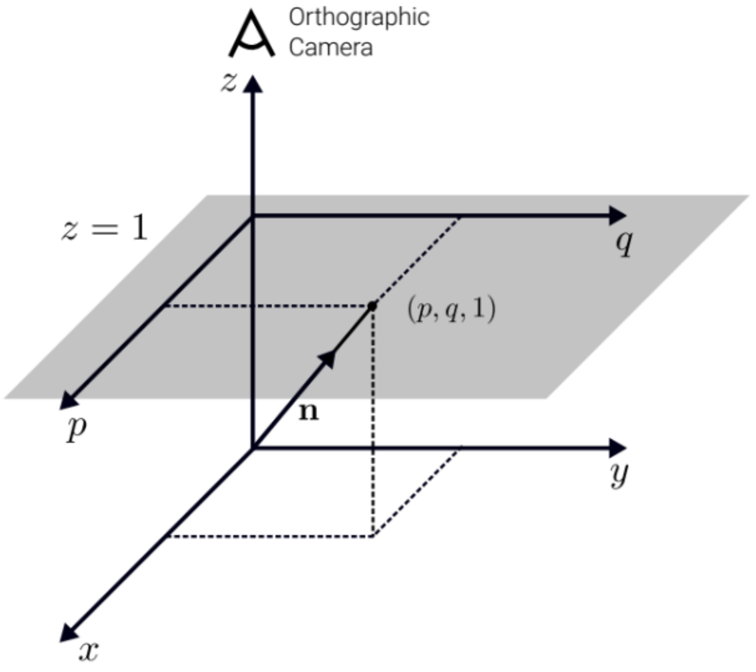
Shape from Shading

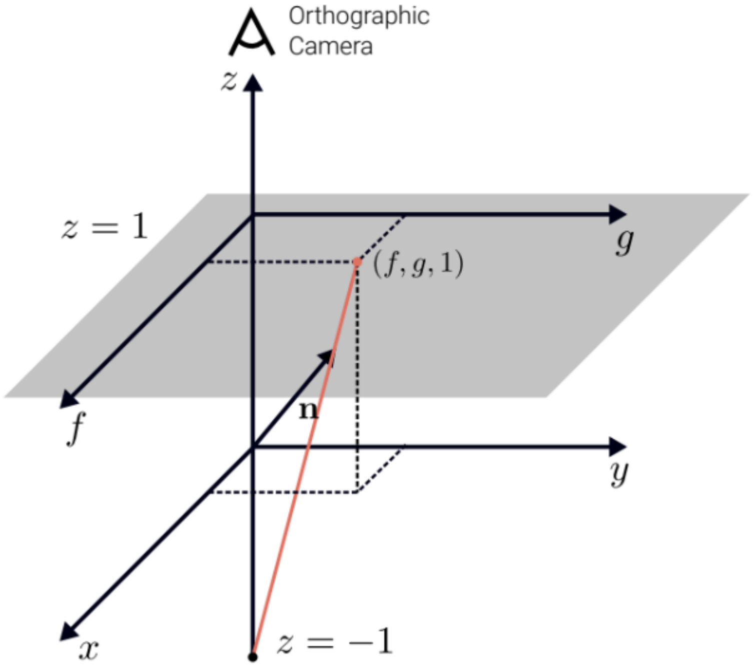


- Shape from Shading tries to solve this problem imposing additional constraints
- A sort of regularization is used in order to find an unique solution

Shape from Shading

- Before we noticed that if **n** lies in the *xy*-plane we cannot map it to the gradient space
- To solve this problem, we can change the representation to the **stereographic mapping**





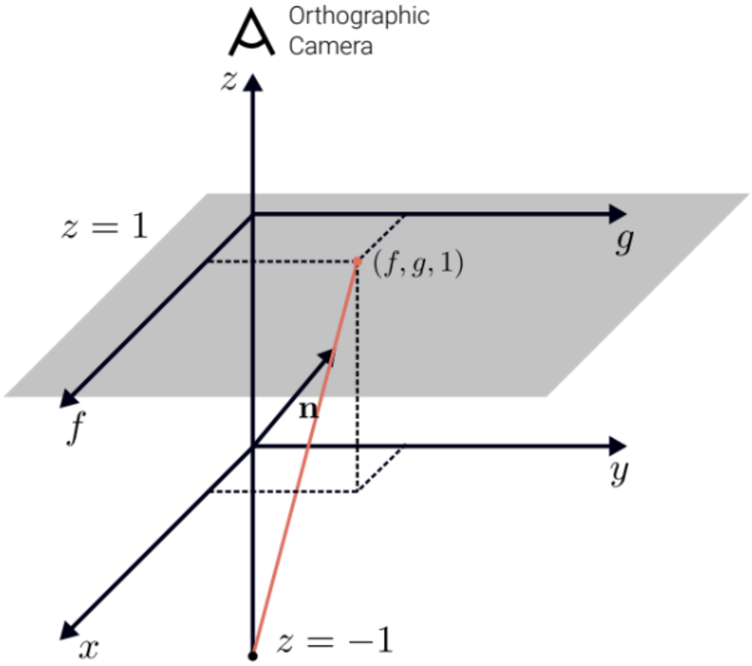
Slide from A. Geiger

Shape from Shading

- Before we noticed that if **n** lies in the *xy*-plane we cannot map it to the gradient space
- To solve this problem, we can change the representation to the **stereographic mapping**

- And we can still move back to the (p,q)

$$f = \frac{2p}{1 + \sqrt{p^2 + q^2 + 1}}$$
$$g = \frac{2q}{1 + \sqrt{p^2 + q^2 + 1}}$$



Slide from A. Geiger

Shape from Shading

- Shape from Shading tries to minimize

$$E_{image}(f,g) = \int\int (I(x,y) - R(f,g))^2 \, dx \, dy$$

Slide from A. Geiger

Shape from Shading

- Shape from Shading tries to minimize

$$E_{image}(f,g) = \int\int (I(x,y) - R(f,g))^2 \, dx \, dy$$

and to obtain unique solutions, impose this additional constraints

- **Smoothness**, to penalize rapid changes in surface gradients

$$E_{smooth}(f,g) = \int\int f_x^2 + f_y^2 + g_x^2 + g_y^2 \, dx \, dy$$

Slide from A. Geiger

Shape from Shading

- Shape from Shading tries to minimize

$$E_{image}(f,g) = \int\int (I(x,y) - R(f,g))^2 \, dx \, dy$$

and to obtain unique solutions, impose this additional constraints

- **Smoothness**, to penalize rapid changes in surface gradients

$$E_{smooth}(f,g) = \int\int f_x^2 + f_y^2 + g_x^2 + g_y^2 \, dx \, dy$$

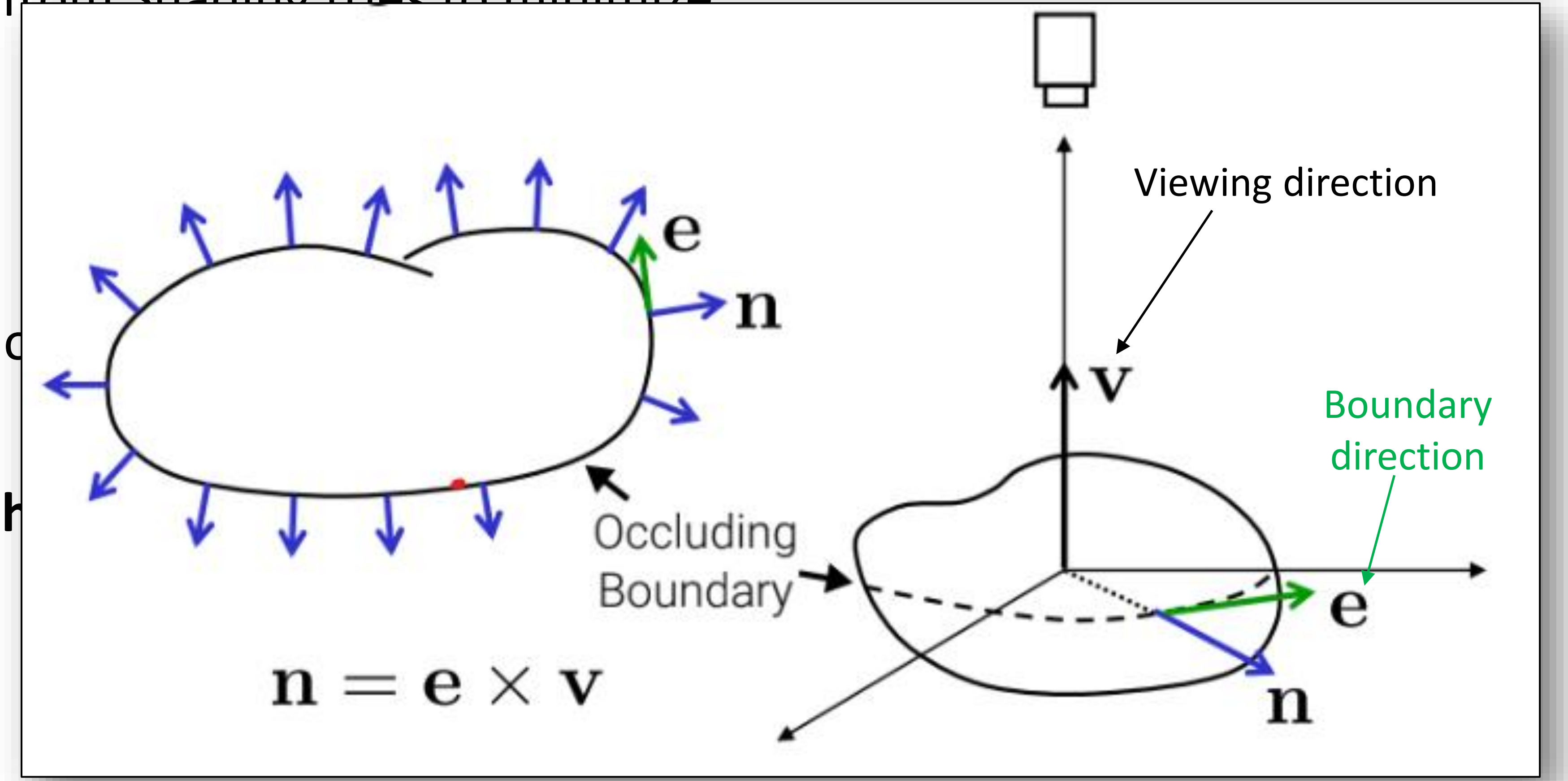
- **Occluding boundaries**, to constraint normal at occluding boundaries since are known

Slide from A. Geiger

Shape from Shading

- Shape from Shading tries to minimize

and to c



- **Smooth**

- **Occluding boundaries**, to constraint normal at occluding boundaries since are known

Shape from Shading

- To obtain the depth from its gradient, we can minimize over Z

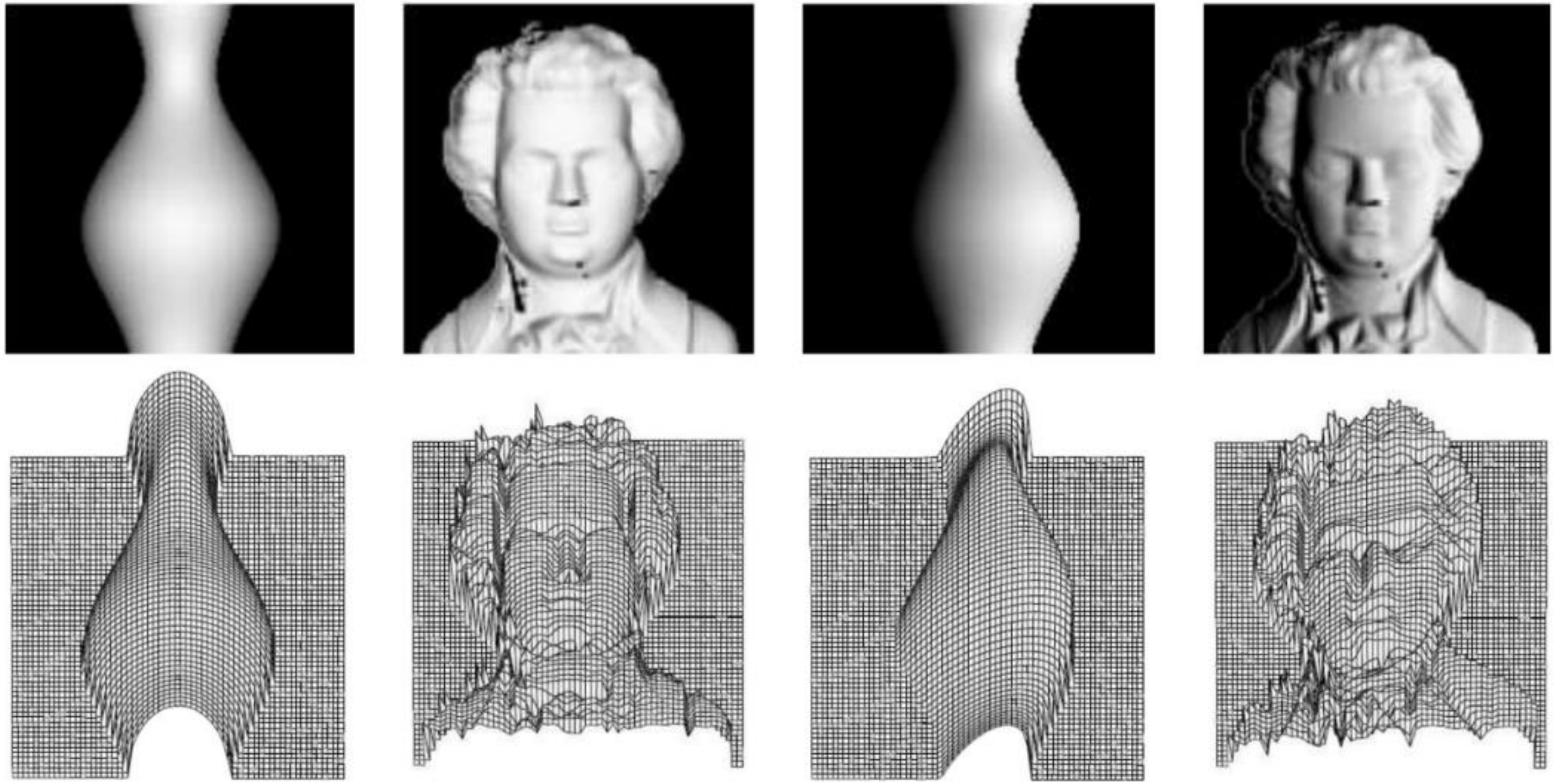
$$E(z) = \int\int \left(\frac{\partial z}{\partial x} + p\right)^2 + \left(\frac{\partial z}{\partial y} + q\right)^2 dx\, dy$$

where, as we know, *(p, q)* are

$$(p, q) = \left(-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}\right)$$

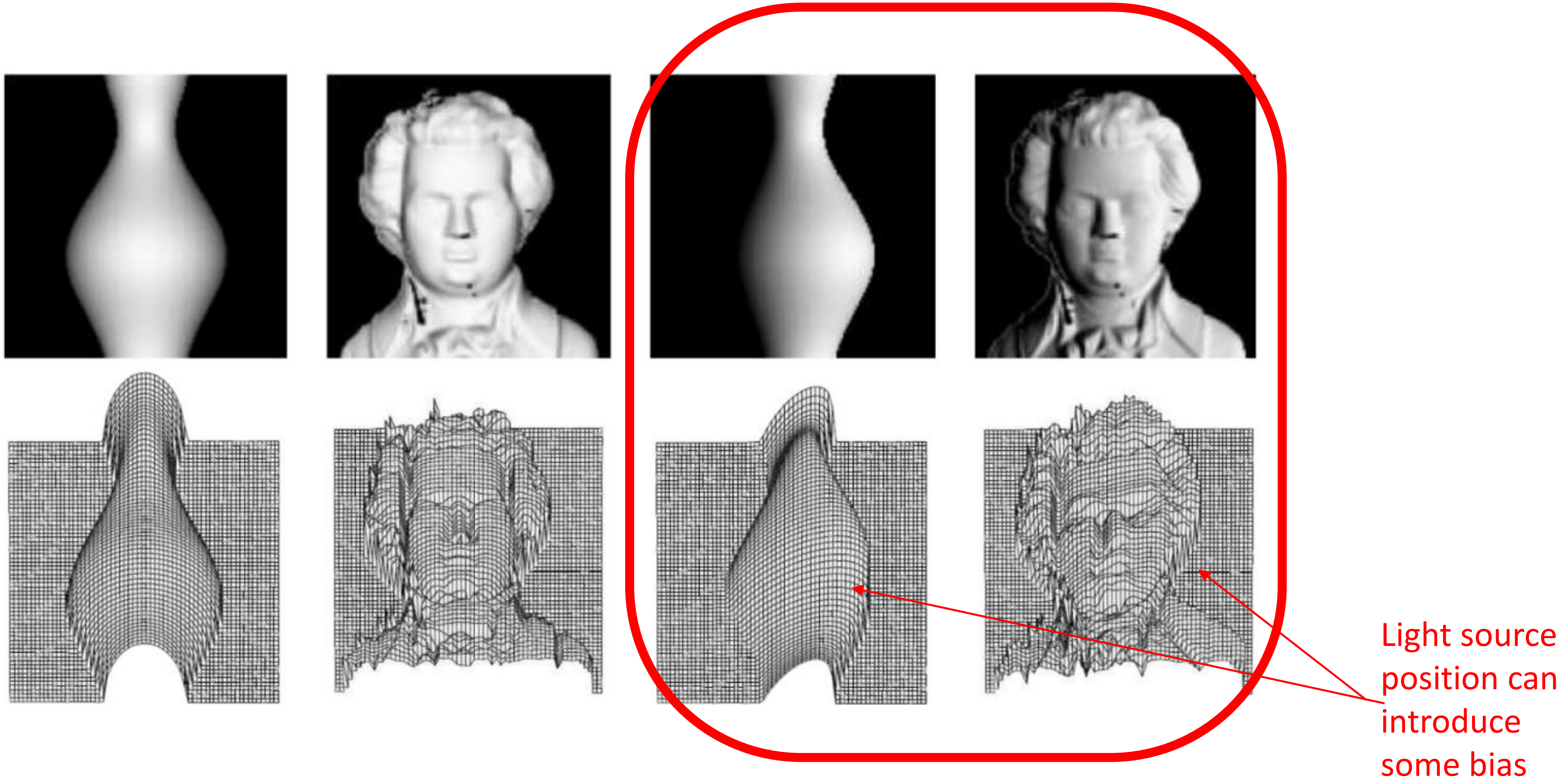
Slide from A. Geiger

Shape from Shading



Frankot and Chellappa: A Method for enforcing integrability in shape from shading algorithms. TPAMI, 1988.

Shape from Shading



Frankot and Chellappa: A Method for enforcing integrability in shape from shading algorithms. TPAMI, 1988.

3D Reconstruction

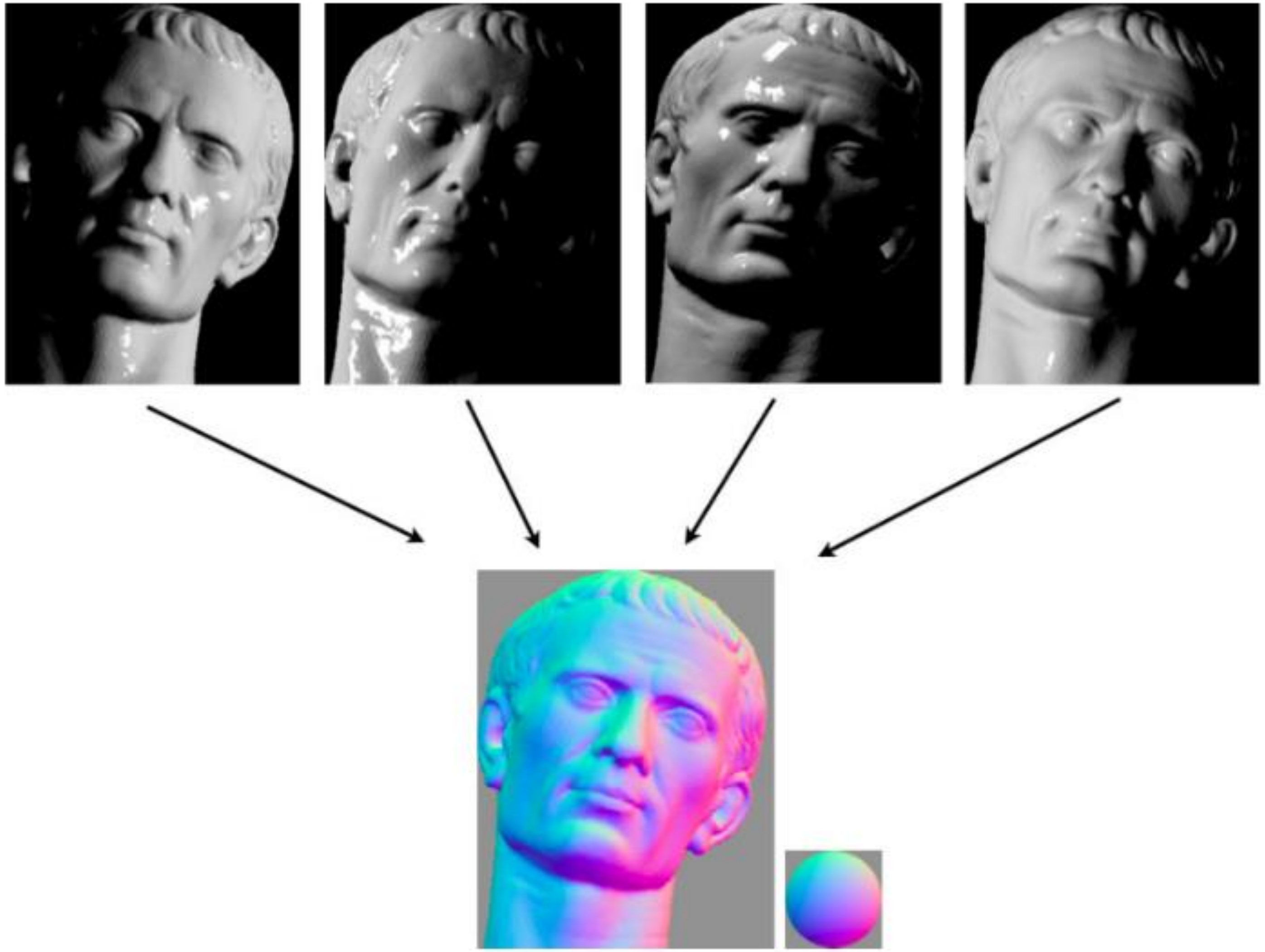
Photometric Stereo

Photometric Stereo

- To address the ambiguity in the Shape from Shading problem, instead of imposing strong smoothness constraints, we can use **multiple measurements** for a single pixel

- Idea: take multiple images **from a fixed point of view** by changing the light source position

- The **light source positions must be known** (we can use some calibration procedures)



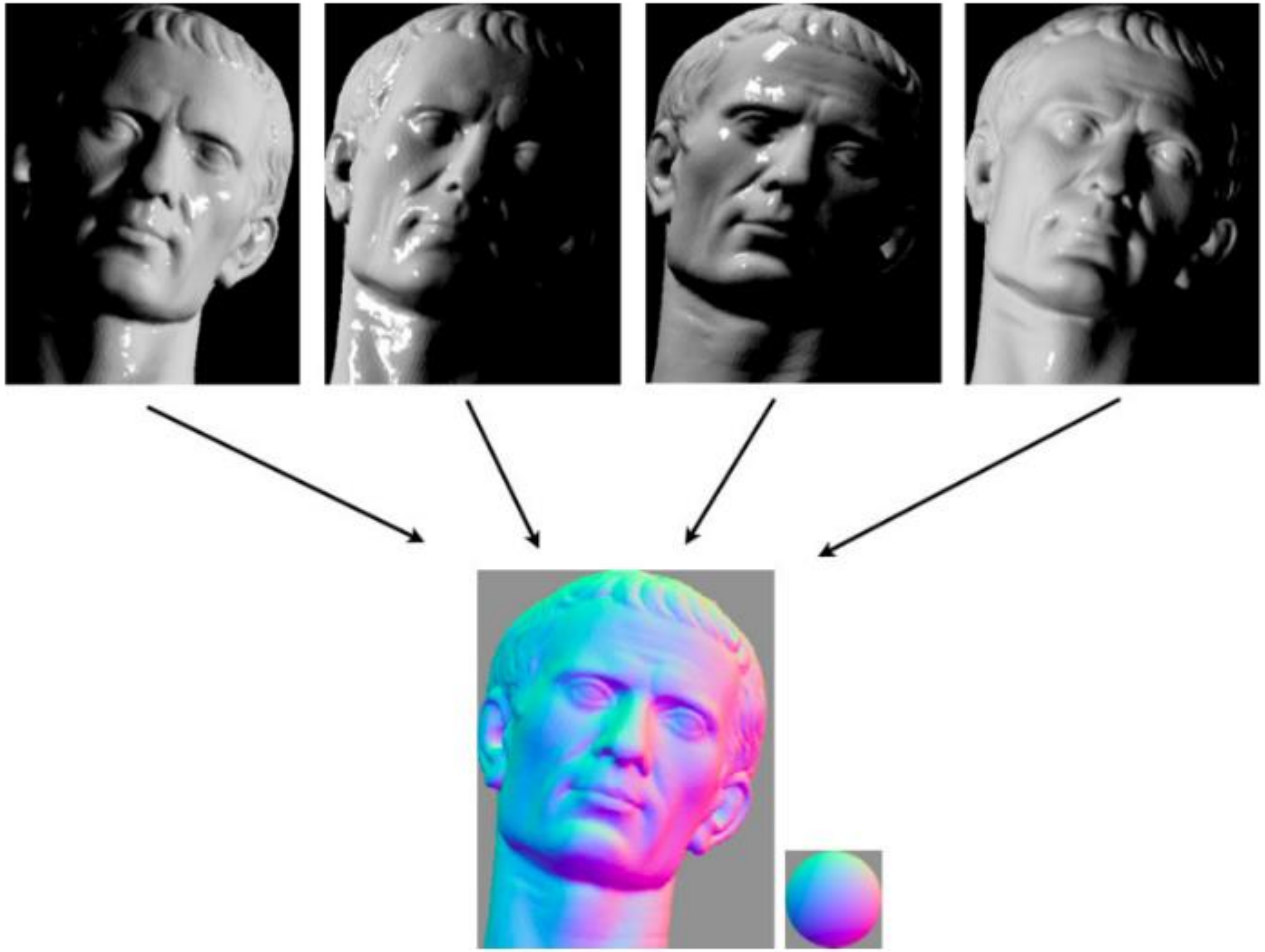
Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

Photometric Stereo

- We can also estimate the **albedo** (ρ) of the surface

- So we have
  - 2 DoF for the normal vector
  - 1 DoF for the albedo

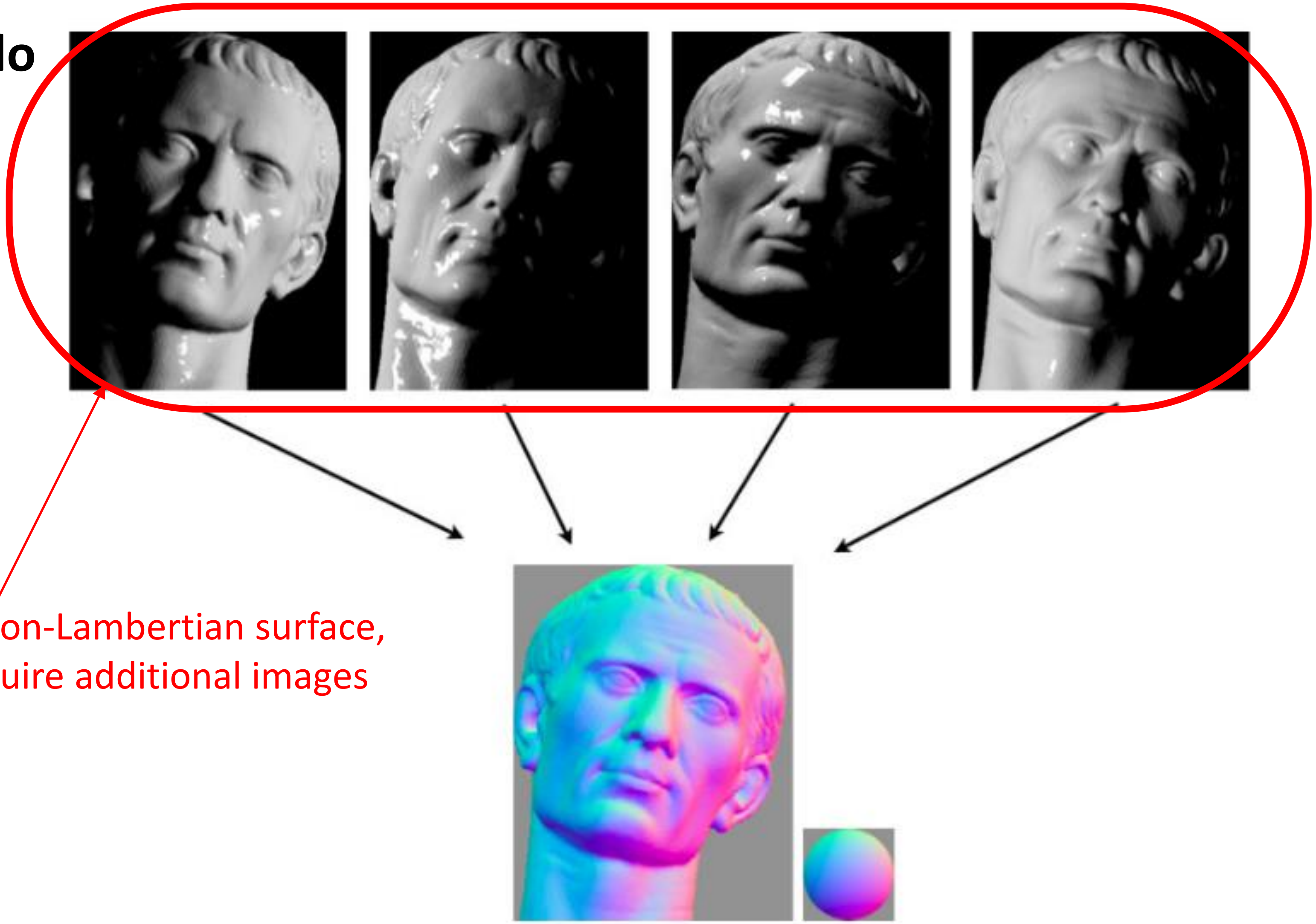
- We need at lest **three observation**



Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

Photometric Stereo

- We can also estimate the **albedo** (ρ) of the surface
- So we have
  - 2 DoF for the normal vector
  - 1 DoF for the albedo
- We need at lest **three observation**

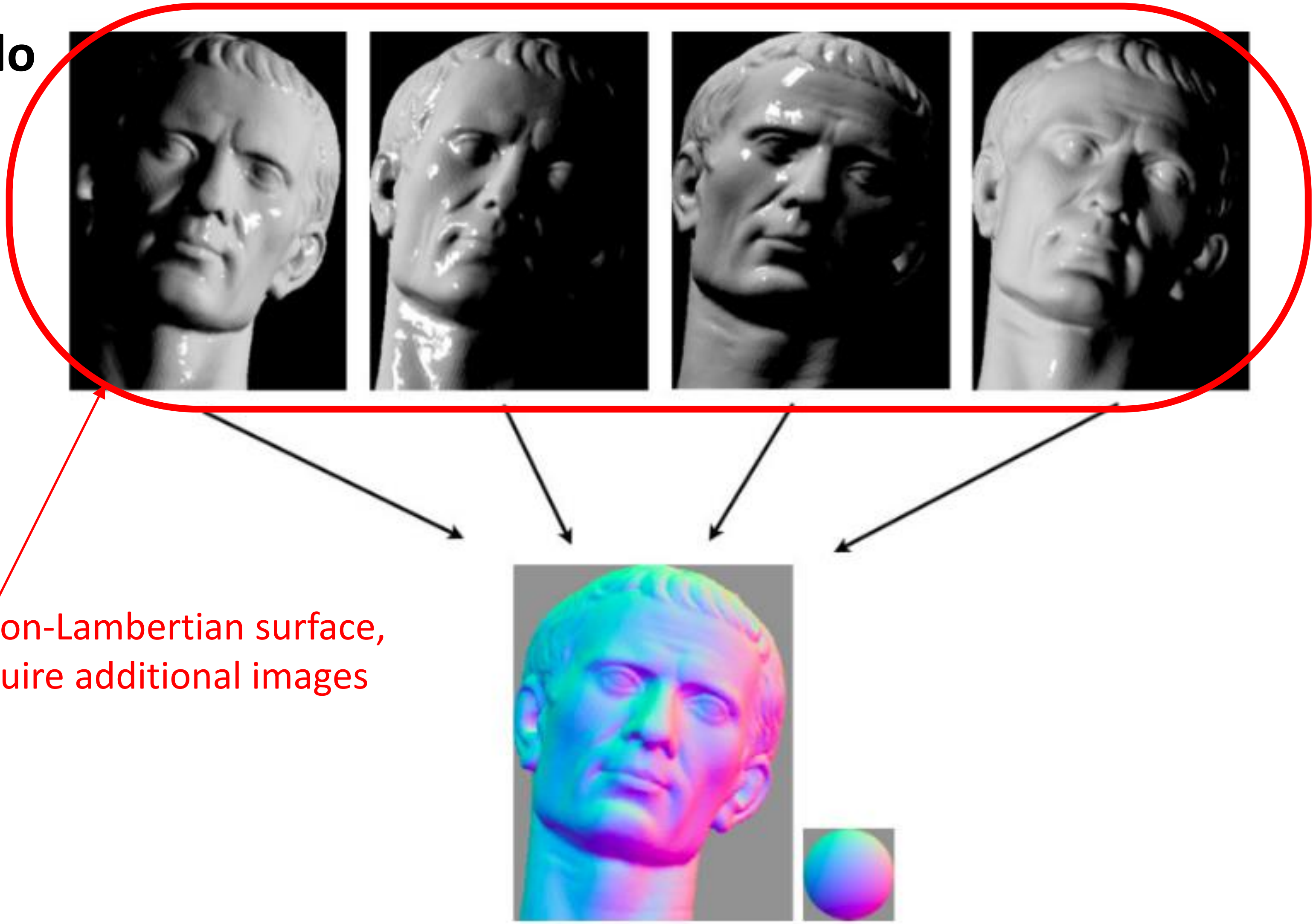


This is a non-Lambertian surface, so we require additional images

Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

Photometric Stereo

- We can also estimate the **albedo** (ρ) of the surface
- So we have
  - 2 DoF for the normal vector
  - 1 DoF for the albedo
- We need at lest **three observation**
- We assume that the light source is far away



This is a non-Lambertian surface, so we require additional images

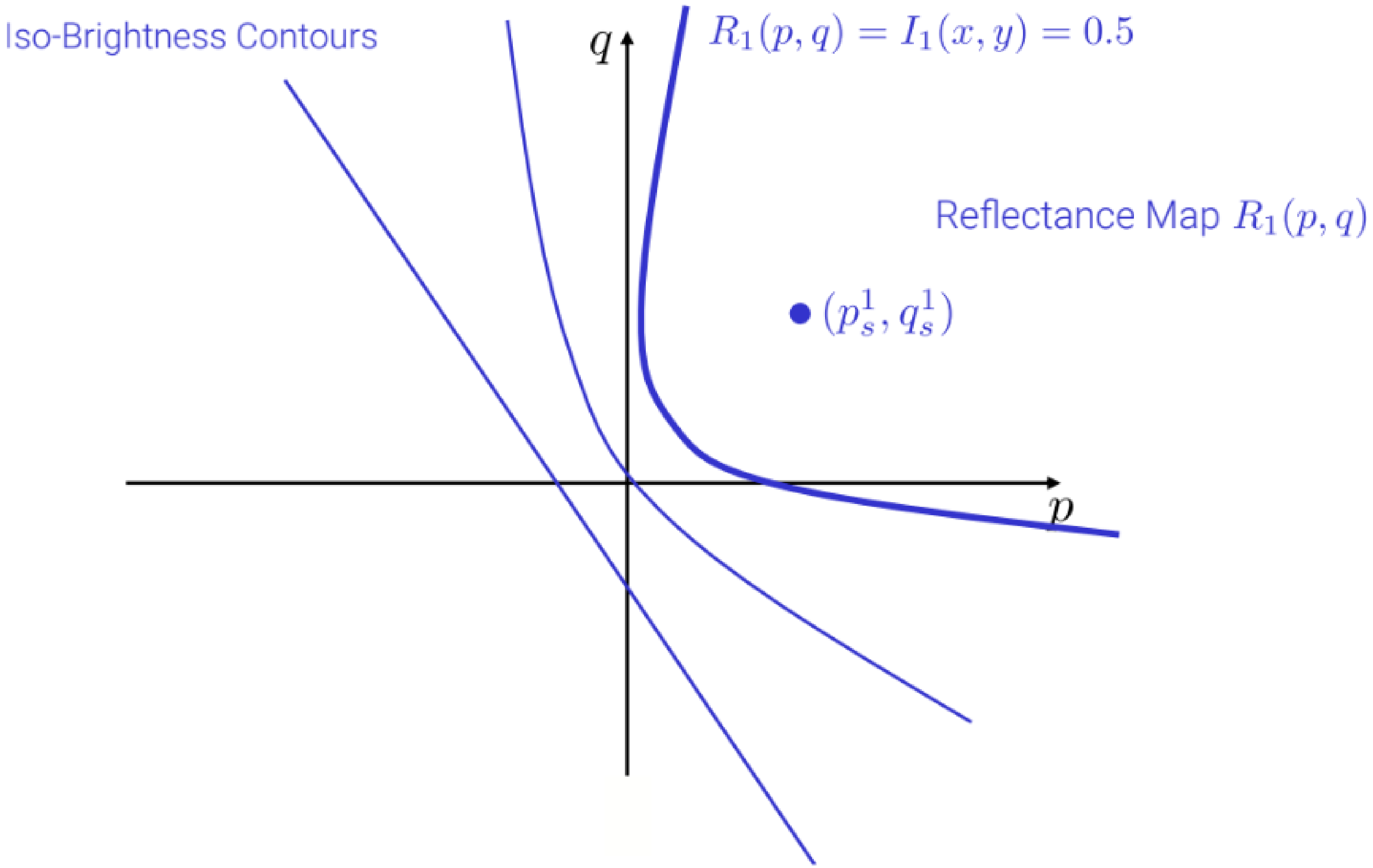
Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

Photometric Stereo

- Light stage

https://www.esperhq.com/product/lightcage-scanning-rig/

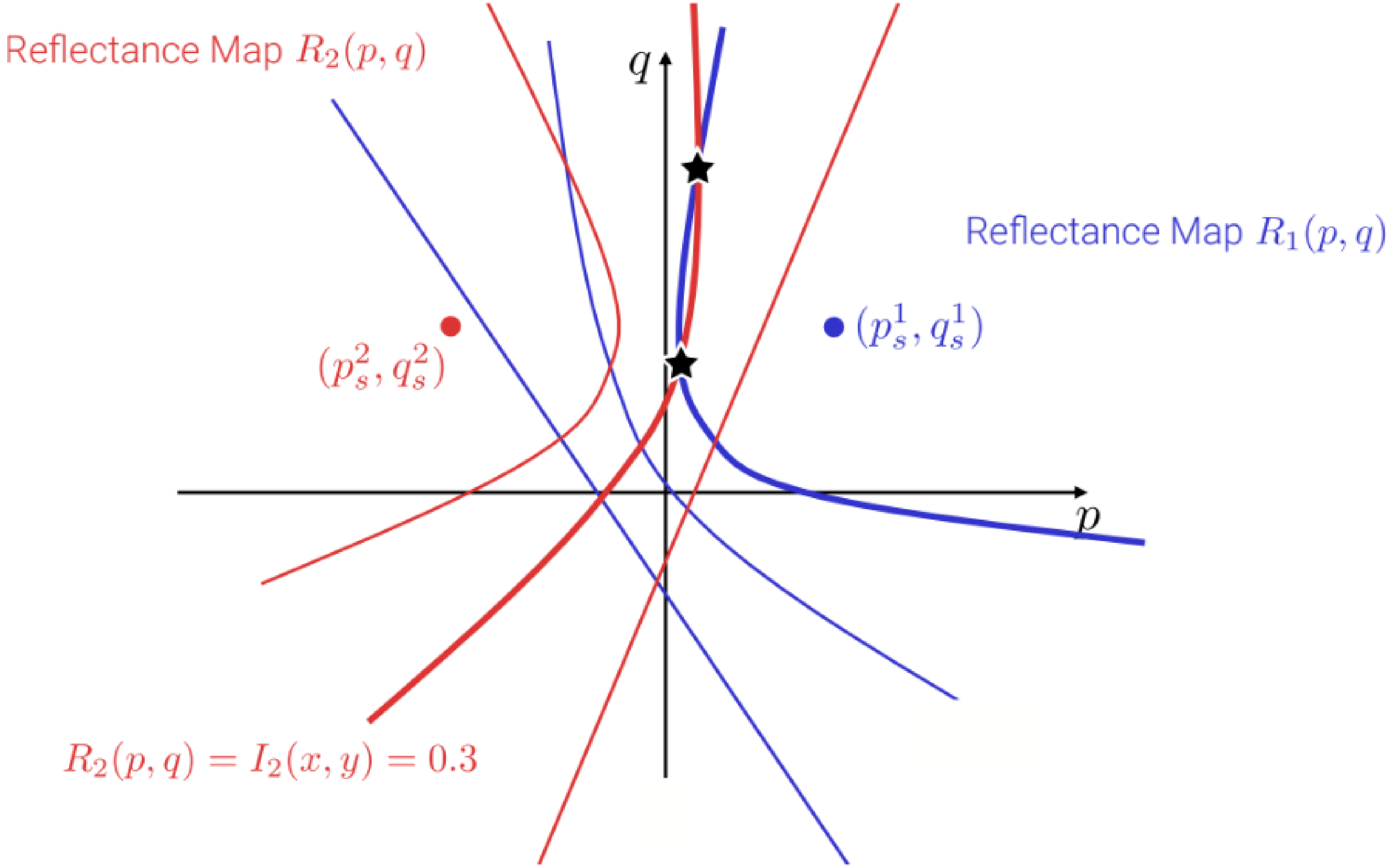
Photometric Stereo



- Capturing K images is possible to obtain an unique solution

Slide from A. Geiger

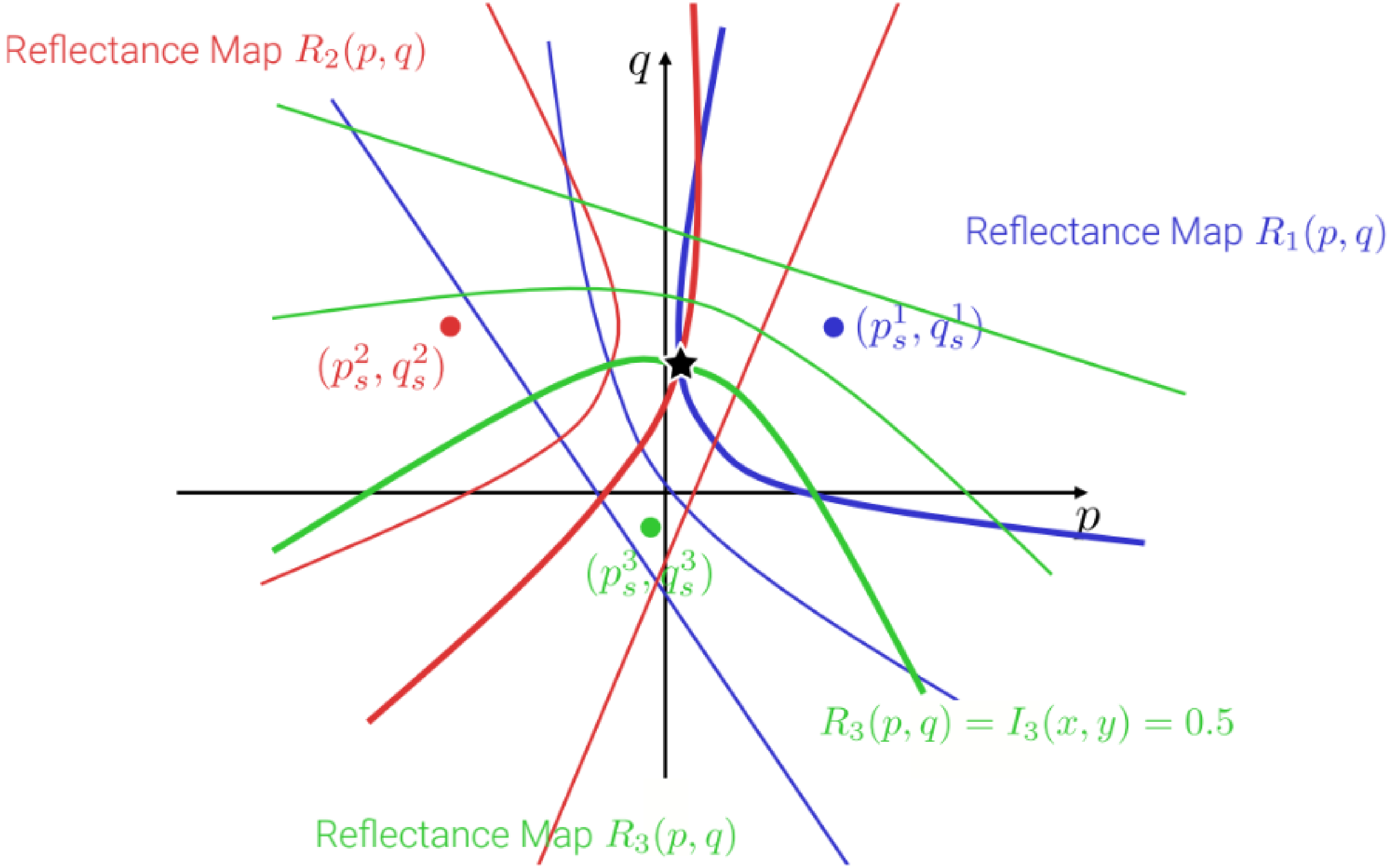
Photometric Stereo



- Capturing K images is possible to obtain an unique solution

Slide from A. Geiger

Photometric Stereo



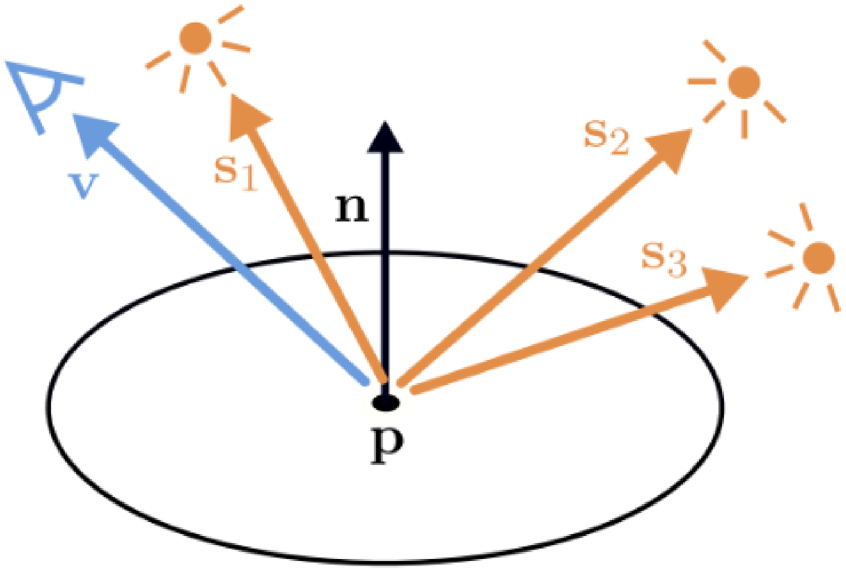
- Capturing K images is possible to obtain an unique solution

Slide from A. Geiger

Photometric Stereo

- Given the reflectance function $R(\mathbf{n}) = \rho\mathbf{n}^\top\mathbf{s} = \mathrm{I}(x, y)$ where $L_{\text{in}} = 1$ we can define the following linear system considering three observations

$$\underbrace{\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix}}_{\mathbf{I}} = \underbrace{\begin{pmatrix} \mathbf{s}_1^\top \\ \mathbf{s}_2^\top \\ \mathbf{s}_3^\top \end{pmatrix}}_{\mathbf{S}} \underbrace{\rho\,\mathbf{n}}_{\tilde{\mathbf{n}}}$$



Slide from A. Geiger

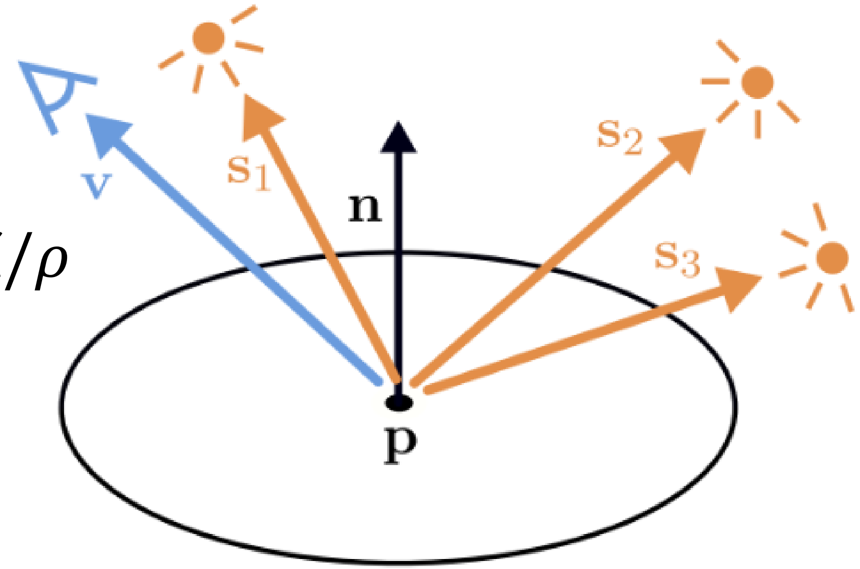
Photometric Stereo

- Given the reflectance function $R(\mathbf{n}) = \rho\mathbf{n}^\top\mathbf{s} = \mathrm{I}(x, y)$ where $L_{\mathrm{in}} = 1$ we can define the following linear system considering three observations

$$\underbrace{\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix}}_{\mathbf{I}} = \underbrace{\begin{pmatrix} \mathbf{s}_1^\top \\ \mathbf{s}_2^\top \\ \mathbf{s}_3^\top \end{pmatrix}}_{\mathbf{S}} \underbrace{\rho\,\mathbf{n}}_{\tilde{\mathbf{n}}}$$

- Then, the solution can be obtained as

$$\tilde{\mathbf{n}} = \mathrm{S}^{-1}\mathrm{I},\ \rho = \|\tilde{\mathbf{n}}\|_2 \text{ and so } \mathbf{n} = \tilde{\mathbf{n}}/\rho$$



Slide from A. Geiger

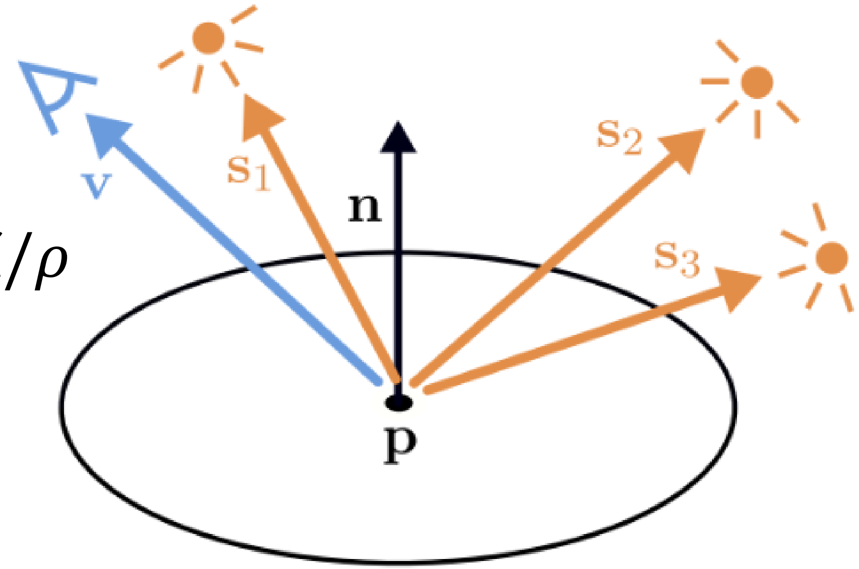
Photometric Stereo

- Given the reflectance function $R(\mathbf{n}) = \rho\mathbf{n}^\top\mathbf{s} = \mathrm{I}(x, y)$ where $L_{\text{in}} = 1$ we can define the following linear system considering three observations

$$\underbrace{\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix}}_{\mathbf{I}} = \underbrace{\begin{pmatrix} \mathbf{s}_1^\top \\ \mathbf{s}_2^\top \\ \mathbf{s}_3^\top \end{pmatrix}}_{\mathbf{S}} \underbrace{\rho\,\mathbf{n}}_{\tilde{\mathbf{n}}}$$

- Then, the solution can be obtained as

$$\tilde{\mathbf{n}} = \mathrm{S}^{-1}\mathrm{I},\ \rho = \|\tilde{\mathbf{n}}\|_2 \text{ and so } \mathbf{n} = \tilde{\mathbf{n}}/\rho$$

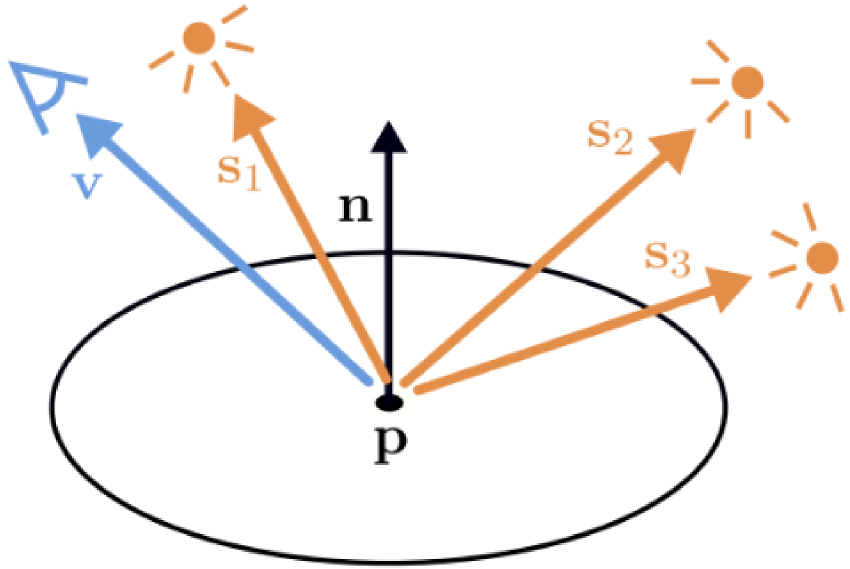


- Note that, we do not use the gradient space, but we parametrized $\mathbf{n}$ as a 3D vector and using ρ as its norm

Slide from A. Geiger

Photometric Stereo

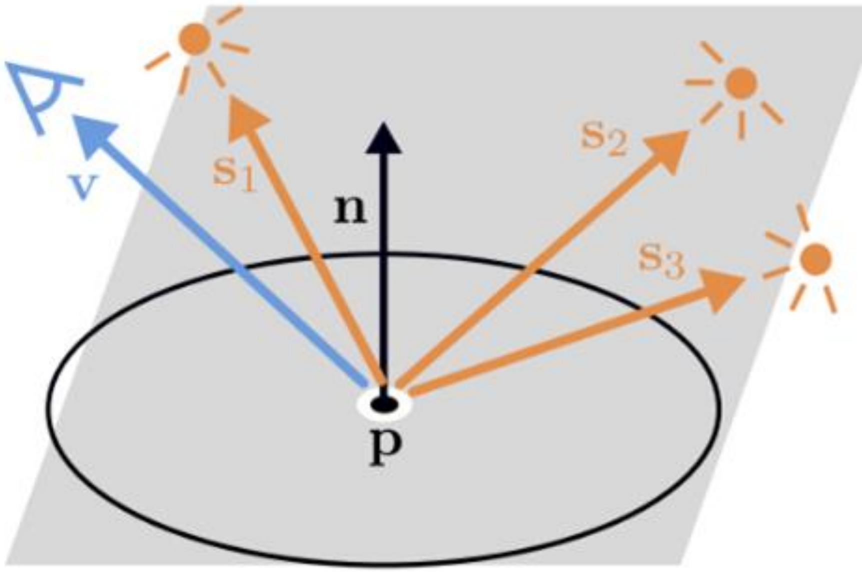
- In order to work, the S matrix must have full rank to be invertible



Photometric Stereo

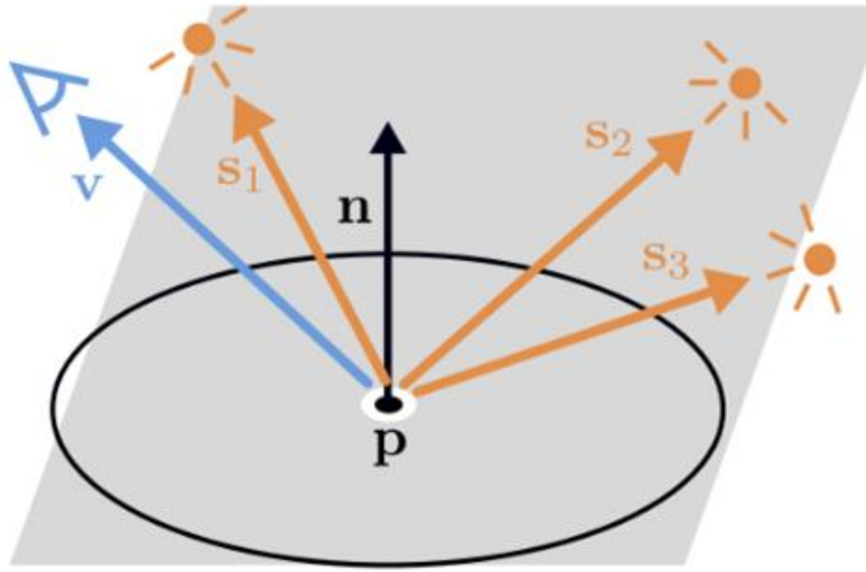
- In order to work, the S matrix must have full rank to be invertible

- This does not happen if the light sources and the points lies on the same 3D plane



Photometric Stereo

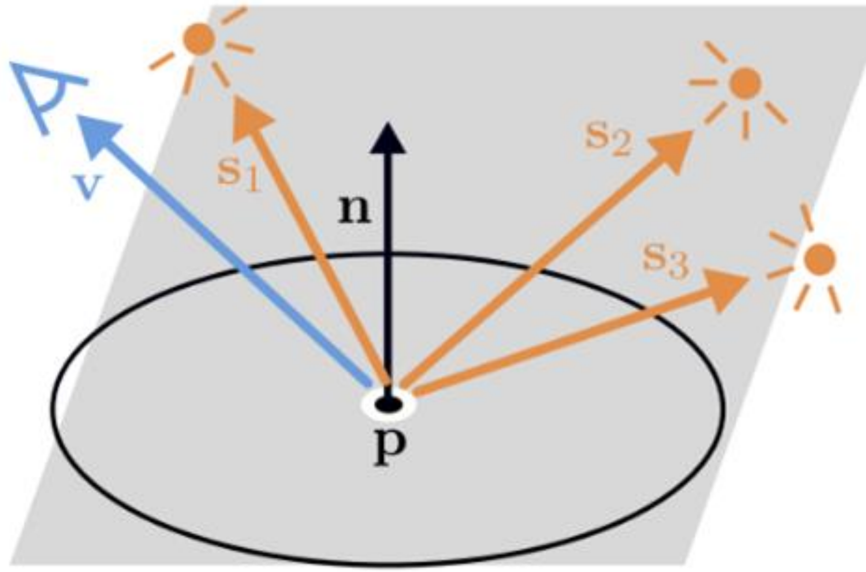
- In order to work, the S matrix must have full rank to be invertible
- This does not happen if the light sources and the points lies on the same 3D plane
- In that case, one of the light direction vector $\mathbf{s}$ become linearly dependent from the other two, and S is rank deficient



Slide from A. Geiger

Photometric Stereo

- In order to work, the S matrix must have full rank to be invertible
- This does not happen if the light sources and the points lies on the same 3D plane
- In that case, one of the light direction vector $\mathbf{s}$ become linearly dependent from the other two, and S is rank deficient



Slide from A. Geiger

Photometric Stereo

- If you have more than three observations, you can solve an over-constrained system

$$\underbrace{\begin{pmatrix} I_1 \\ \vdots \\ I_K \end{pmatrix}}_{\mathbf{I}} = \underbrace{\begin{pmatrix} \mathbf{s}_1^\top \\ \vdots \\ \mathbf{s}_K^\top \end{pmatrix}}_{\mathbf{S}} \underbrace{\rho\,\mathbf{n}}_{\tilde{\mathbf{n}}}$$

Slide from A. Geiger

Photometric Stereo

- If you have more than three observations, you can solve an over-constrained system

$$\underbrace{\begin{pmatrix} I_1 \\ \vdots \\ I_K \end{pmatrix}}_{\mathbf{I}} = \underbrace{\begin{pmatrix} \mathbf{s}_1^\top \\ \vdots \\ \mathbf{s}_K^\top \end{pmatrix}}_{\mathbf{S}} \underbrace{\rho\,\mathbf{n}}_{\tilde{\mathbf{n}}}$$

- And the solution can be obtained with

$$\mathbf{S}^\top \mathbf{I} = \mathbf{S}^\top \mathbf{S} \tilde{\mathbf{n}}$$

$$\tilde{\mathbf{n}} = \underbrace{(\mathbf{S}^\top \mathbf{S})^{-1} \mathbf{S}^\top}_{\text{Pseudoinverse}} \mathbf{I}$$

Slide from A. Geiger

Photometric Stereo

- If you have more than three observations, you can solve an over-constrained system

$$\underbrace{\begin{pmatrix} I_1 \\ \vdots \\ I_K \end{pmatrix}}_{\mathbf{I}} = \underbrace{\begin{pmatrix} \mathbf{s}_1^\top \\ \vdots \\ \mathbf{s}_K^\top \end{pmatrix}}_{\mathbf{S}} \underbrace{\rho\,\mathbf{n}}_{\tilde{\mathbf{n}}}$$

- And the solution can be obtained with

$$\mathbf{S}^\top \mathbf{I} = \mathbf{S}^\top \mathbf{S} \tilde{\mathbf{n}}$$

$$\tilde{\mathbf{n}} = \underbrace{(\mathbf{S}^\top \mathbf{S})^{-1} \mathbf{S}^\top}_{\text{Pseudoinverse}} \mathbf{I}$$

Note that, we still suppose to have a Lambertian surface. To work with **non Lambertian surfaces** we have

- to relax some hypothesis on the rendering equation,

or

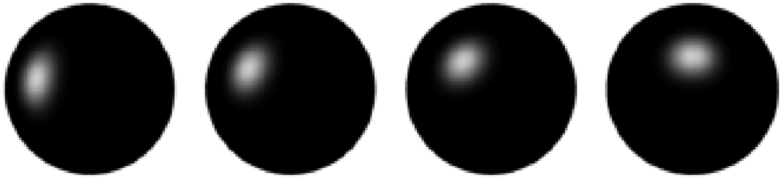
- to impose some regularization

Slide from A. Geiger

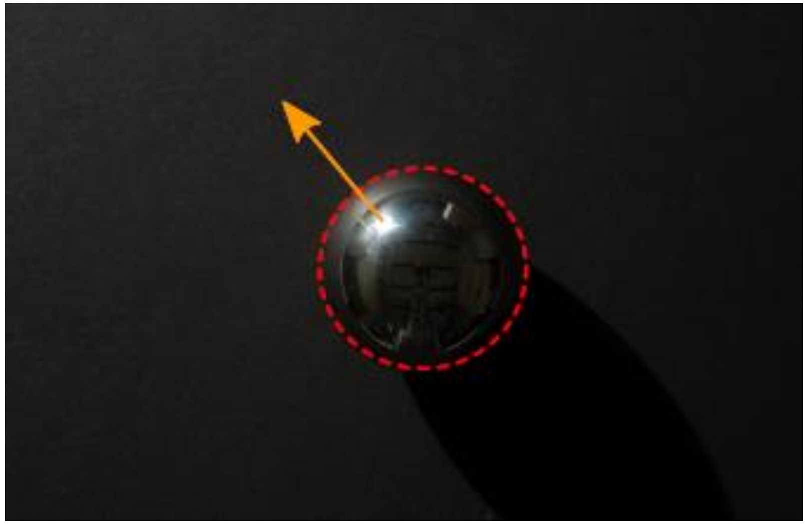
Photometric Stereo

Calibration of light source directions

- To obtain the position of the light sources in a controlled setup we can use a sphere with specular surface



- Its geometry (i.e., the normal vectors) it is known



- By detecting the specular reflection over the sphere, the light source direction can be estimated

Photometric Stereo

Input

Normal map

Albedo

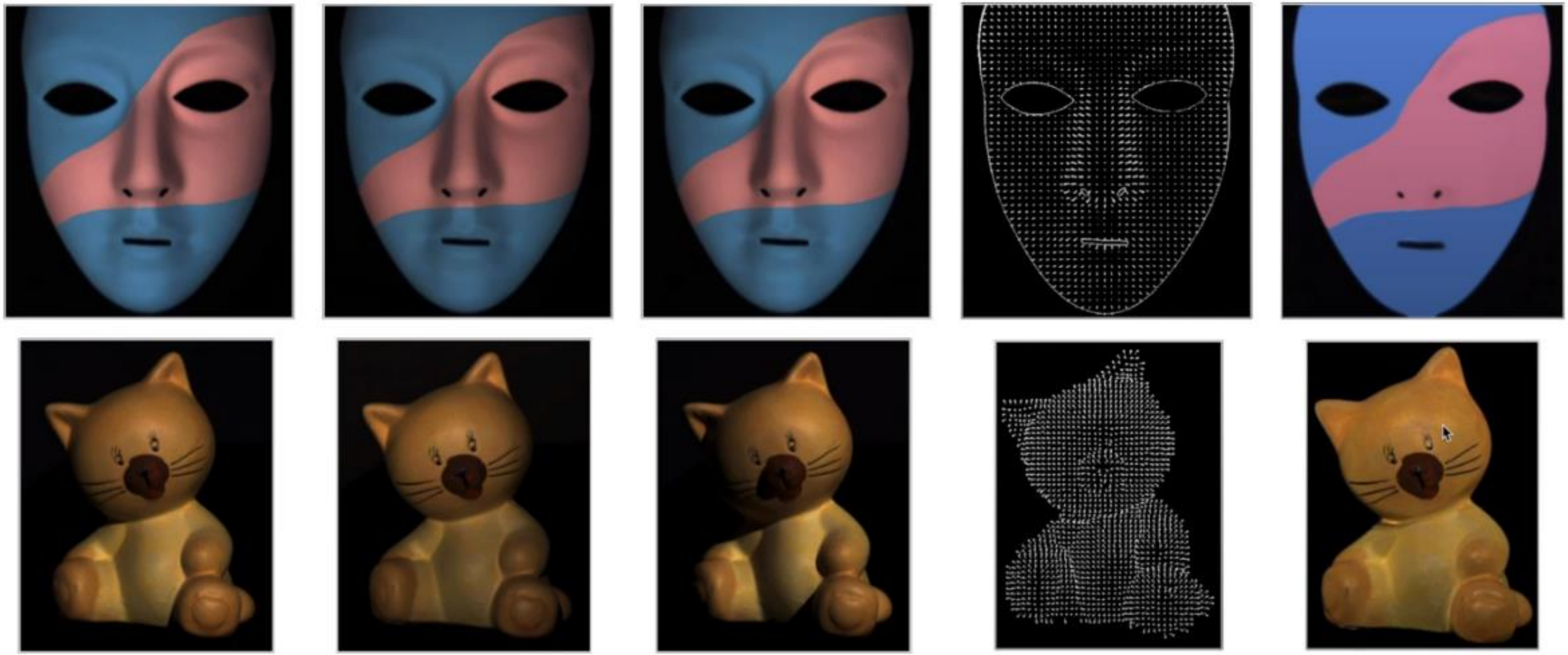
Integrated normal

Image of the
relighted object
(with uniform albedo)

Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

Photometric Stereo

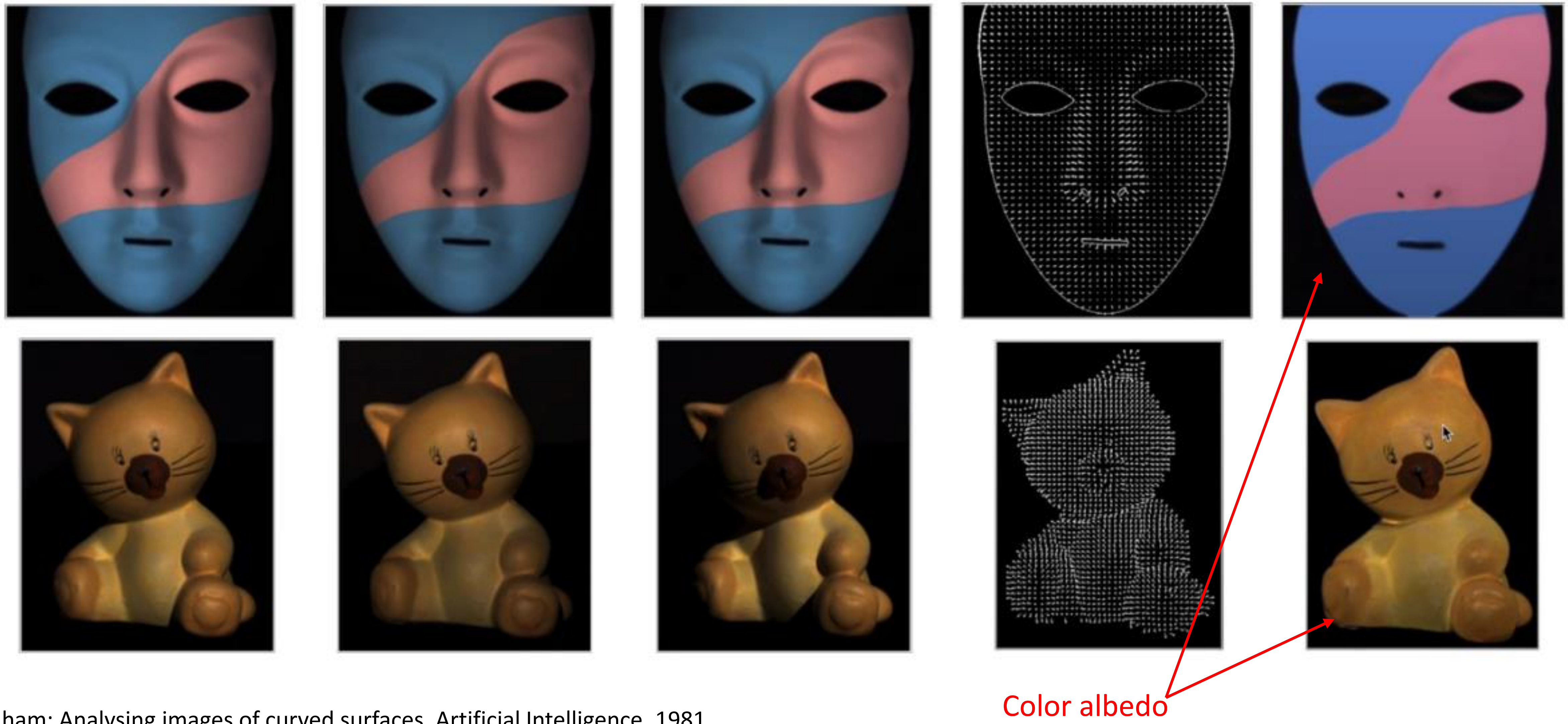
- In case of coloured images, we work separately for each single R, G, or B channel



Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

Photometric Stereo

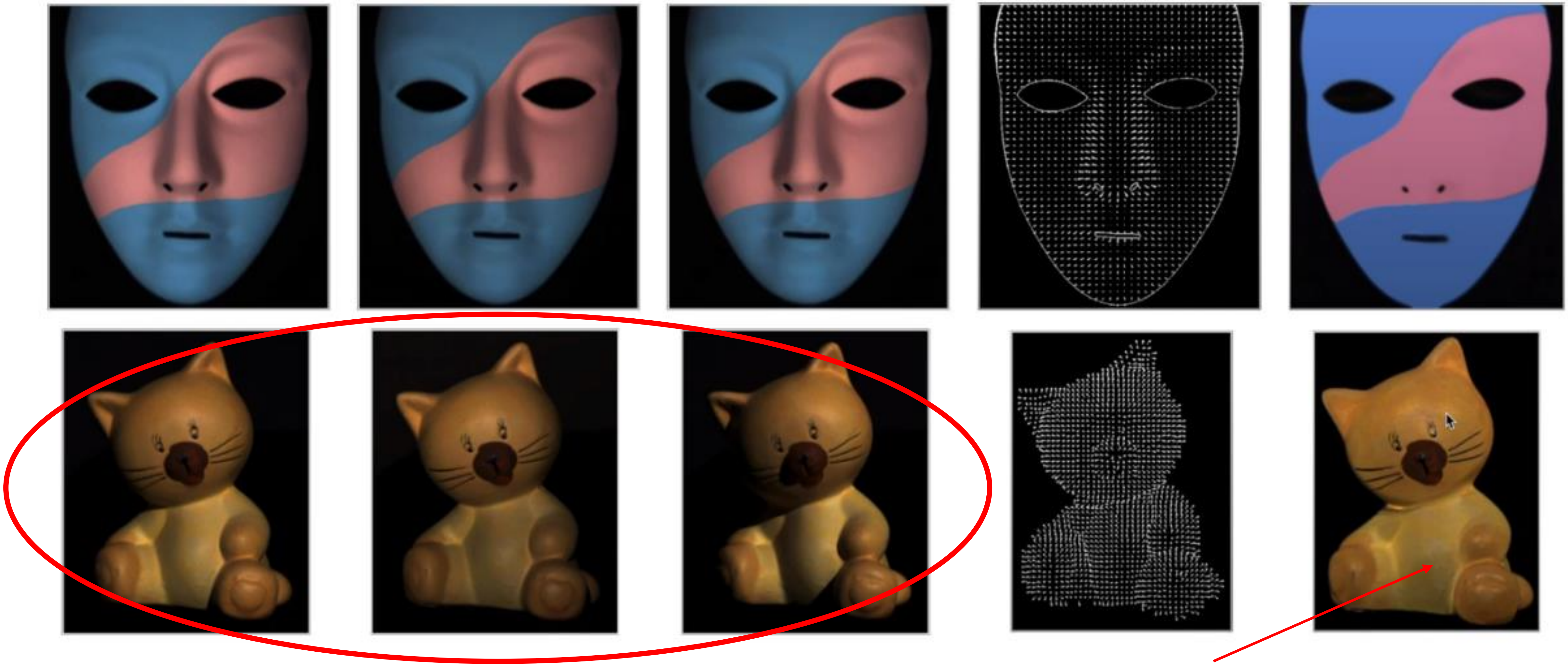
- In case of coloured images, we work separately for each single R, G, or B channel



Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

Photometric Stereo

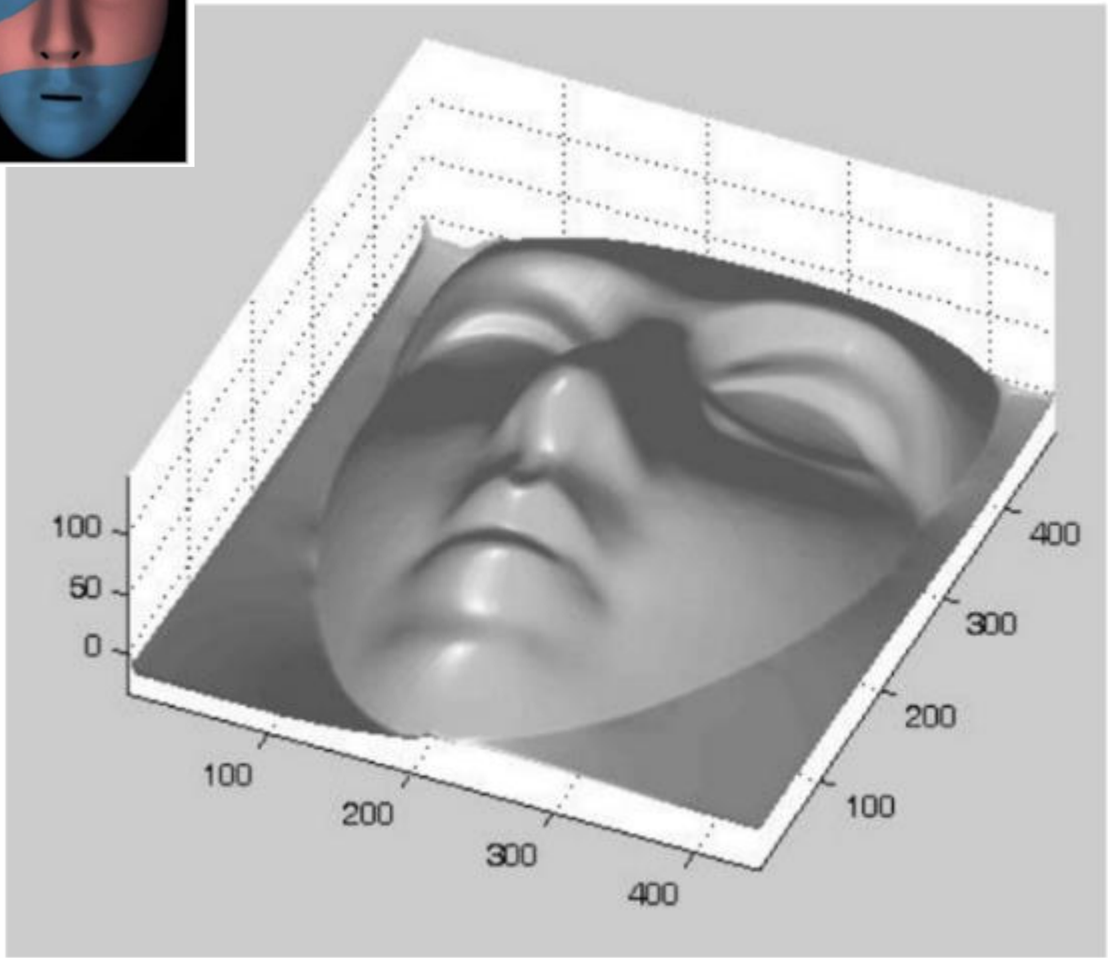
- In case of coloured images, we work separately for each single R, G, or B channel

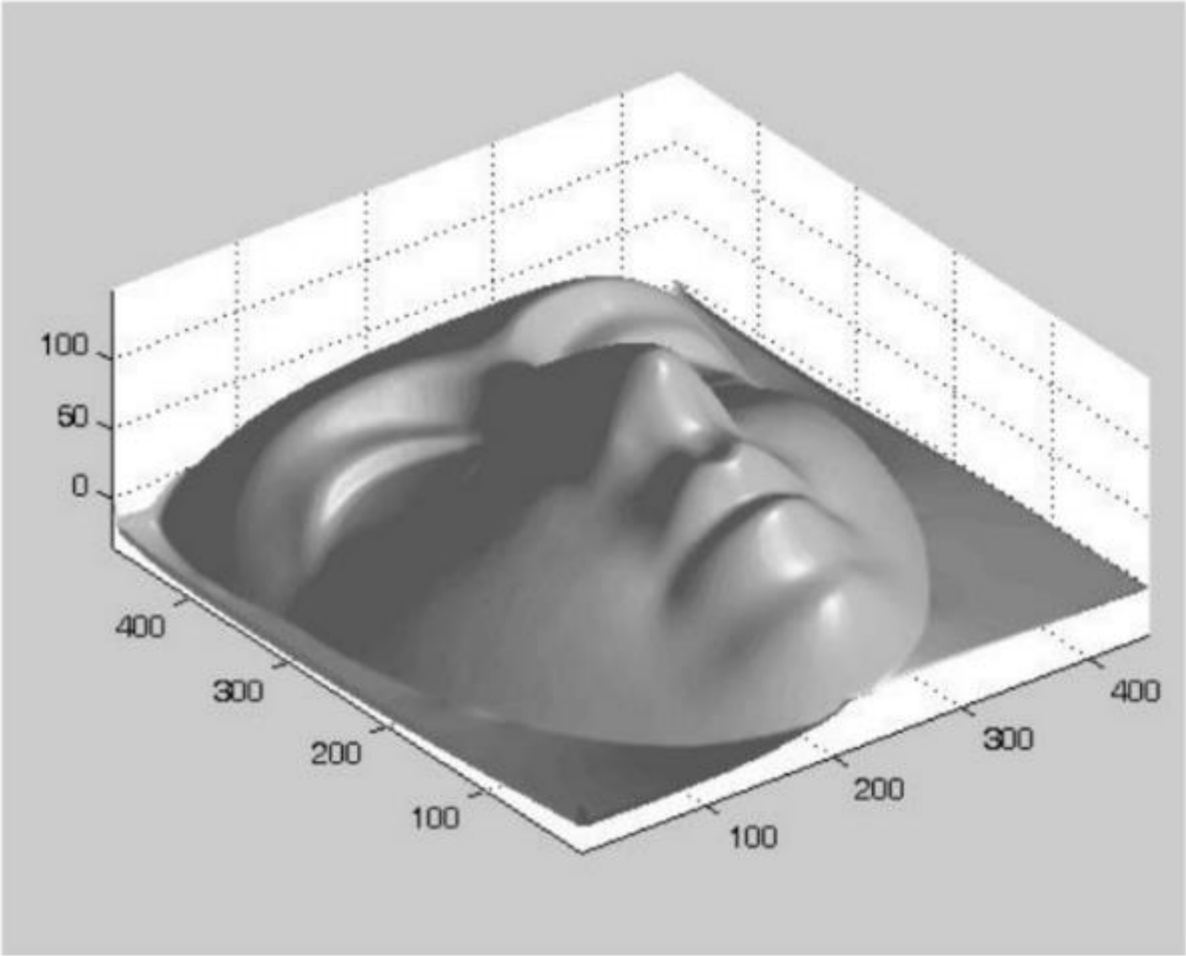


Deviating from the Lambertian assumption leads to errors, e.g., artifacts in the albedo map

Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

Photometric Stereo



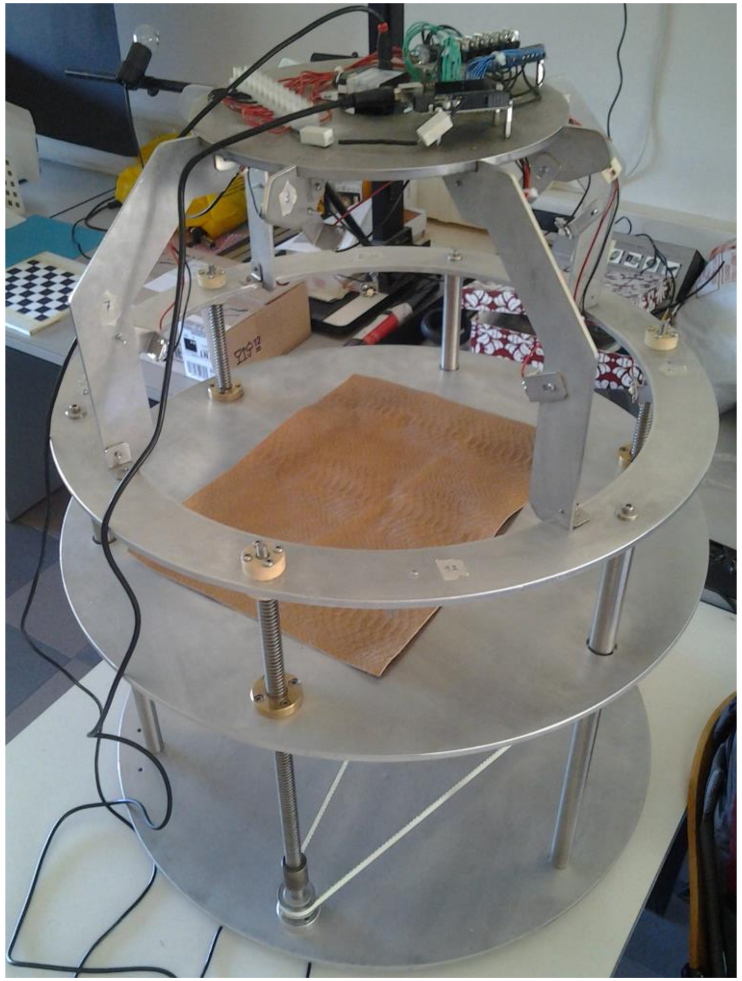


Woodham: Analysing images of curved surfaces. Artificial Intelligence, 1981.

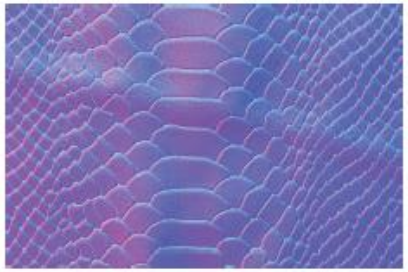
Photometric Stereo

Verbiest and Van Gool. "Photometric stereo with coherent outlier handling and confidence estimation." CVPR 2008

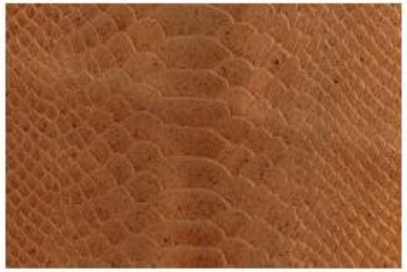
Photometric Stereo

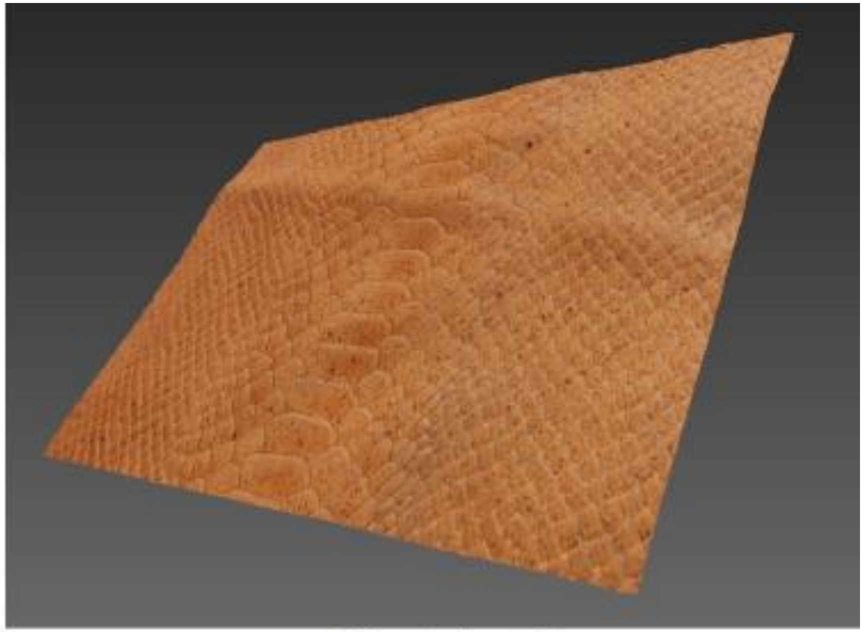


Normal Map (RGB)



Albedo Map





Modello 3D

Photometric Stereo

- Photometric stereo can also be used in outdoor scenarios

- Sun is a perfect light source

- Multiple images can be acquired during different times of the day or in different days

Ackermann, Langguth, Fuhrmann and Goesele: Photometric stereo for outdoor webcams. CVPR, 2012.

Visual Odometry
&
SLAM

Visual Odometry & SLAM

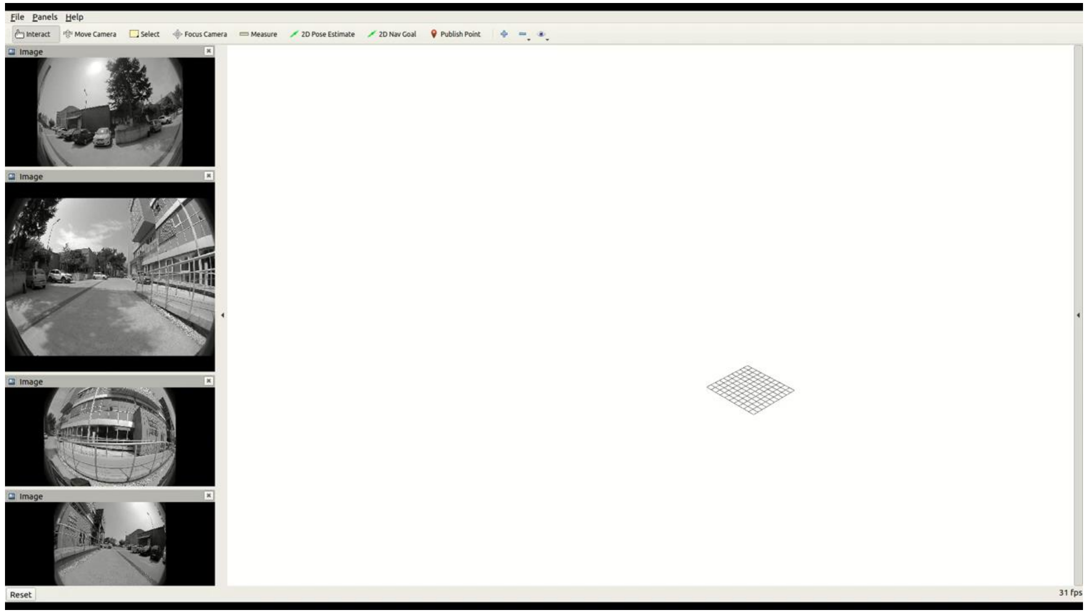
- Visual odometry is a method focused on **finding the system/camera position** in an unknown environment
- **Camera positions** (i.e., trajectory) and **structure** are estimated simultaneously and incrementally
- Images are **temporally ordered**, typically we use (live) videos
- All image are acquired by the same camera, usually with **known calibration**
- Real time constraint

Visual Odometry & SLAM

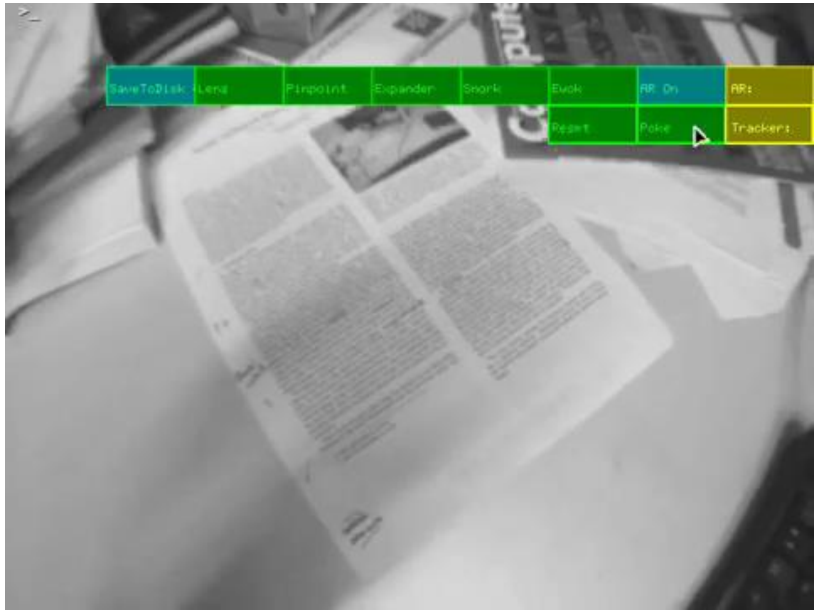
- SLAM, **Simultaneous Localization and Mapping**, is used for the same task
- We talk about SLAM when we impose **global consistency**, implemented typically using **loop-closure** solutions

Visual Odometry & SLAM

- The capability of localizing the camera in real-time is key functionality for several application, e.g., self-driving vehicle, augmented reality, etc.



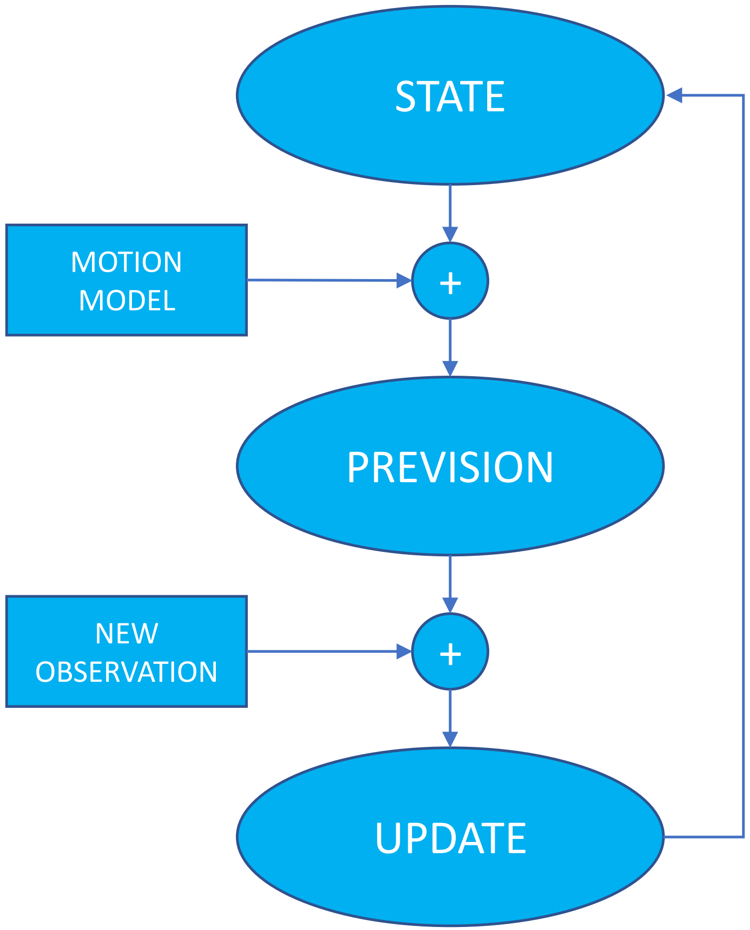
https://www.youtube.com/watch?v=LbbY3M4nt68



https://www.youtube.com/watch?v=F3s3M0mokNc

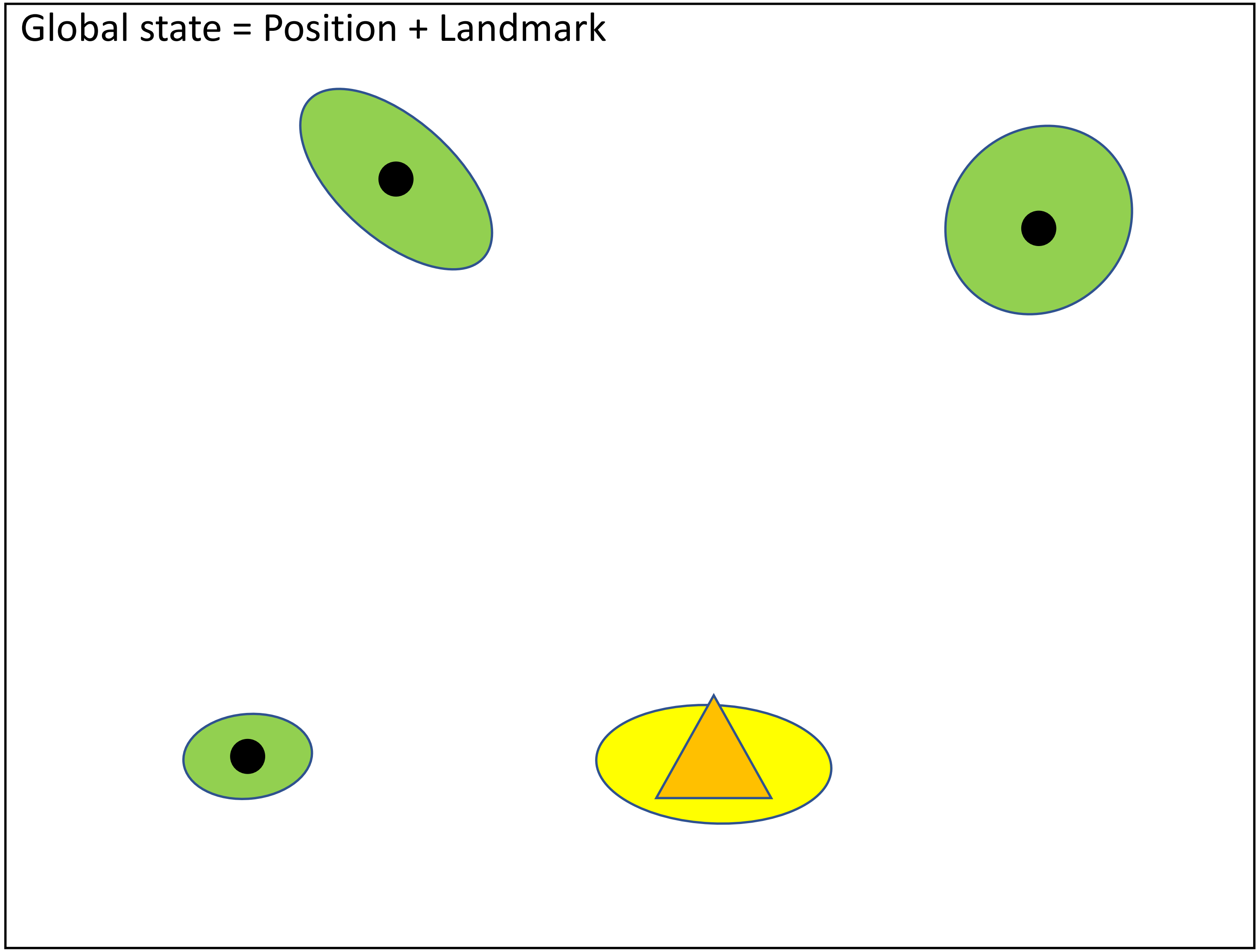
Bayesian Solution

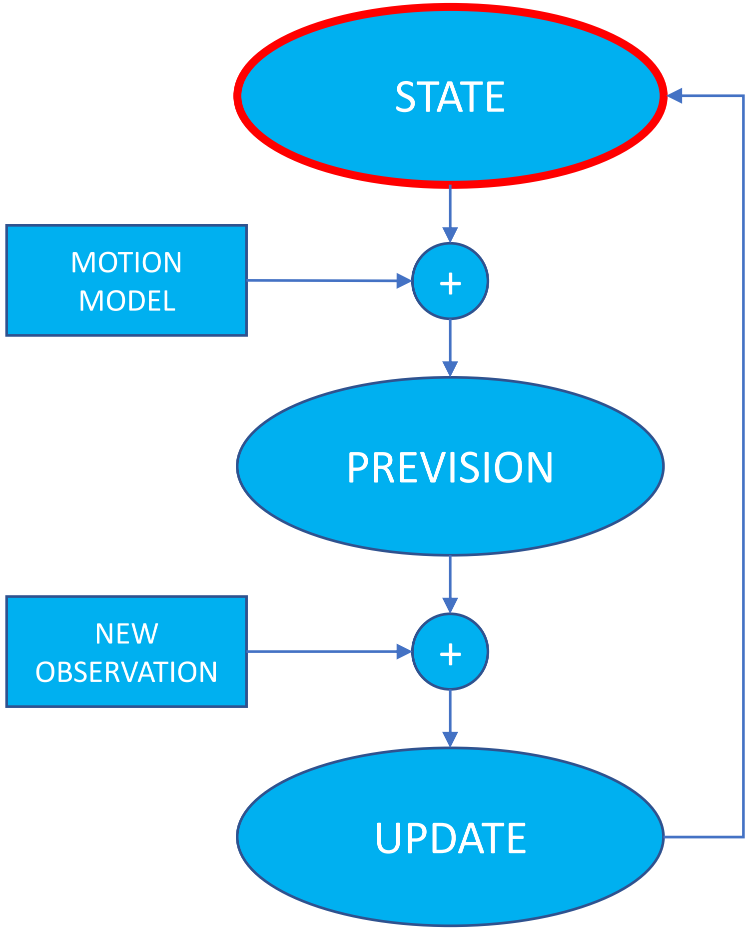
- Usually implemented with **Kalman filter**
- Camera position and 3D structure as **random variables**
- Simultaneous update of structure and trajectory
- Non-linear observation model
- **Strong limit on landmark (3D point) number**



A. J. Davison, "Real-time simultaneous localization and mapping with a single camera," ICCV, 2003

Bayesian Solution

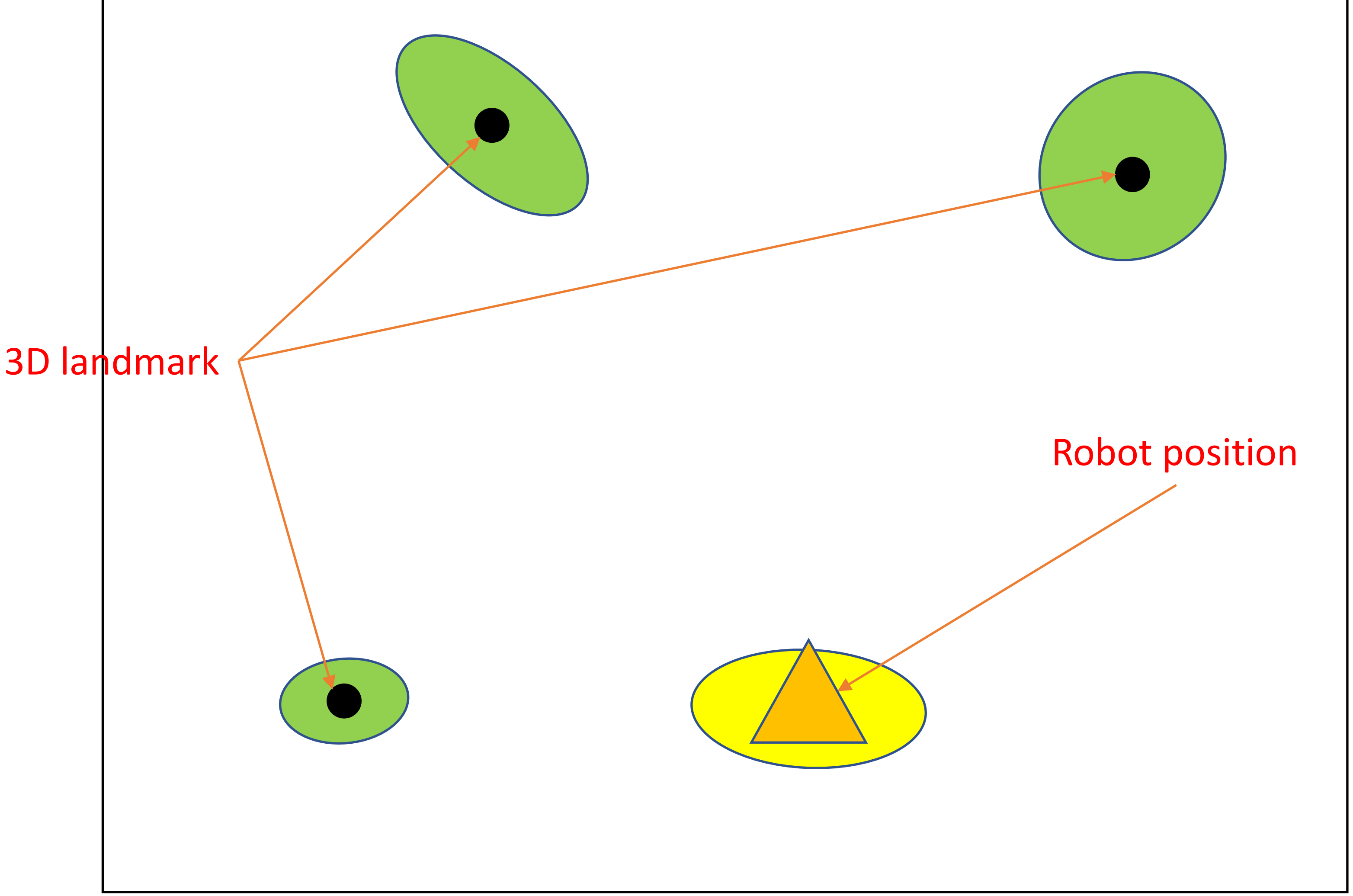


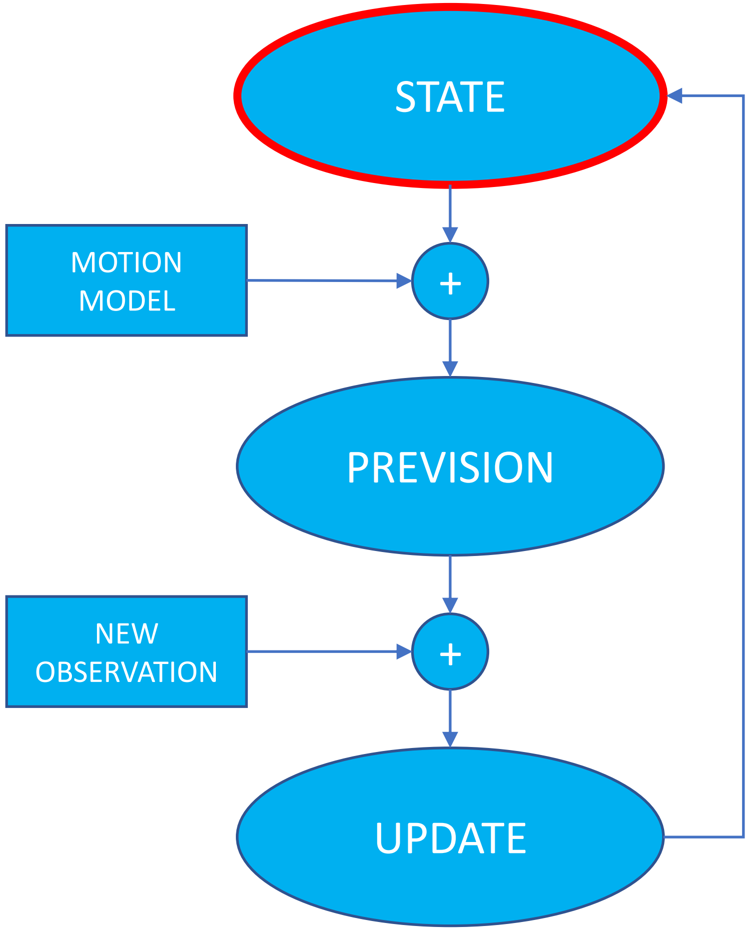


A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Bayesian Solution

Global state = Position + Landmark

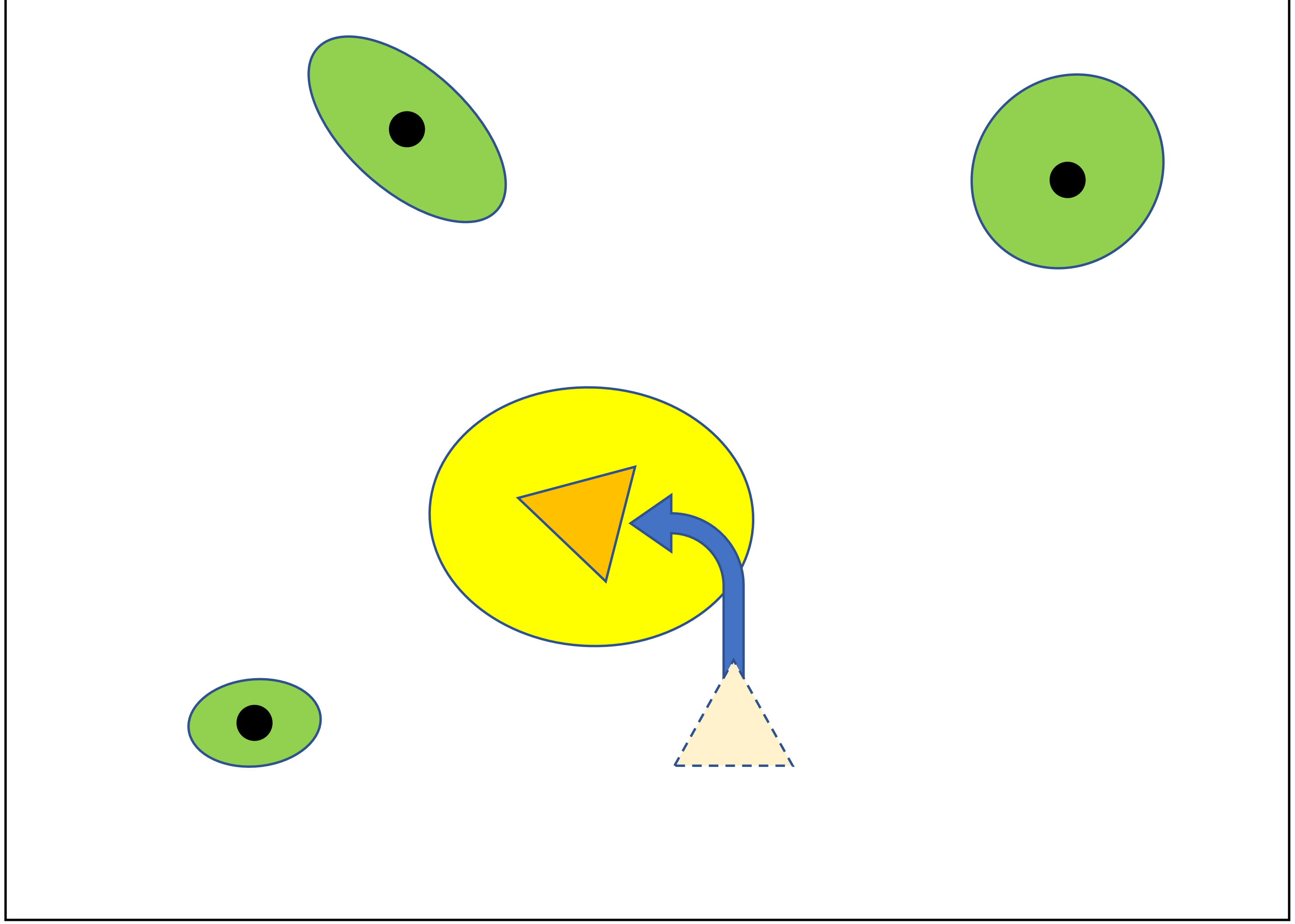


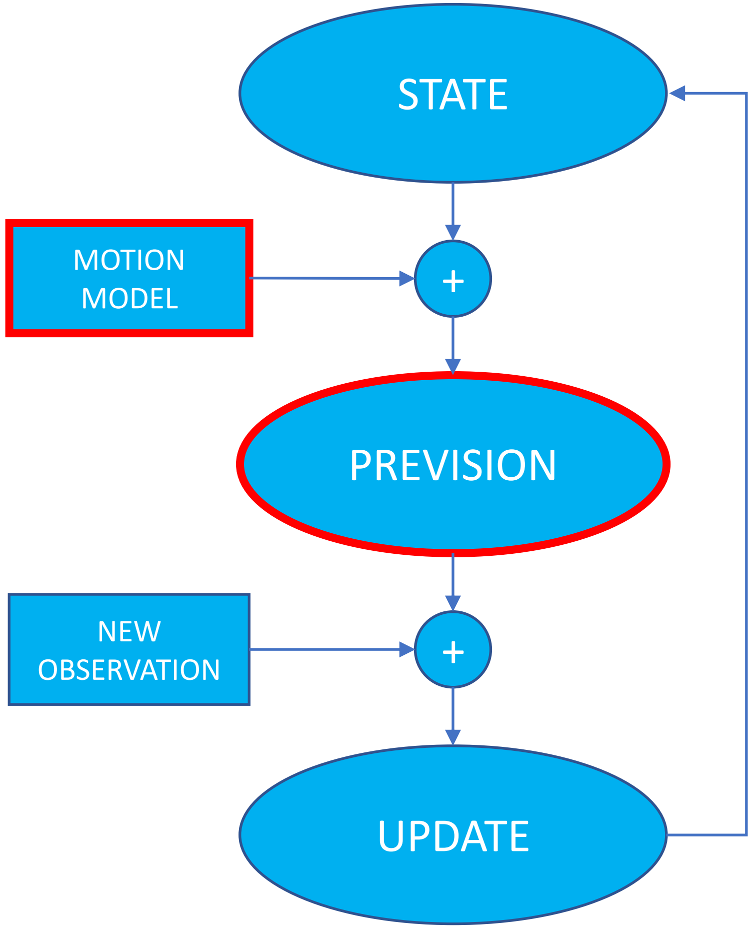


A. J. Davison, "Real-time simultaneous localization and mapping with a single camera," ICCV, 2003

Bayesian Solution

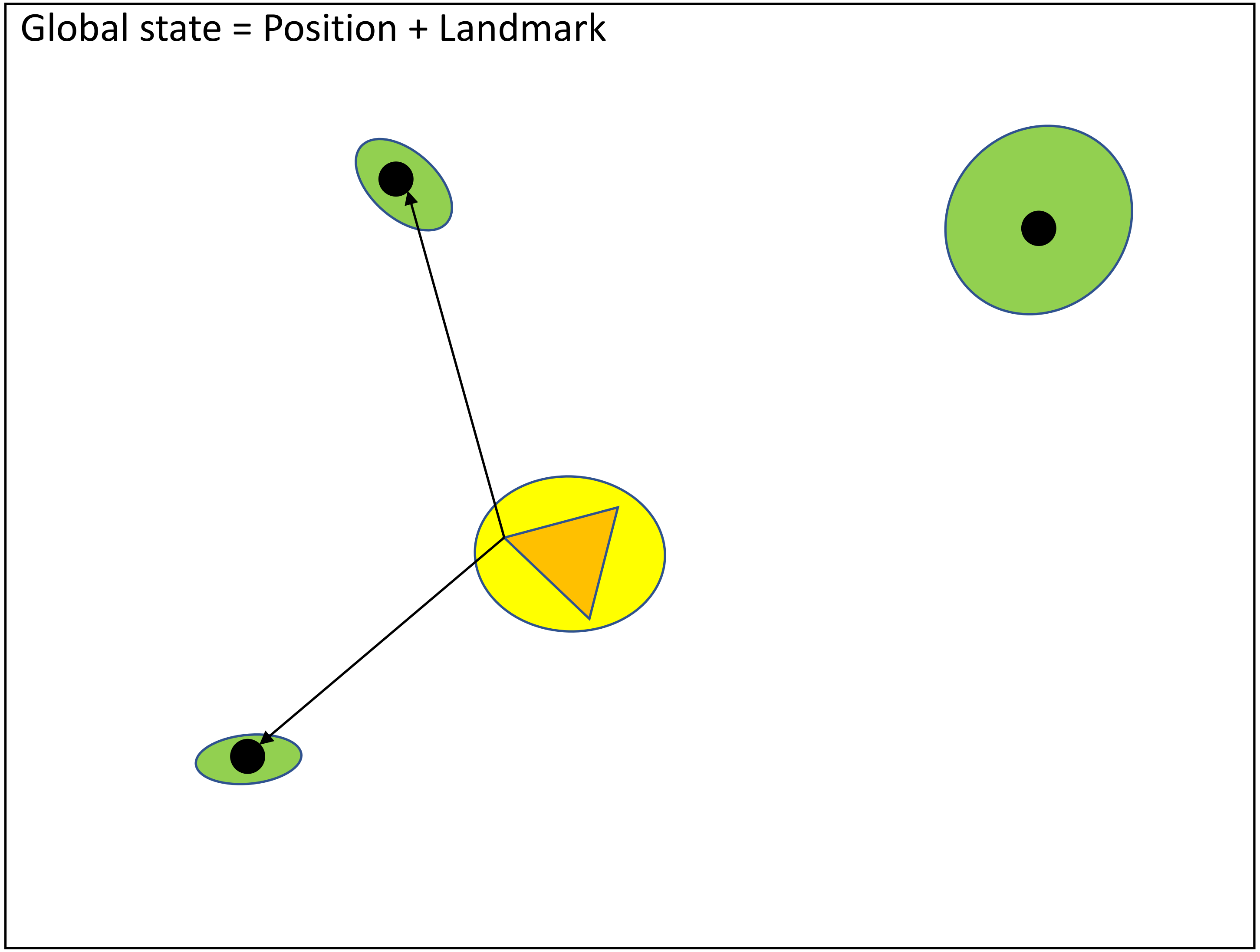
Global state = Position + Landmark

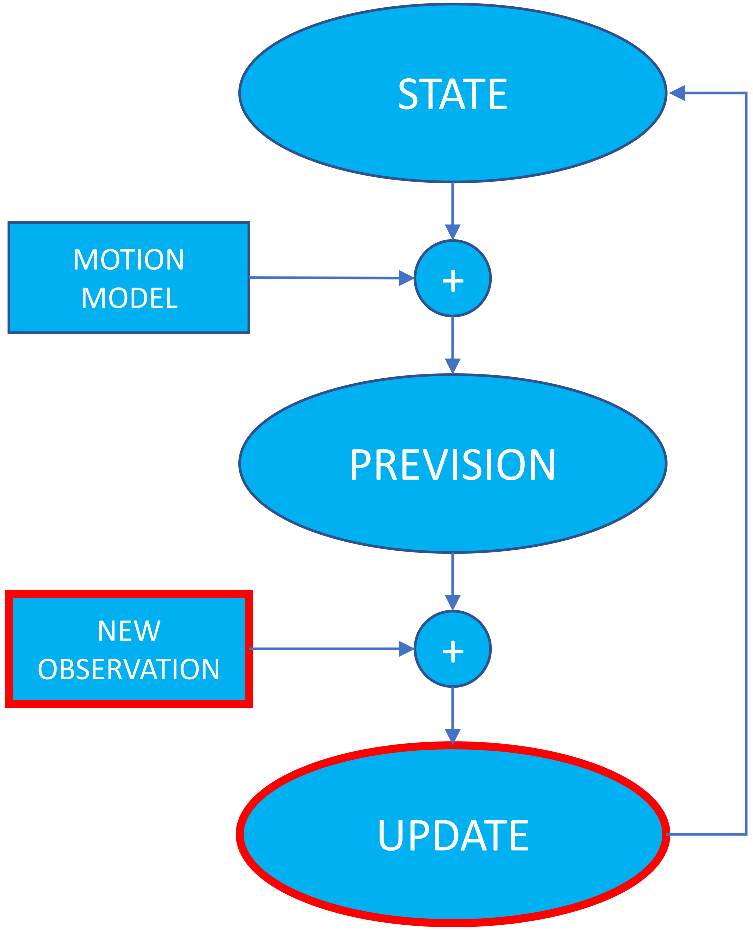




A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Bayesian Solution





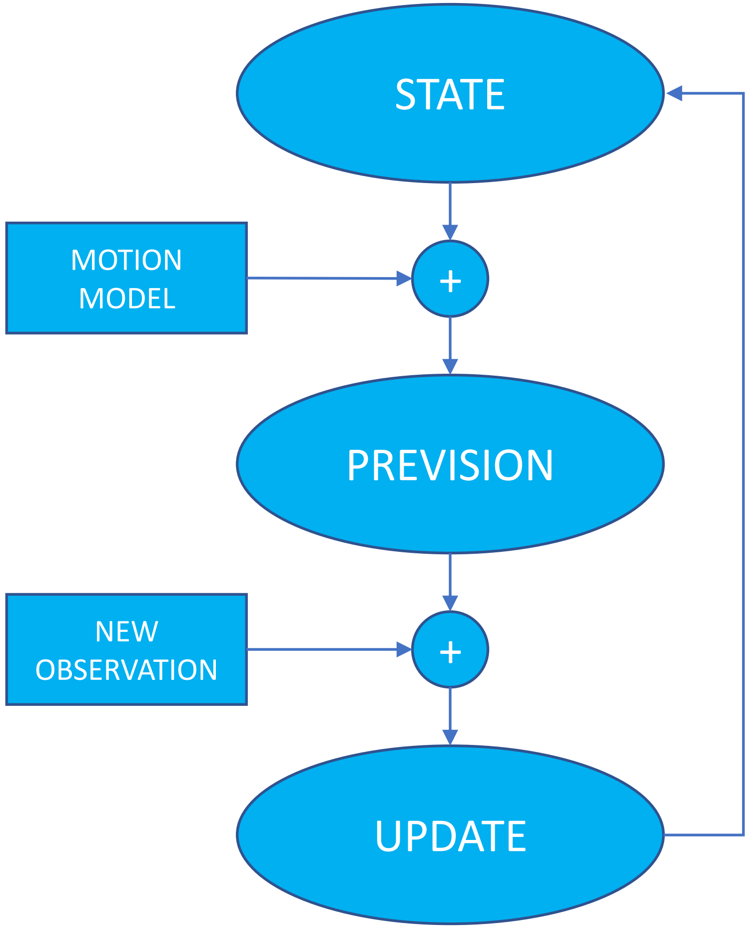
A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Bayesian Solution

- The state of the system include
  - State of the camera (camera position + motion model)

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{r} \\ \mathbf{q} \\ \mathbf{v} \\ \omega \end{bmatrix}$$

$\mathbf{r}$	position	3
$\mathbf{q}$	rotation (quaternion)	4
$\mathbf{v}$	velocity	3
ω	angular velocity	3
		13 total



A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Bayesian Solution

- The state of the system include
  - State of the camera (camera position + motion model)
  - The position of the 3D landmarks

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}$$

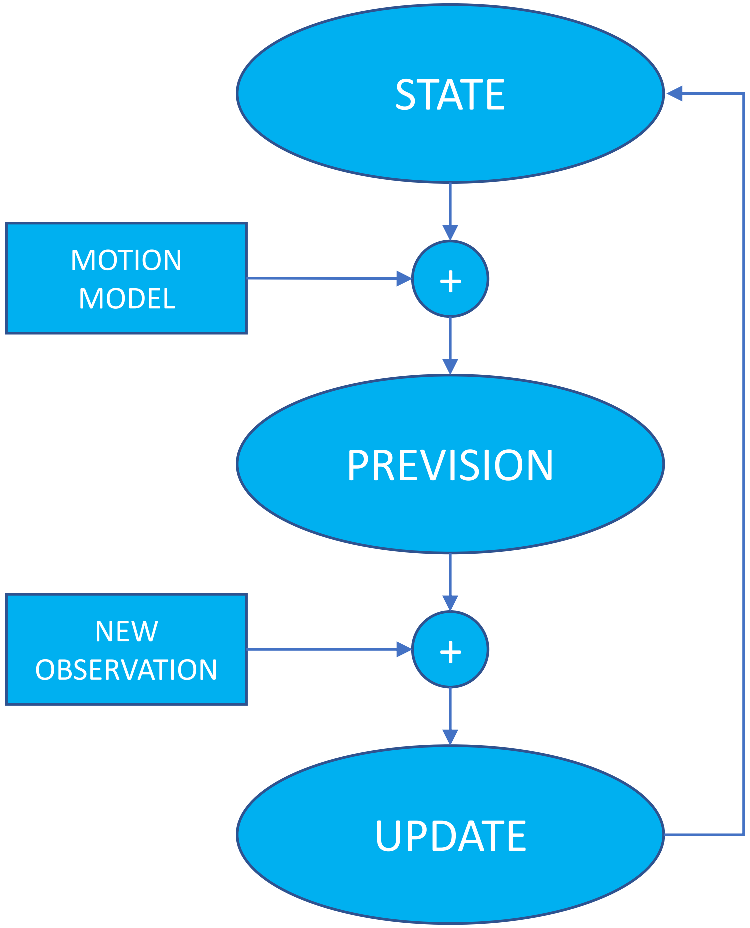
state of the camera

location of feature 1

location of feature 2

location of feature N

13+3N total



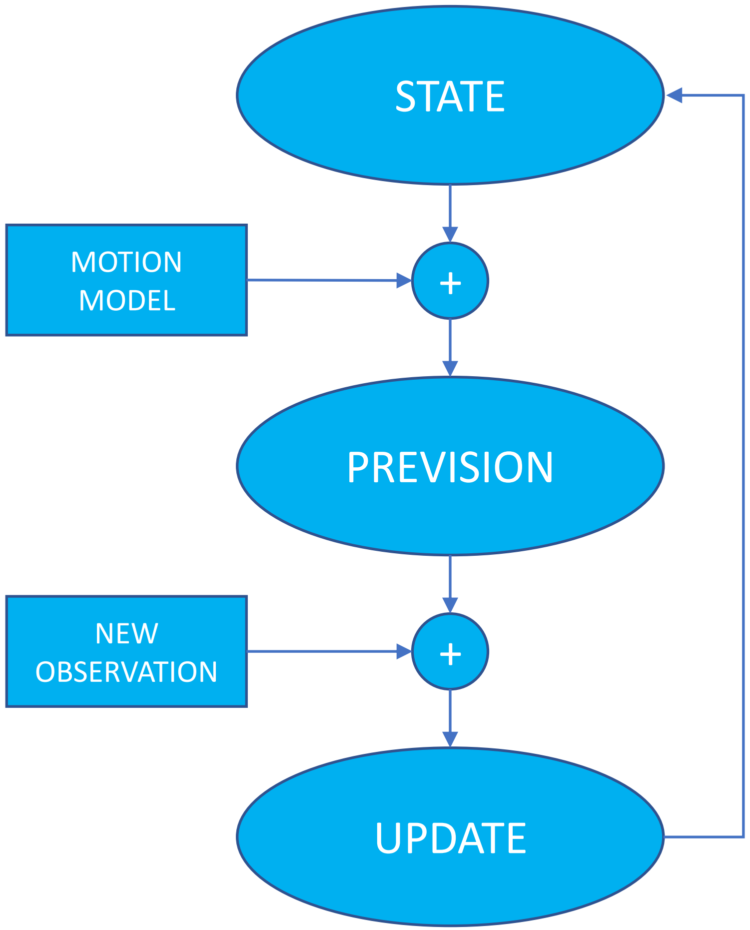
A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Bayesian Solution

- The state of the system include
  - State of the camera (camera position + motion model)
  - The position of the 3D landmarks
- Uncertainties are described with covariance matrices

$$\Sigma = \begin{bmatrix} \Sigma_{\mathbf{x}_c\mathbf{x}_c} & \Sigma_{\mathbf{x}_c\mathbf{y}_1} & \cdots & \Sigma_{\mathbf{x}_c\mathbf{y}_N} \\ \Sigma_{\mathbf{y}_1\mathbf{x}_c} & \Sigma_{\mathbf{y}_1\mathbf{y}_1} & \cdots & \Sigma_{\mathbf{y}_1\mathbf{y}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{\mathbf{y}_N\mathbf{x}_c} & \Sigma_{\mathbf{y}_N\mathbf{y}_1} & \cdots & \Sigma_{\mathbf{y}_N\mathbf{y}_N} \end{bmatrix}$$

Σ: 13+3N x 13+3N

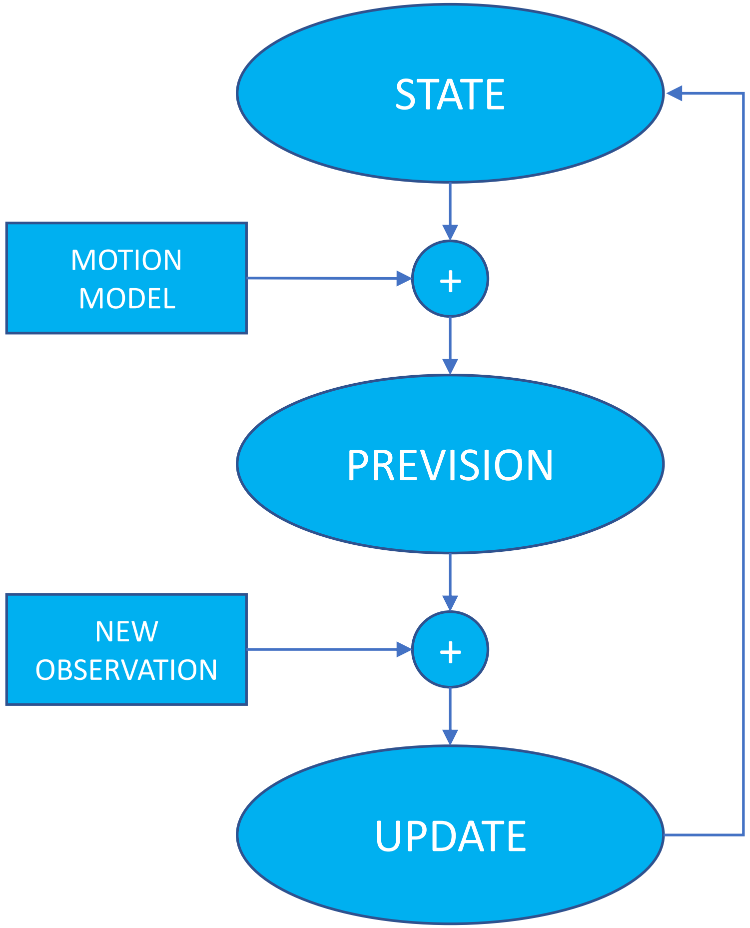


A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Bayesian Solution

- During the prediction step, the state of the camera is updated (using a constant velocity model)

$$\mathbf{f}_t = \begin{bmatrix} \mathbf{r}_t \\ \mathbf{q}_t \\ \mathbf{v}_t \\ \omega_t \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{t-1} + \mathbf{v}_{t-1}\Delta t \\ \mathbf{q}_{t-1} + \mathbf{q}(\omega)_{t-1}\Delta t \\ \mathbf{v}_{t-1} \\ \omega_{t-1} \end{bmatrix}$$



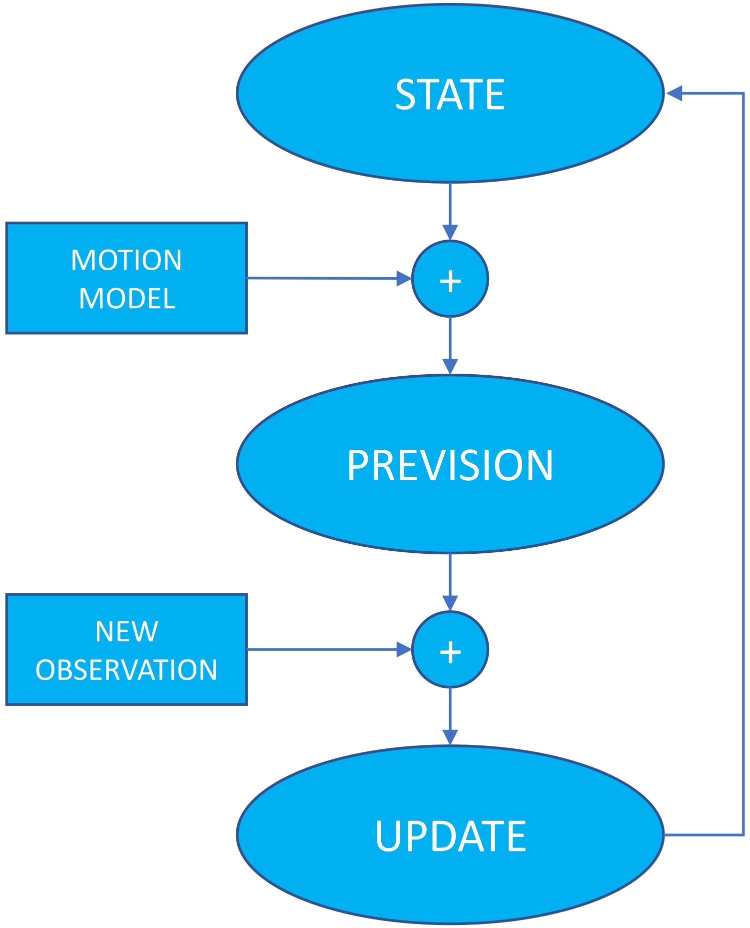
A. J. Davison, "Real-time simultaneous localization and mapping with a single camera," ICCV, 2003

Bayesian Solution

- During the prediction step, the state of the camera is updated (using a constant velocity model)
- And covariances are modified also

$$\bar{\Sigma}_{\mathbf{xx}} = \frac{\partial \mathbf{f}_t}{\partial \mathbf{x}} \Sigma_{\mathbf{xx}} \frac{\partial \mathbf{f}_t}{\partial \mathbf{x}}^\top + \mathbf{Q}_t$$

new covariance; change around new state; old covariance; change around new state; system noise (process noise)



A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Bayesian Solution

- During the prediction step, the state of the camera is updated (using a constant velocity model)
- And covariances are modified also
- Then, during the update step, both the state and the covariance are updated exploiting the new measurements

Kalman gain

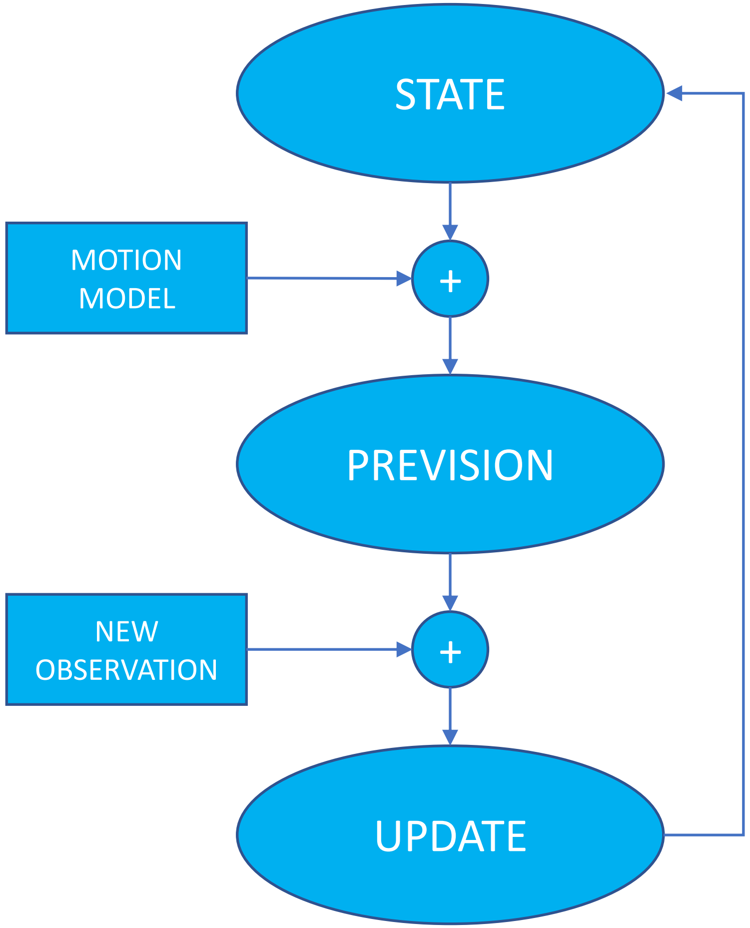
$$\mathbf{x}_t = \mathbf{x}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{h}(\mathbf{y};\mathbf{x}_t))$$

Updated state — Predicted state — Matched 2D features — 2D projection of 3D point

Identity — Kalman gain

$$\mathbf{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{\Sigma}_t$$

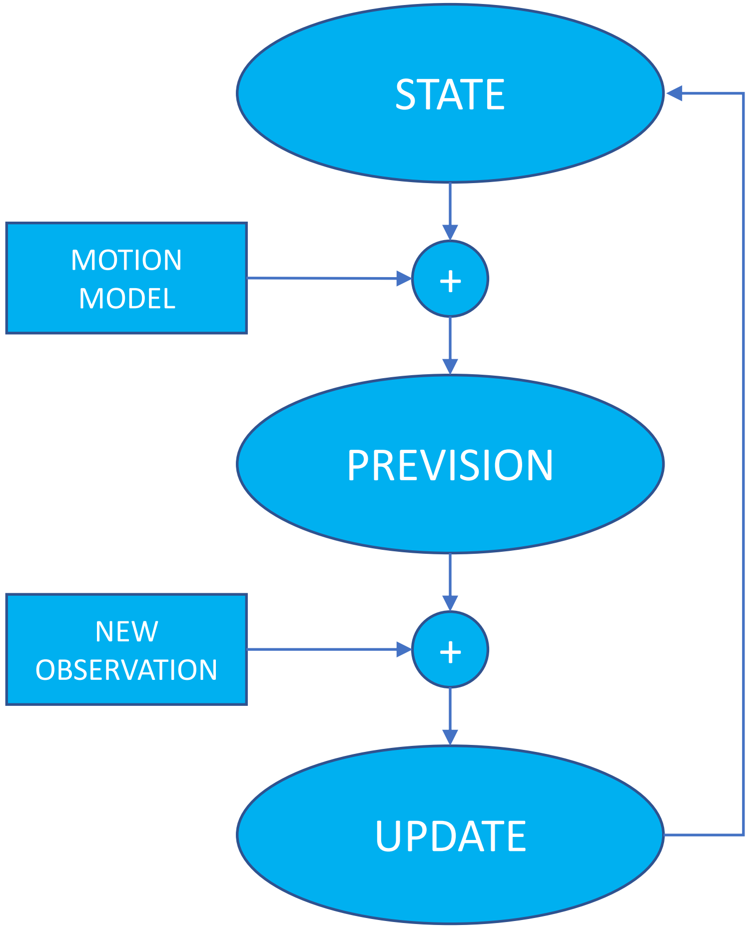
Covariance (updated) — Jacobian — Covariance (predicted)



A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Bayesian Solution

- The main problem of this kind of approaches is the dimension of the covariance matrix
- By progressively adding new landmark, the covariance matrix grows and become difficult to respect the real-time constraints
- For this reason, the number of used landmark must be kept limited, but this can lower the estimation precision of the camera pose



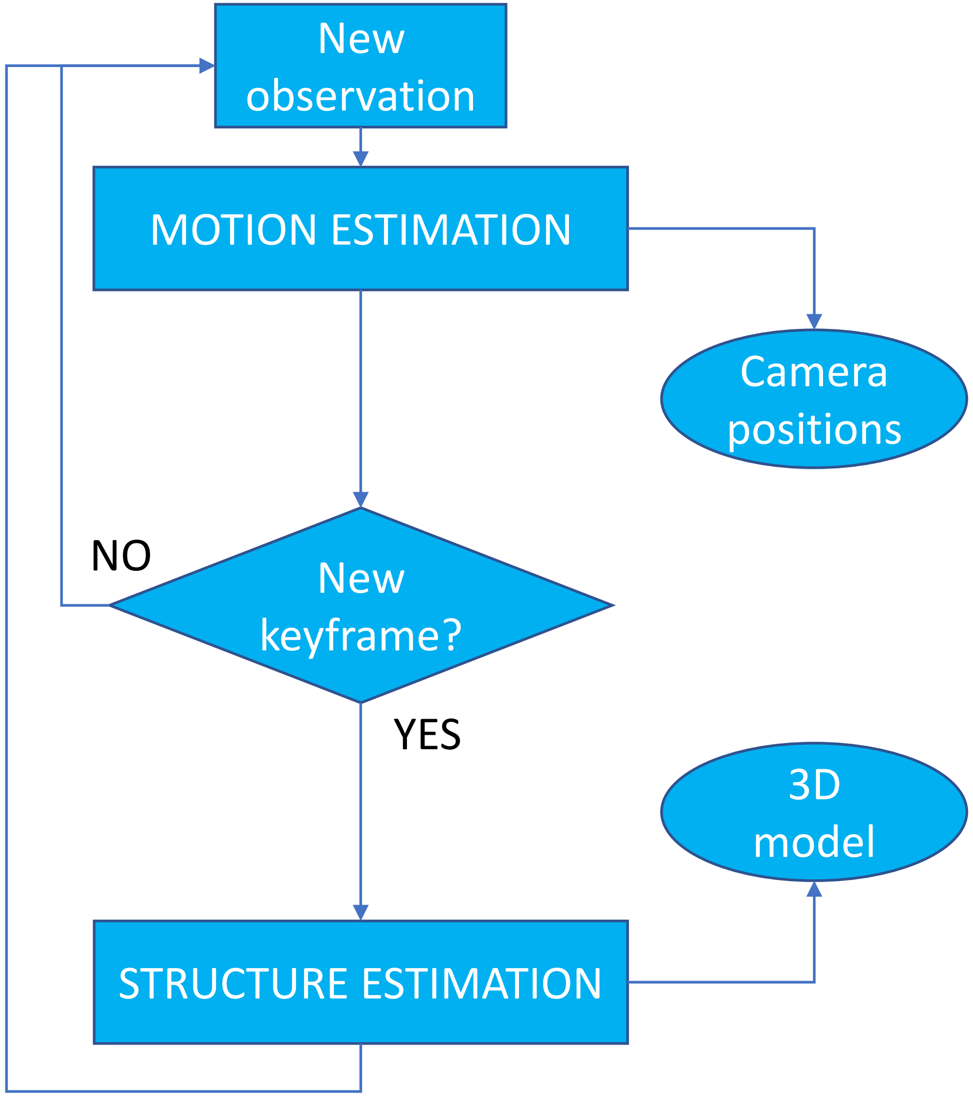
A. J. Davison, "Real-time simultaneous localization and mapping with a single camera," ICCV, 2003

Bayesian Solution

A. J. Davison, “Real-time simultaneous localization and mapping with a single camera,” ICCV, 2003

Keyframe-based Solution

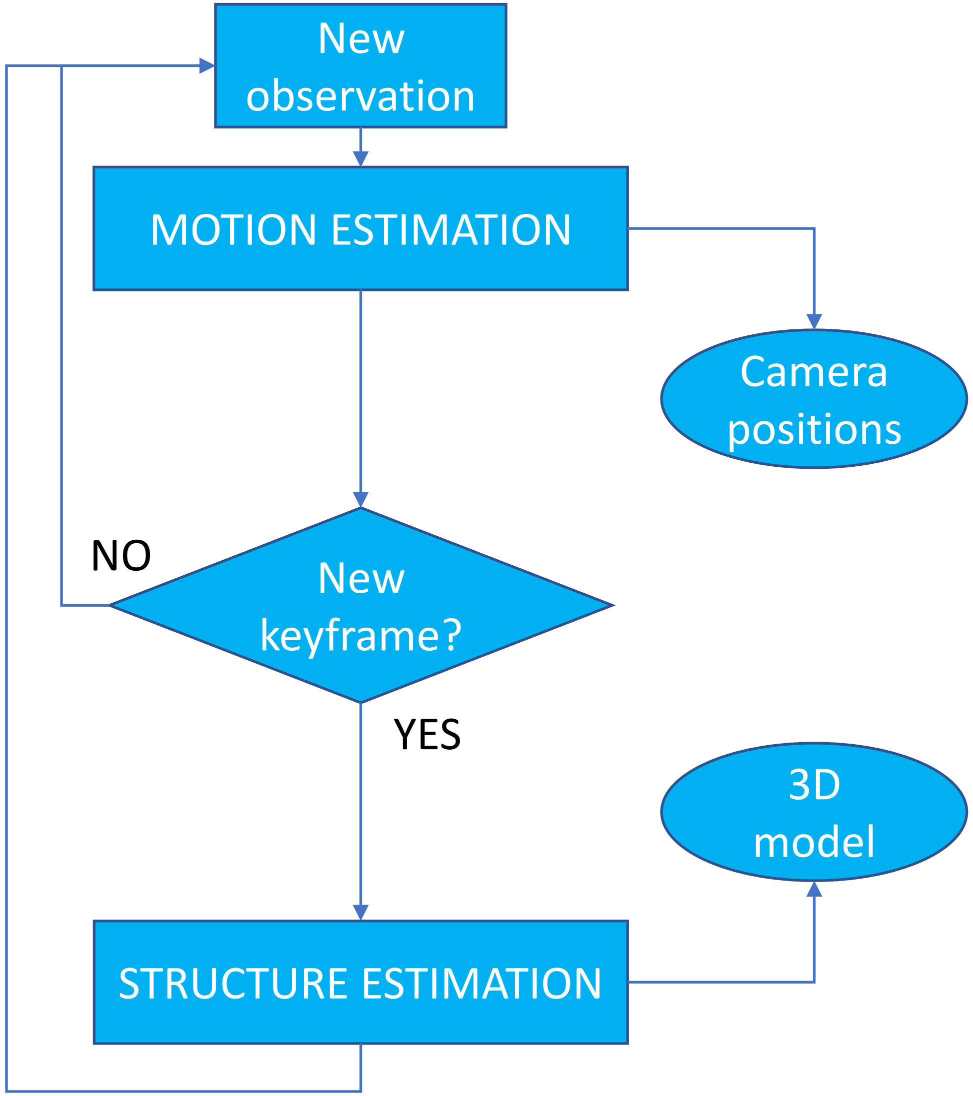
- Based on **SfM**



G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," ISMAR, 2007.

Keyframe-based Solution

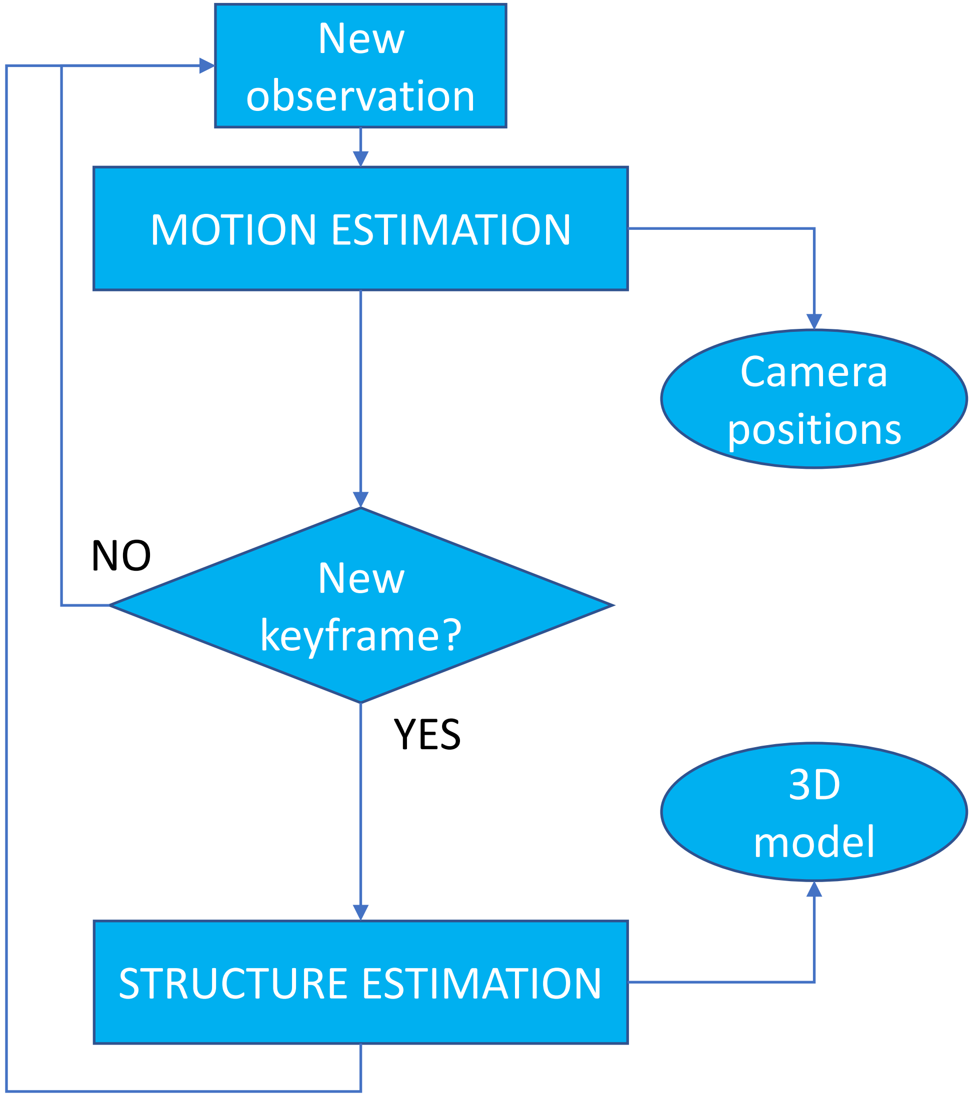
- Based on **SfM**
- Update of structure and trajectory may happen at **different time**
    - The **scene is static**, no need to update the 3D model for each new observation



G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” ISMAR, 2007.

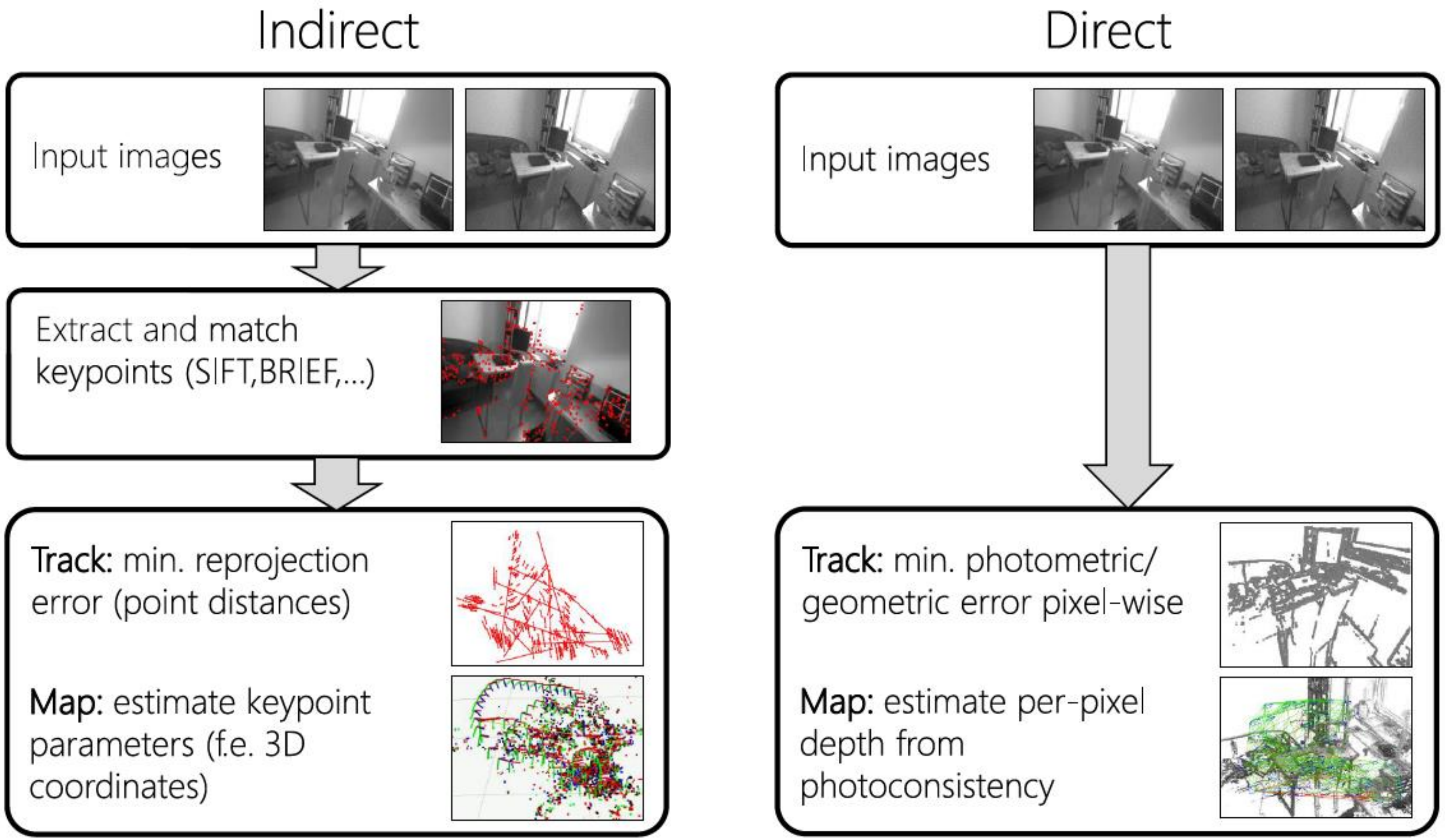
Keyframe-based Solution

- Based on **SfM**
- Update of structure and trajectory may happen at **different time**
  - The **scene is static**, no need to update the 3D model for each new observation
- **Bundle Adjustment** on 3D map and keyframe poses



G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” ISMAR, 2007.

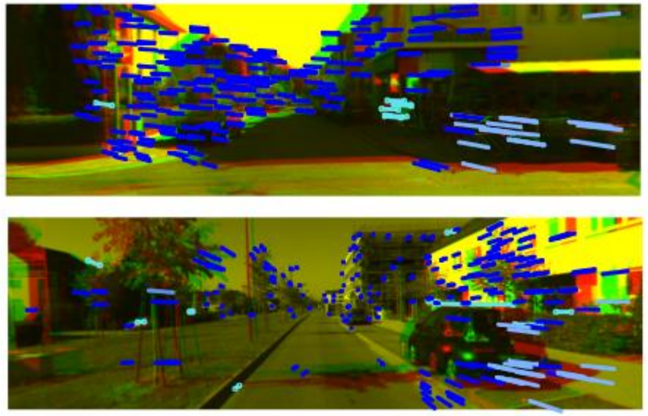
Indirect *vs* Direct Methods

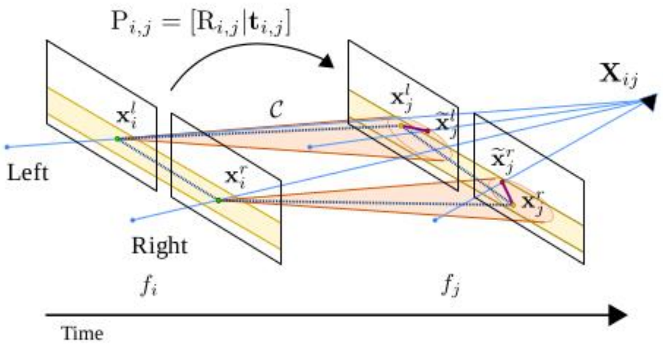


Slide Credits: Jorg Stuckler

Stereo Visual Odometry

- Input: sequence of rectified **stereo images**, with known calibration
- No global optimization (no BA)
- Keyframe selection to maintain low estimation error

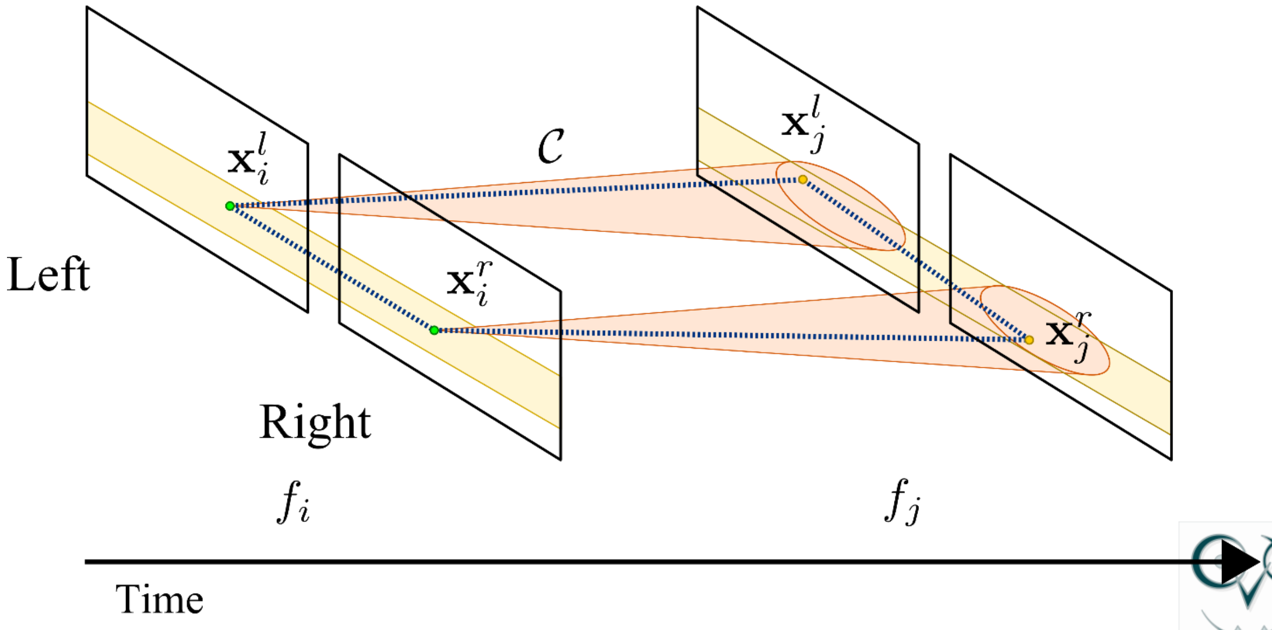




M. Fanfani, F. Bellavia, and C. Colombo, "Accurate keyframe selection and keypoint tracking for robust visual odometry". MVA, 2016
F. Bellavia, M. Fanfani, and C. Colombo, "Selective visual odometry for accurate AUV localization". Aut. Rob., 2017

Stereo Visual Odometry

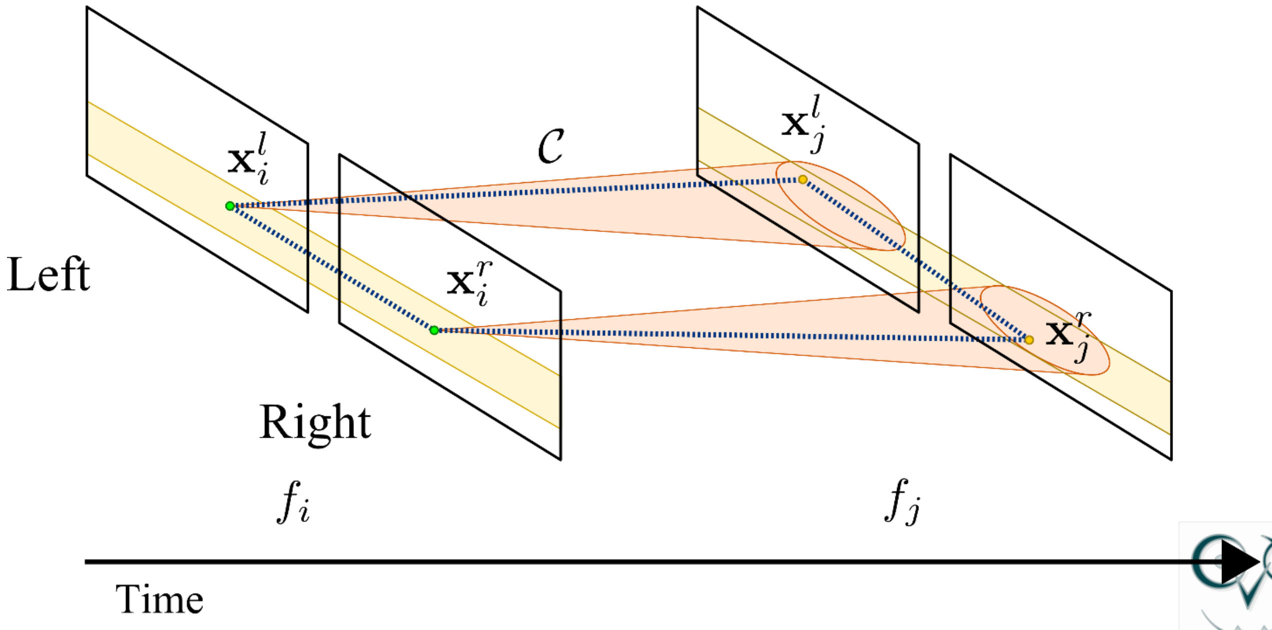
- Keypoints extracted and described with SIFT-like HarrisZ detector[1] and sGLOH descriptor[2]



[1] Bellavia et al., "Improving Harris corner selection strategy", IET Computer Vision, 2011
[2] Bellavia et al., "Improving SIFT-based descriptors stability to rotations", ICPR , 2010

Stereo Visual Odometry

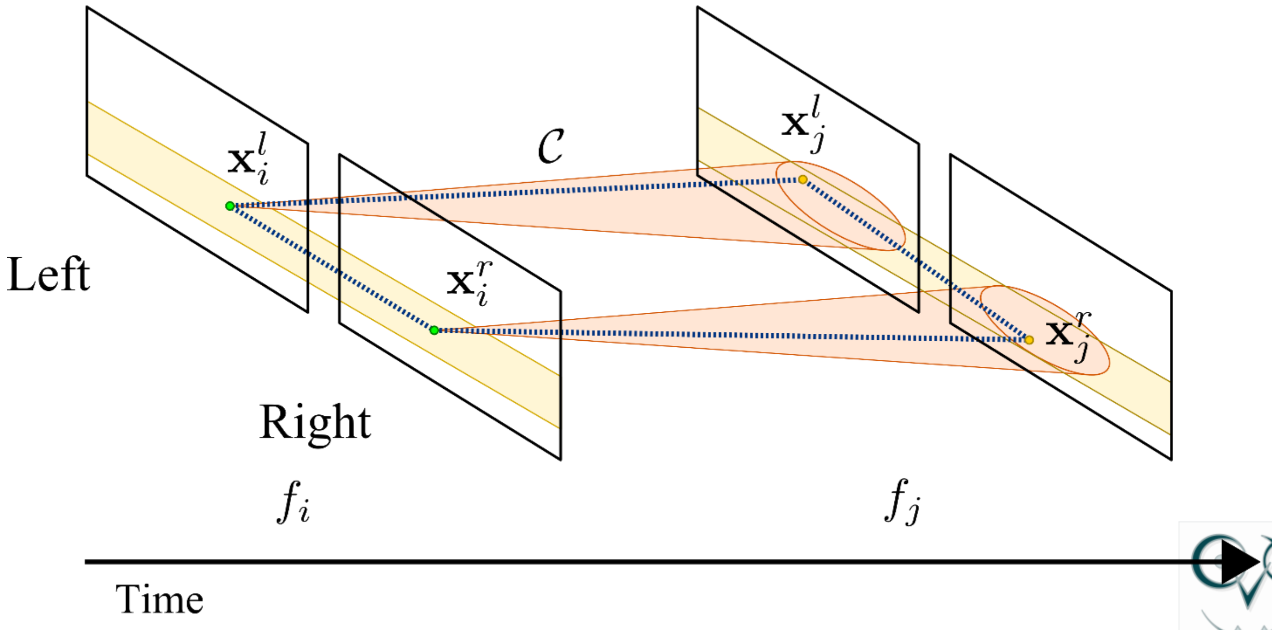
- Keypoints extracted and described with SIFT-like HarrisZ detector[1] and sGLOH descriptor[2]
- **Stereo matching** constrained by epipolar line



[1] Bellavia et al., “Improving Harris corner selection strategy”, IET Computer Vision, 2011
[2] Bellavia et al., “Improving SIFT-based descriptors stability to rotations”, ICPR , 2010

Stereo Visual Odometry

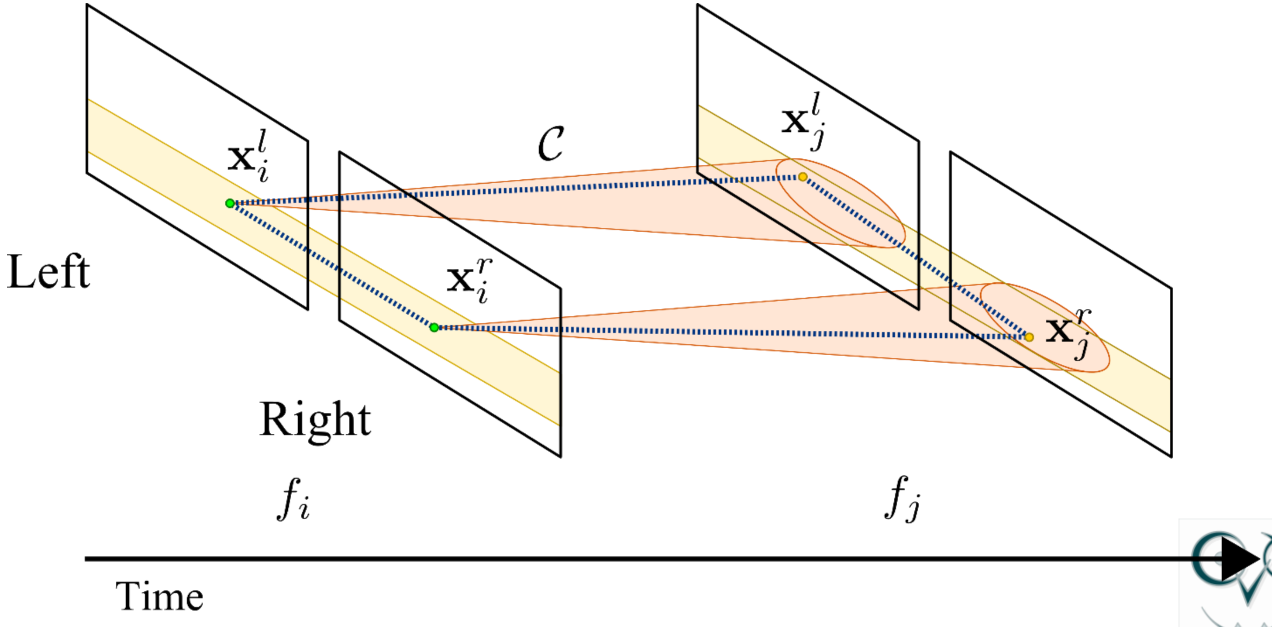
- Keypoints extracted and described with SIFT-like HarrisZ detector[1] and sGLOH descriptor[2]
- **Stereo matching** constrained by epipolar line
- **Temporal matching** constrained by flow motion restriction



[1] Bellavia et al., "Improving Harris corner selection strategy", IET Computer Vision, 2011
[2] Bellavia et al., "Improving SIFT-based descriptors stability to rotations", ICPR , 2010

Stereo Visual Odometry

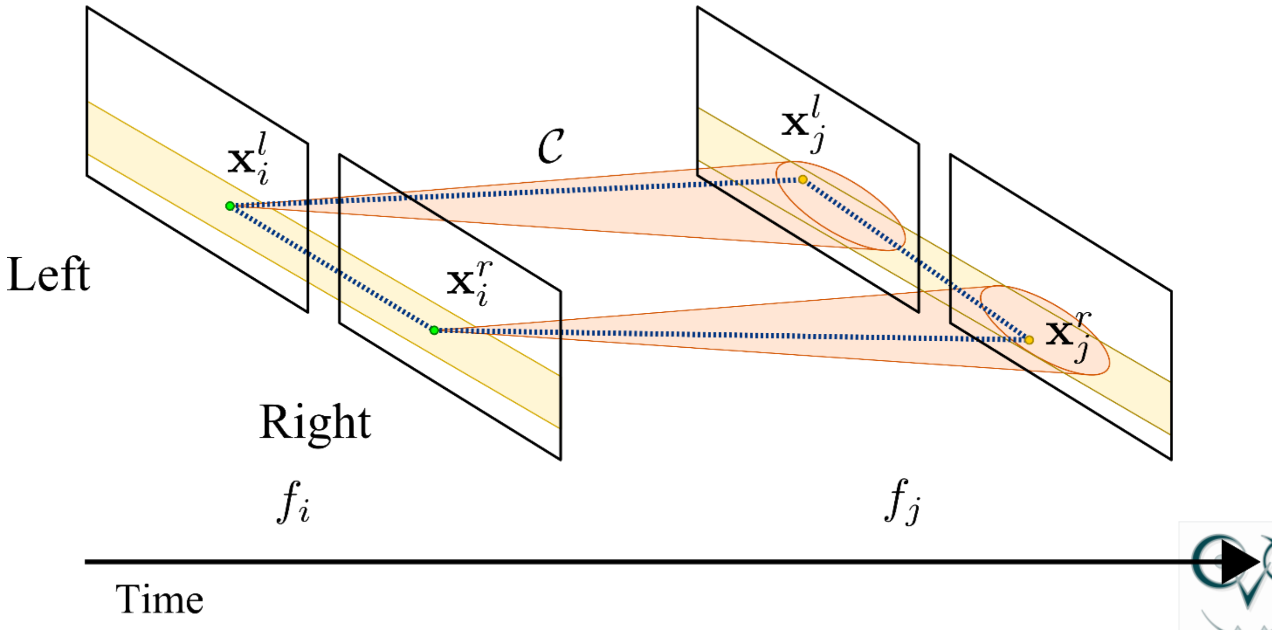
- Keypoints extracted and described with SIFT-like HarrisZ detector[1] and sGLOH descriptor[2]
- **Stereo matching** constrained by epipolar line
- **Temporal matching** constrained by flow motion restriction
- Matching loop chain construction



[1] Bellavia et al., “Improving Harris corner selection strategy”, IET Computer Vision, 2011
[2] Bellavia et al., “Improving SIFT-based descriptors stability to rotations”, ICPR , 2010

Stereo Visual Odometry

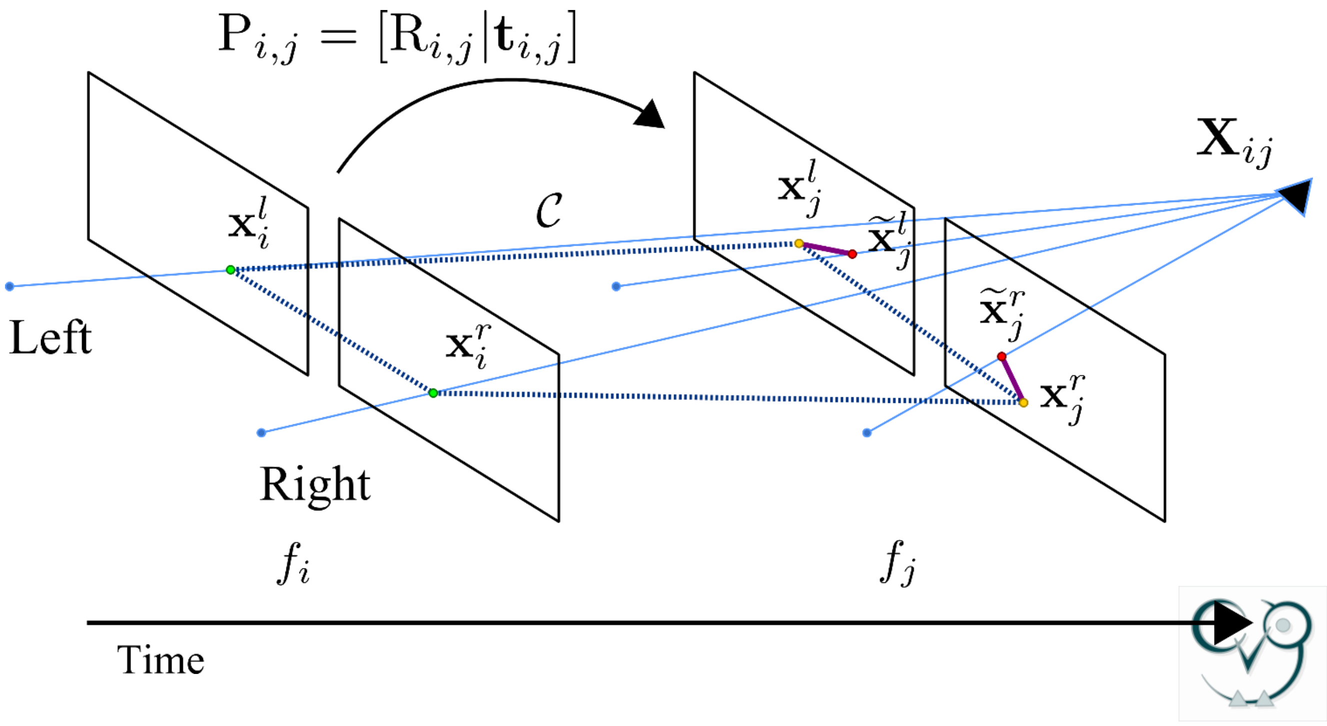
- Keypoints extracted and described with SIFT-like HarrisZ detector[1] and sGLOH descriptor[2]
- **Stereo matching** constrained by epipolar line
- **Temporal matching** constrained by flow motion restriction
- Matching loop chain construction
- Outlier removal with four distinct RANSAC



[1] Bellavia et al., “Improving Harris corner selection strategy”, IET Computer Vision, 2011
[2] Bellavia et al., “Improving SIFT-based descriptors stability to rotations”, ICPR , 2010

Stereo Visual Odometry

- Keypoints $\{\mathbf{x}_i^l, \mathbf{x}_i^r\}$ of previous keyframe-pair are put in 3D by triangulation, obtaining a local map $\{\mathbf{X}_{ij}\}$

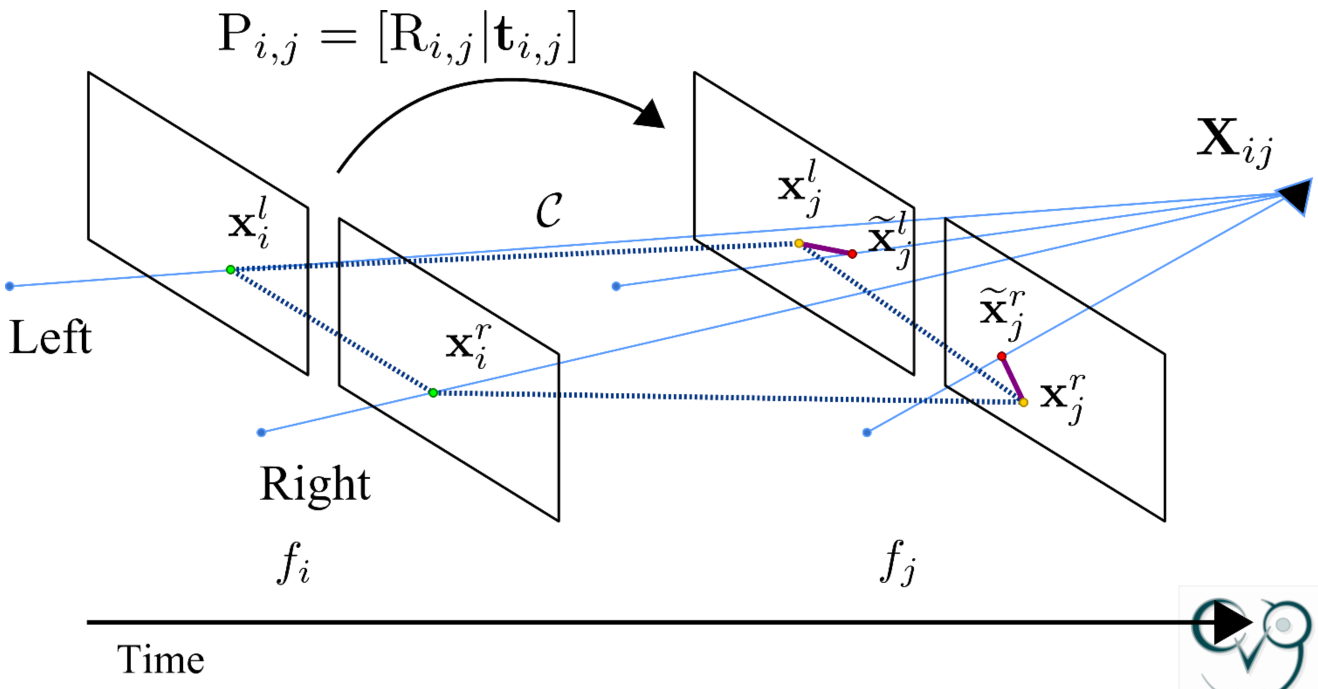


Stereo Visual Odometry

- Keypoints $\{\mathbf{x}_i^l, \mathbf{x}_i^r\}$ of previous keyframe-pair are put in 3D by triangulation, obtaining a local map $\{\mathbf{X}_{ij}\}$
- The incremental roto-translation between the previous (keyframe) and the current pose is estimated minimizing

$$\sum ||\tilde{\mathbf{x}}_j^l - \mathbf{x}_j^l||^2 + ||\tilde{\mathbf{x}}_j^r - \mathbf{x}_j^r||^2$$

where $\tilde{\mathbf{x}}_j^* = K[R\mathbf{X}_{ij} + \mathbf{t}]$



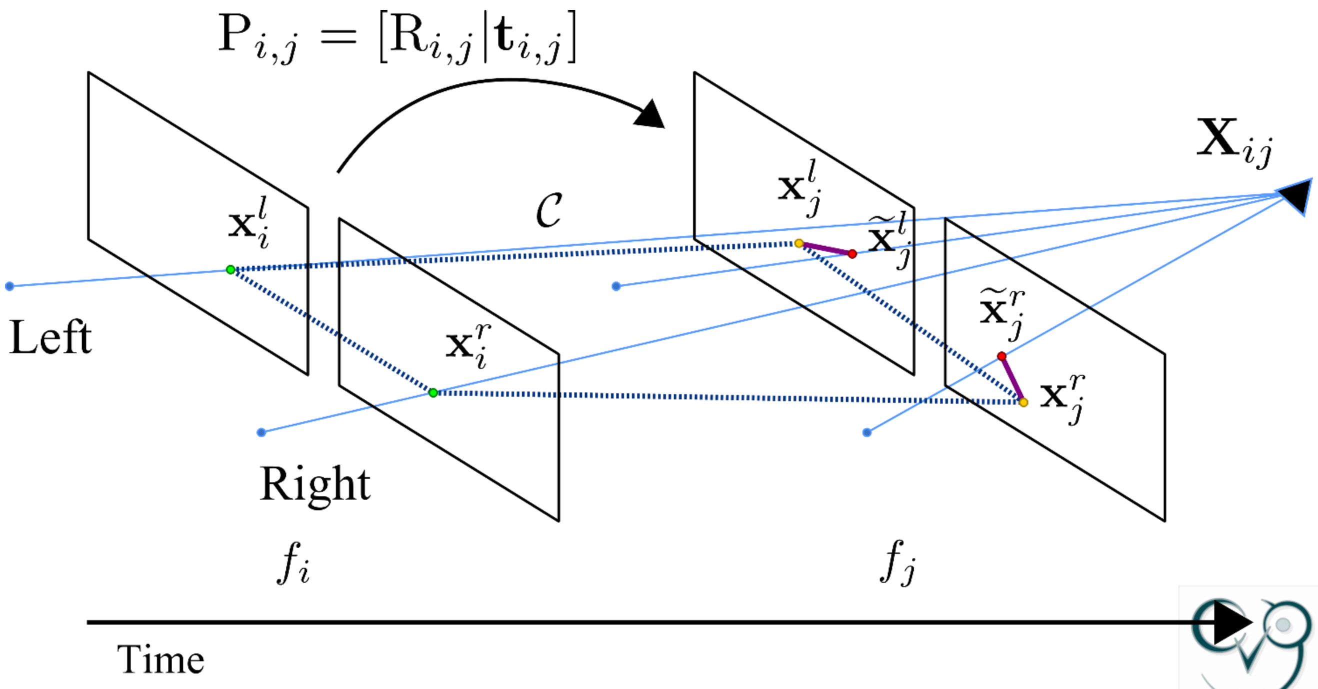
Stereo Visual Odometry

- Keypoints $\{\mathbf{x}_i^l, \mathbf{x}_i^r\}$ of previous keyframe-pair are put in 3D by triangulation, obtaining a local map $\{\mathbf{X}_{ij}\}$
- The incremental roto-translation between the previous (keyframe) and the current pose is estimated minimizing

$$\sum ||\tilde{\mathbf{x}}_j^l - \mathbf{x}_j^l||^2 + ||\tilde{\mathbf{x}}_j^r - \mathbf{x}_j^r||^2$$

where $\tilde{\mathbf{x}}_j^* = K[R\mathbf{X}_{ij} + \mathbf{t}]$

- Estimation is carried out in a RANSAC framework
- Global poses are obtained by concatenation



Stereo Visual Odometry

- The incremental roto-translations are computed for each frame w.r.t. the last keyframe

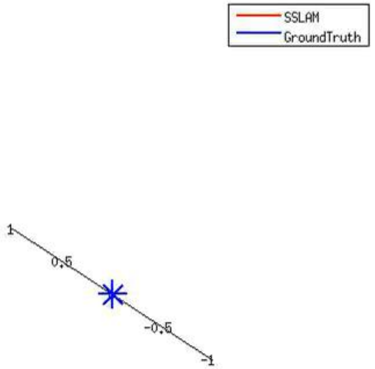
Stereo Visual Odometry

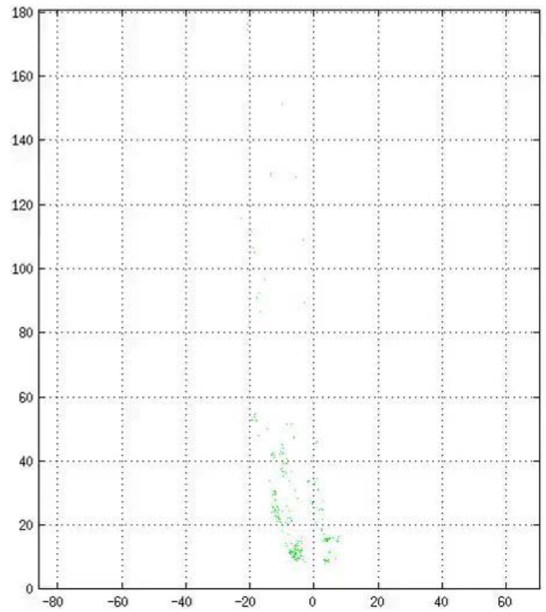
- The incremental roto-translations are computed for each frame w.r.t. the last keyframe
- A frame is selected as keyframe if an appreciable motion is found. Given the match-chain $C_{i,j}$, the fixed points are selected

$$F_{i,j} = \{c \in C_{i,j} : ||\mathbf{x}_i^* - \mathbf{x}_j^*|| \leq \delta_f\}$$

If $\frac{|F_{i,j}|}{|C_{i,j}|} < \delta_m$, a new keyframe is selected.

Stereo Visual Odometry

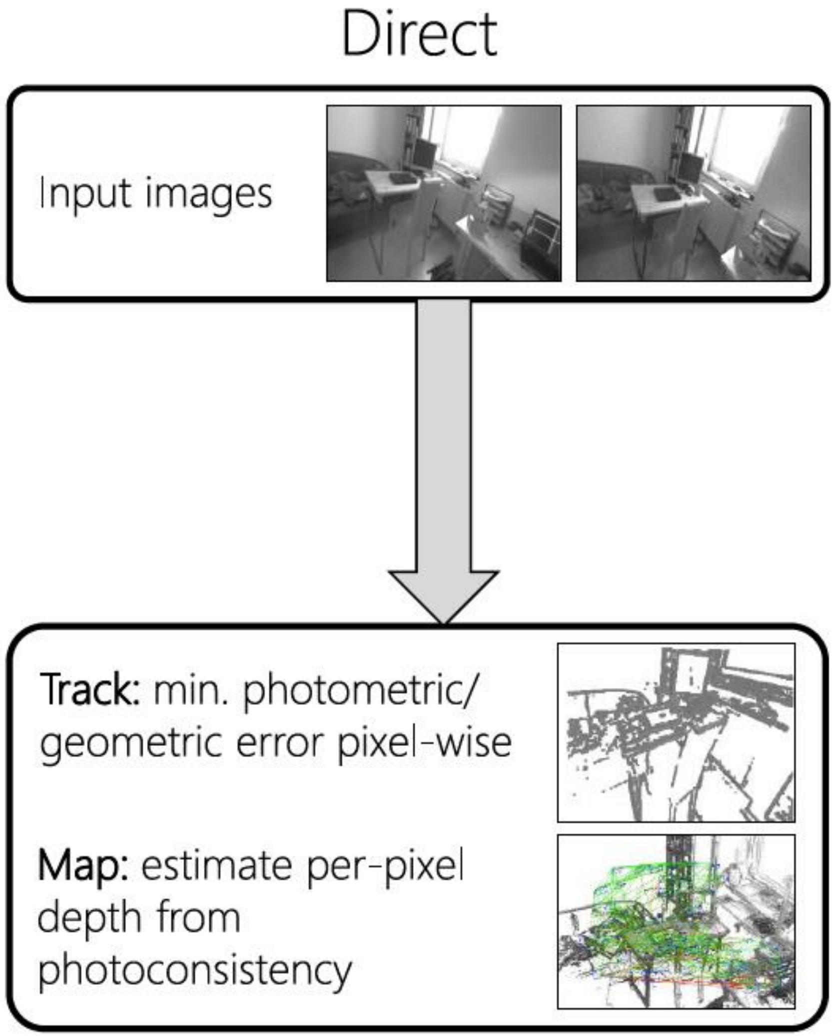




http://cvg.dsi.unifi.it/SSLAM_KITTI.mp4

Direct methods

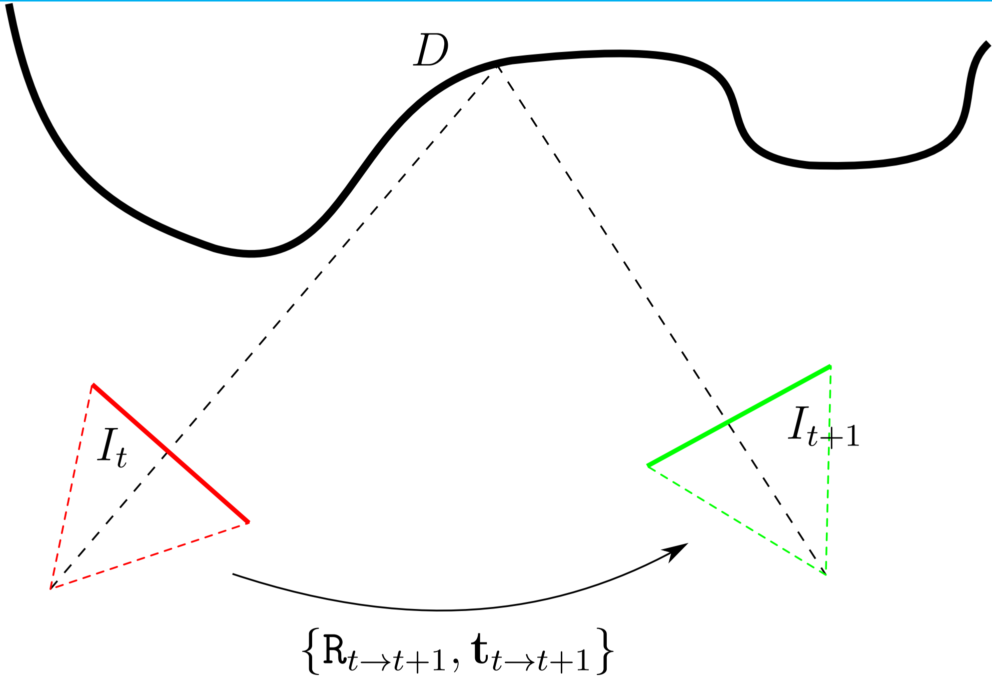
- Direct methods skip the keypoint detection/matching step
- Subsequent images are **densely put in correspondences**
- PRO: higher precision due to the greater number of matches
- CON: computationally more demanding



Direct methods

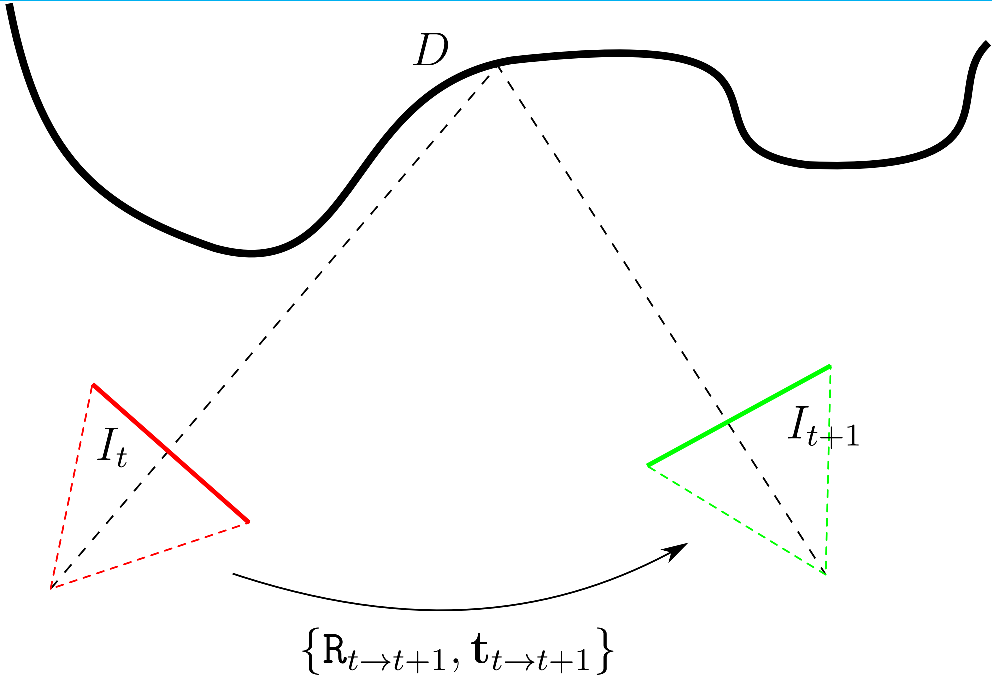
- Based on the idea of **image resynthesis**, i.e., obtain a new image from a different point of view by knowing the scene structure

- Given two subsequent images I_t and I_{t+1}



$$R^*_{t\to t+1}, \mathbf{t}^*_{t\to t+1}, D^* = \operatorname{argmin}_{R_{t\to t+1}, \mathbf{t}_{t\to t+1}, D} \sum_{\mathbf{x}\in I_t} \left\| I_t(\mathbf{x}) - I_{t+1}\big(\pi(\mathbf{x}; R_{t\to t+1}, \mathbf{t}_{t\to t+1}, D)\big) \right\|^2$$

Direct methods



- Based on the idea of **image resynthesis**, i.e., obtain a new image from a different point of view by knowing the scene structure

- Given two subsequent images I_t and I_{t+1}

$$R^*_{t\to t+1}, \mathbf{t}^*_{t\to t+1}, D^* = \text{argmin}_{R_{t\to t+1}, \mathbf{t}_{t\to t+1}, D} \sum_{\mathbf{x}\in I_t} \left\| I_t(\mathbf{x}) - I_{t+1}\big(\pi(\mathbf{x}; R_{t\to t+1}, \mathbf{t}_{t\to t+1}, D)\big) \right\|^2$$

- Easier if we know D, or some approximation, e.g., using **RGBD cameras**

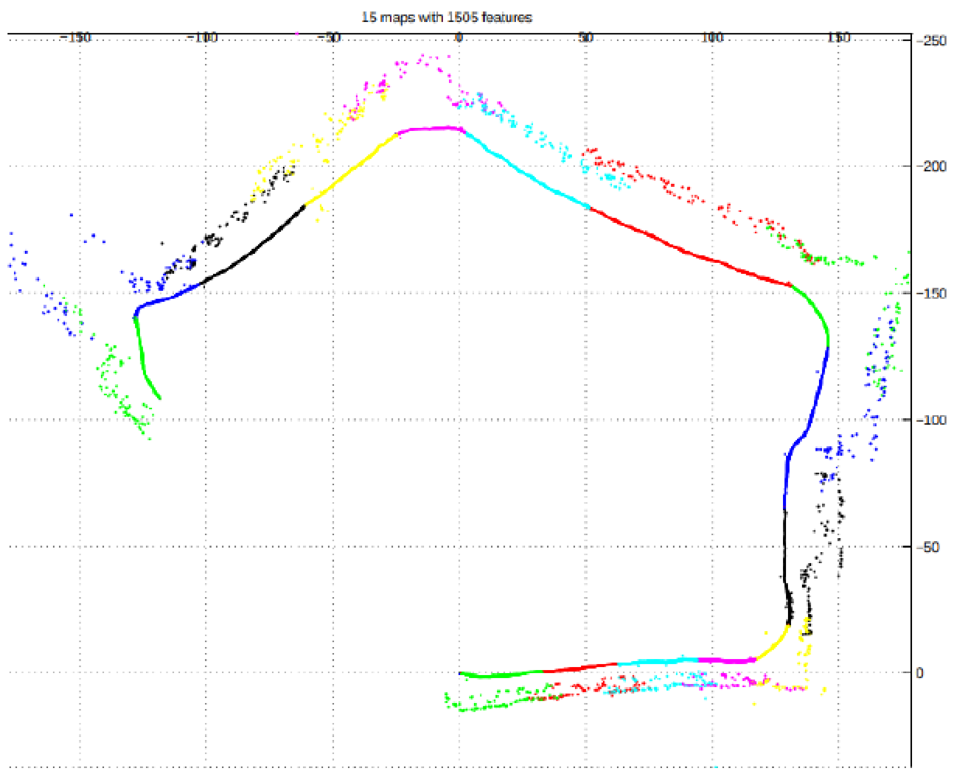
Direct methods

- There are some problem when dealing with
  - Occlusions
  - No-texture/saturated areas
  - Reflections
  - Changes in illumination

- Direct method can be used together with indirect feature-based methods

J. Engel et al., “Direct Sparse Odometry”, PAMI, 2018.

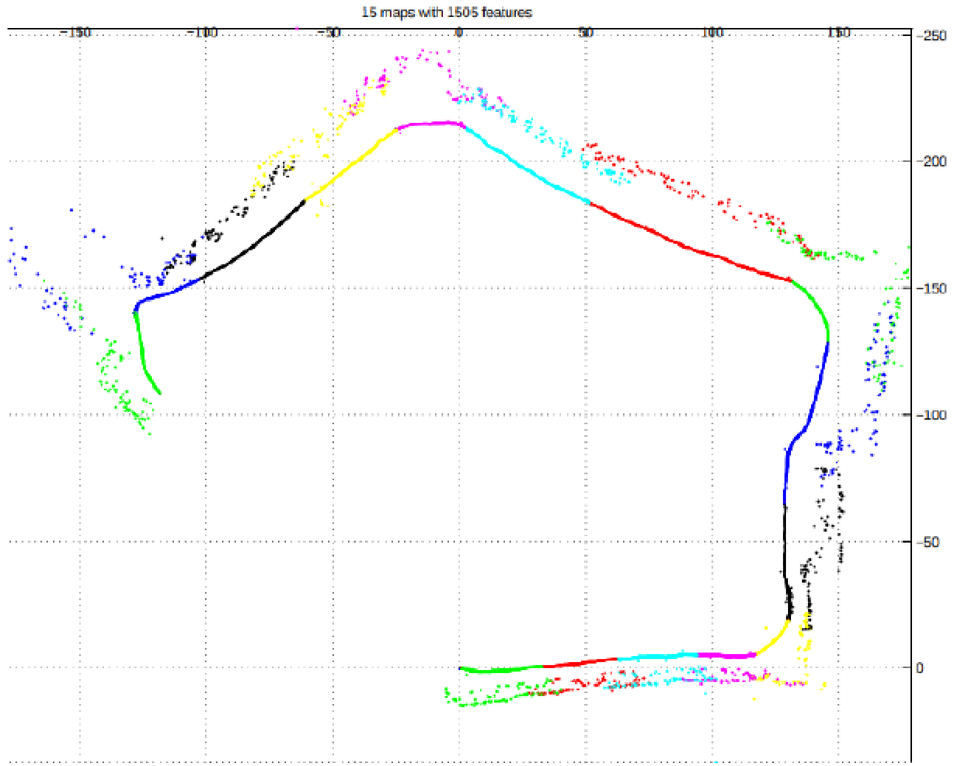
Loop closure

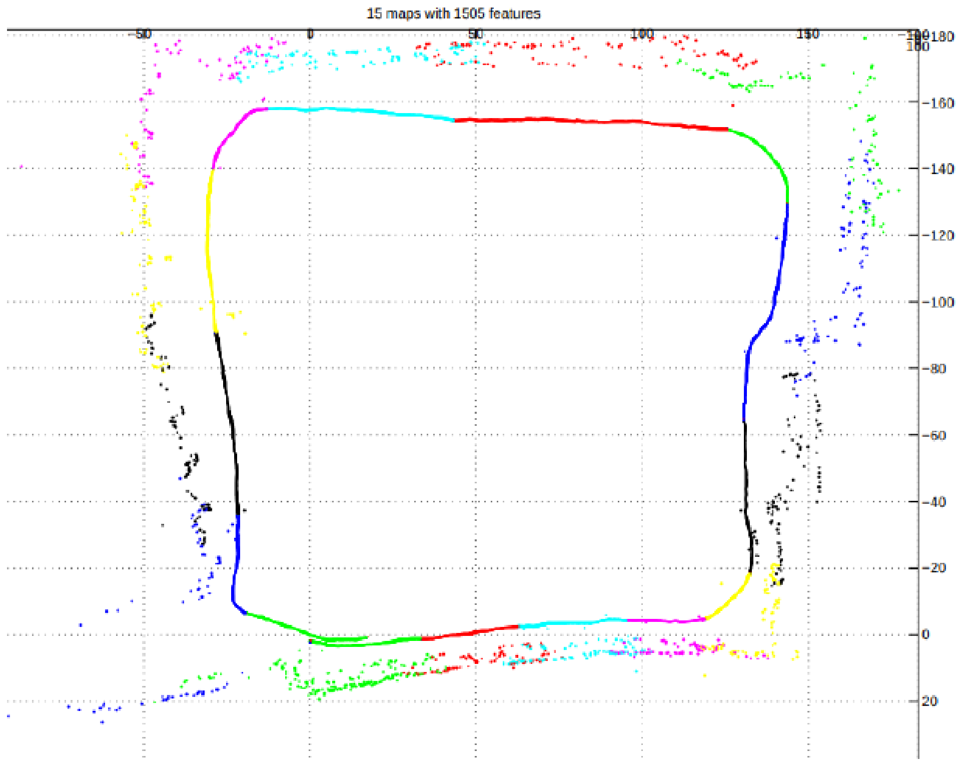


- **Problem**: by incrementally estimates poses and 3D, the error increases with time
- This leads to an **ever-increasing divergence** between the estimated and real trajectory

Clemente, et al., “Mapping Large Loops with a Single Hand-Held Camera”, RSS, 2007.

Loop closure

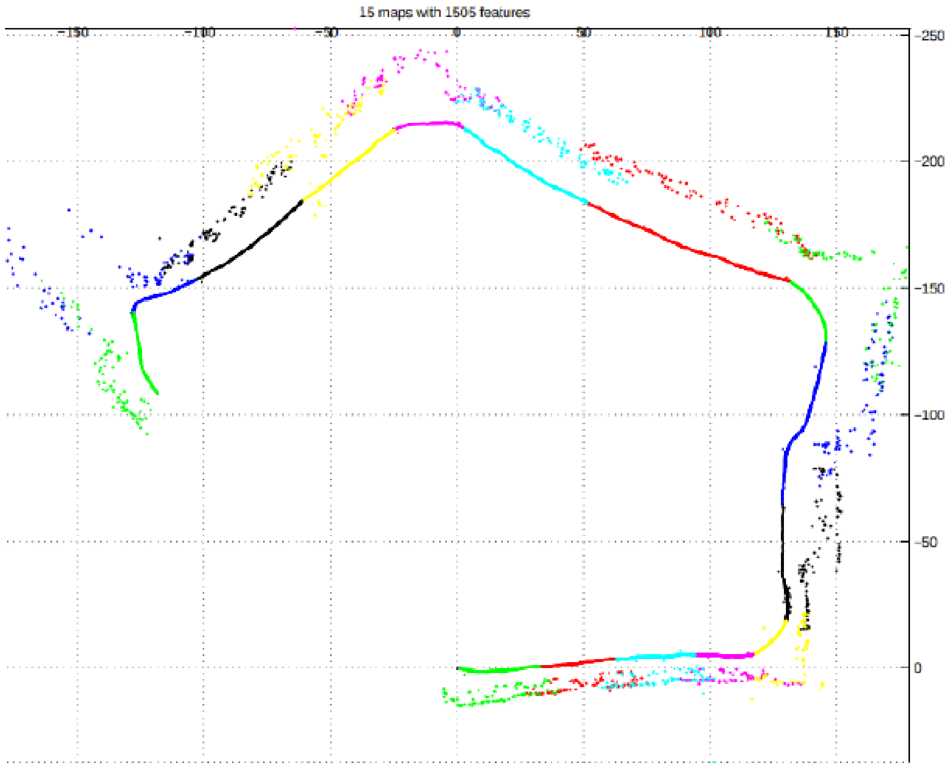


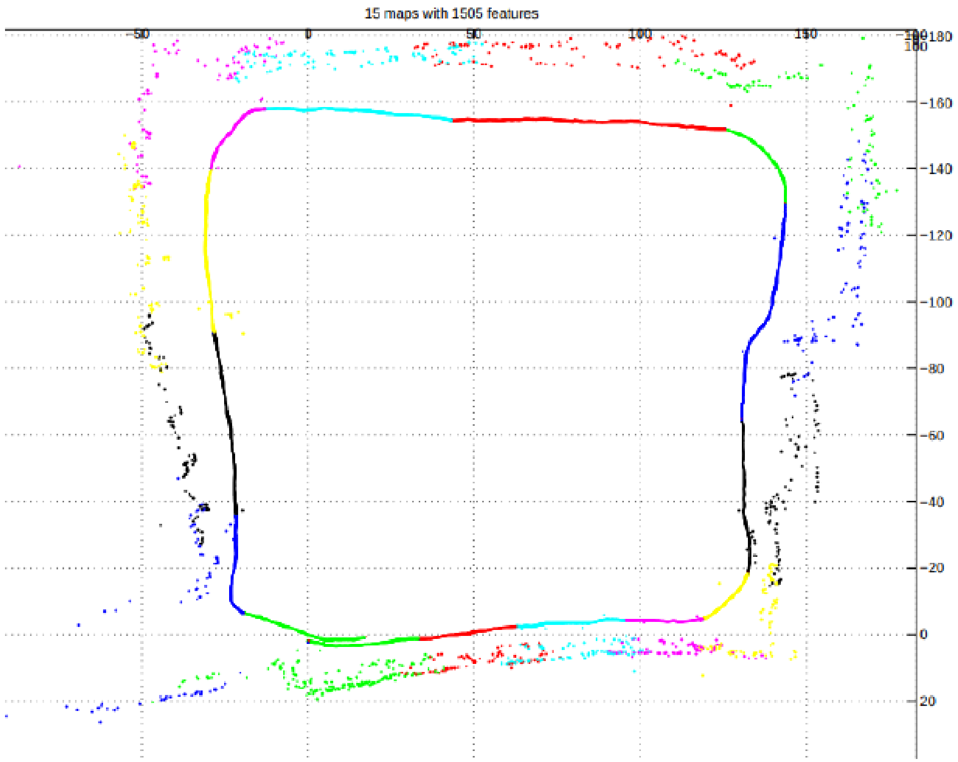


- When revisiting already seen scenes, **additional constraints** can be put into place
- It require to find correspondences between the new image and previously viewed ones

Clemente, et al., “Mapping Large Loops with a Single Hand-Held Camera”, RSS, 2007.

Loop closure

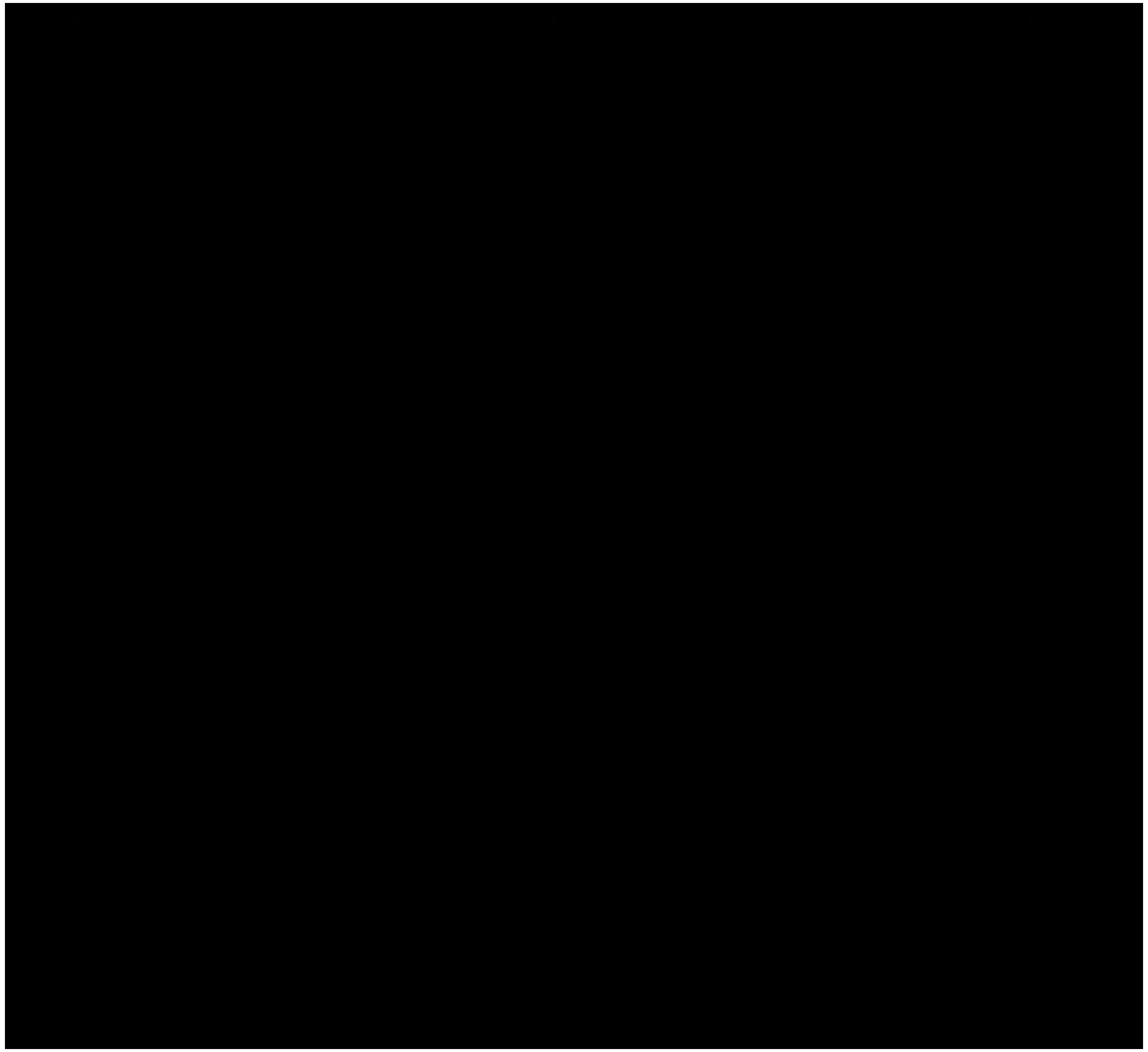




- It requires two main step
  1. Recognize already seen scenes
  2. Optimize the camera poses and the 3D map with the new constraints (using for example the bundle adjustment)

Clemente, et al., “Mapping Large Loops with a Single Hand-Held Camera”, RSS, 2007.

Loop closure



https://www.youtube.com/watch?v=O4xRdzUYAQs

ORB-SLAM

- Ready to use indirect SLAM implementation
- Integrated also in the ROS framework
- Works with **mono, stereo, and RGBD** cameras
- Implement **loop-closure** detection and global consistency with **bundle adjustment**

- Recently, **ORB-SLAM 3** was made available (https://github.com/UZ-SLAMLab/ORB_SLAM3)

README.md

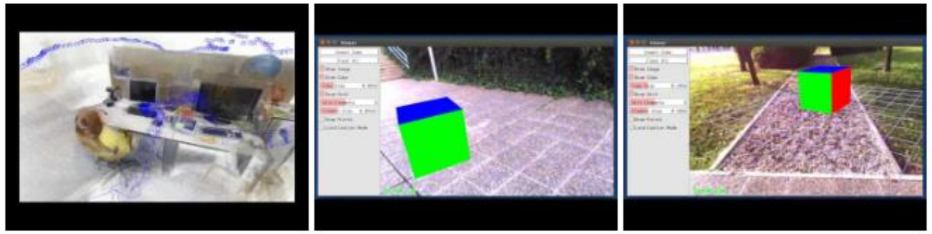
ORB-SLAM2

Authors: Raul Mur-Artal, Juan D. Tardos, J. M. M. Montiel and Dorian Galvez-Lopez (DBoW2)

13 Jan 2017: OpenCV 3 and Eigen 3.3 are now supported.

22 Dec 2016: Added AR demo (see section 7).

ORB-SLAM2 is a real-time SLAM library for **Monocular**, **Stereo** and **RGB-D** cameras that computes the camera trajectory and a sparse 3D reconstruction (in the stereo and RGB-D case with true scale). It is able to detect loops and relocalize the camera in real time. We provide examples to run the SLAM system in the KITTI dataset as stereo or monocular, in the TUM dataset as RGB-D or monocular, and in the EuRoC dataset as stereo or monocular. We also provide a ROS node to process live monocular, stereo or RGB-D streams. **The library can be compiled without ROS.** ORB-SLAM2 provides a GUI to change between a *SLAM Mode* and *Localization Mode*, see section 9 of this document.



Related Publications:

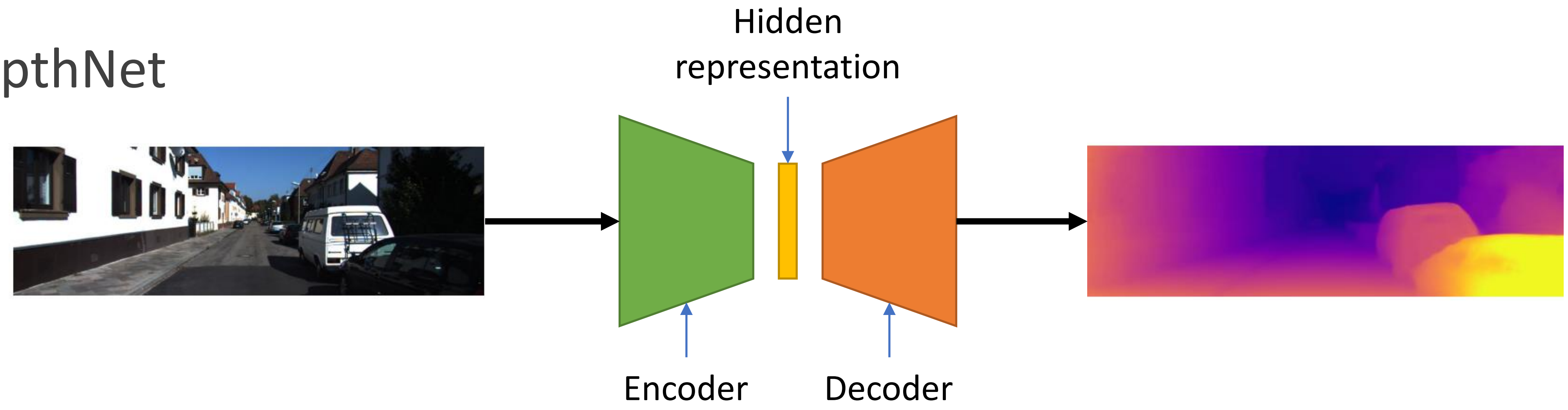
[Monocular] Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós. **ORB-SLAM: A Versatile and Accurate Monocular SLAM System.** *IEEE Transactions on Robotics,* vol. 31, no. 5, pp. 1147-1163, 2015. (**2015 IEEE Transactions on Robotics Best Paper Award**). PDF.

[Stereo and RGB-D] Raúl Mur-Artal and Juan D. Tardós. **ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras.** *IEEE Transactions on Robotics,* vol. 33, no. 5, pp. 1255-1262, 2017. PDF.

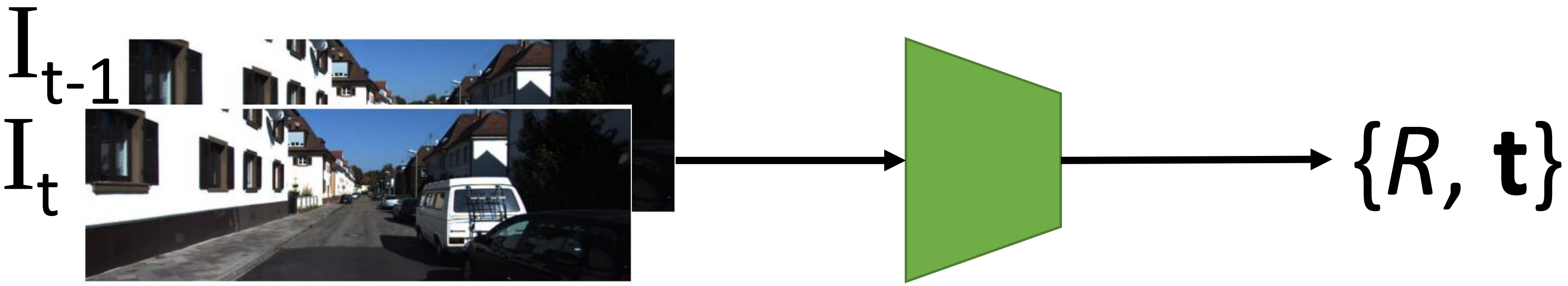
[DBoW2 Place Recognizer] Dorian Gálvez-López and Juan D. Tardós. **Bags of Binary Words for Fast Place Recognition in Image Sequences.** *IEEE Transactions on Robotics,* vol. 28, no. 5, pp. 1188-1197, 2012. PDF

Learning based

DepthNet

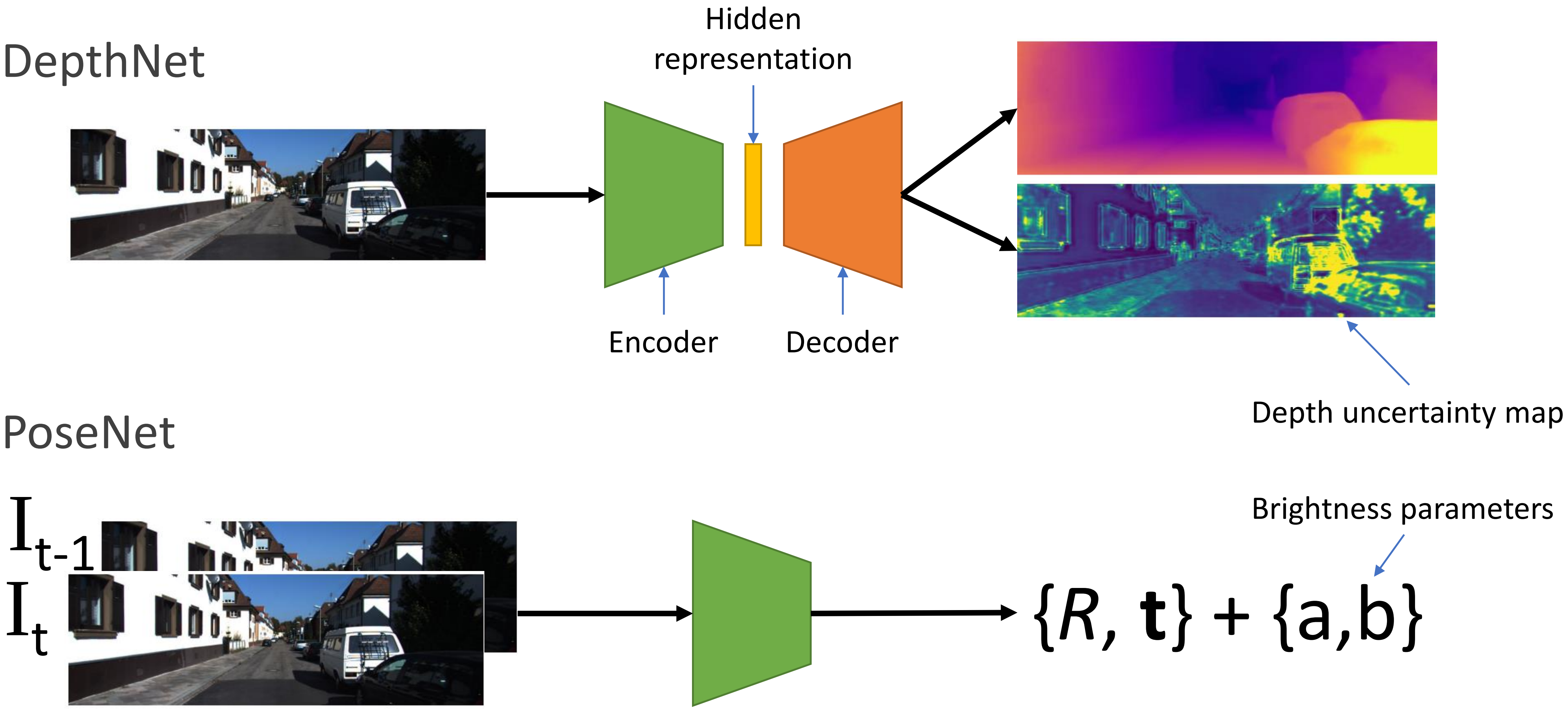


PoseNet



N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

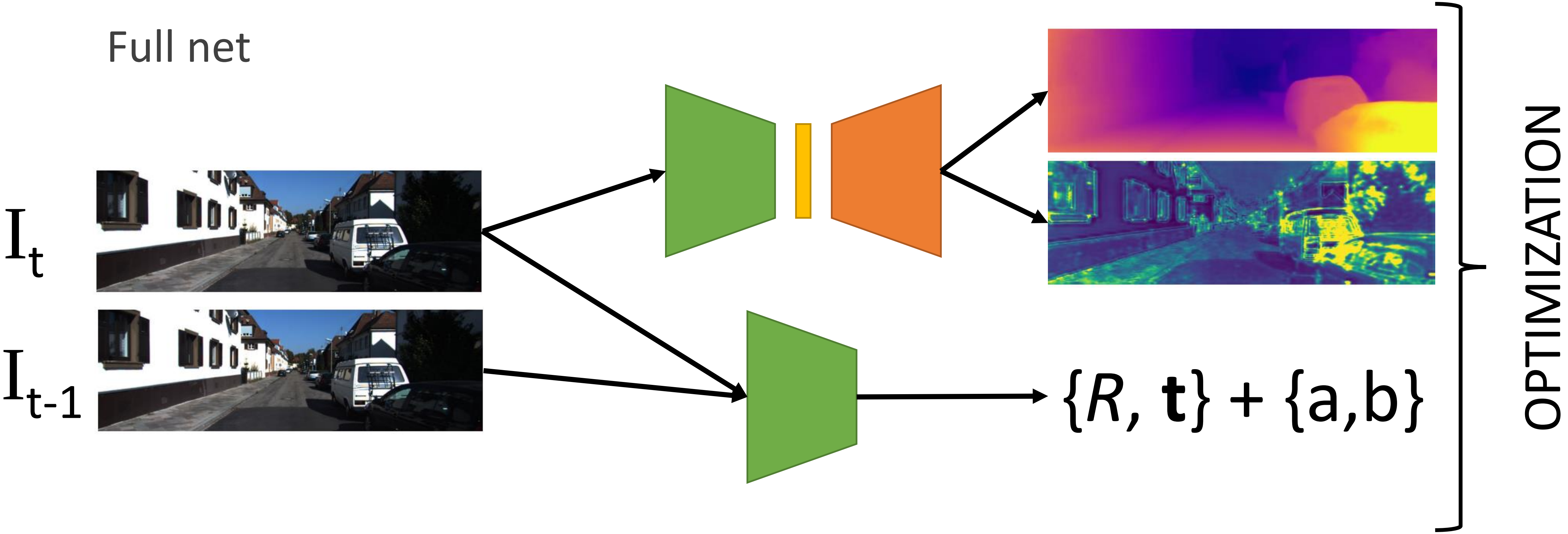
Learning based



N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

Learning based

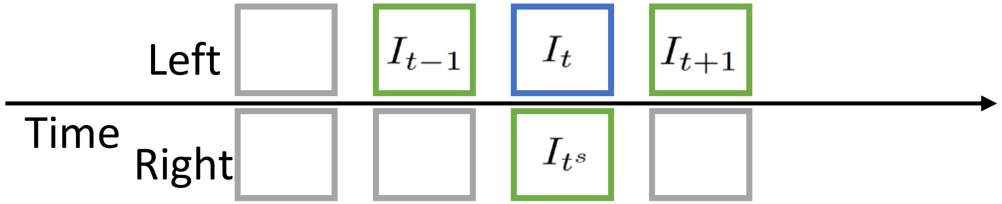
Full net



N. Yang, et al., “D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry” CVPR 2020.

Learning based

- To train the net, stereo pairs are required



- At first the loss function try to minimize the photometric re-projection error

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \min_{t'} r(I_t, I_{t' \to t})$$

where $I_{t'} \in \{I_{t-1}, I_{t+1}, I_{t^s}\}$ and $I_{t' \to t}$ is the warping of I_t using the transformation $\mathbf{T}_t^{t'}$ predicted by the PoseNet, and D_t , i.e. the depth map predicted by the DepthNet. Note that $\mathbf{T}_t^{t^s}$ is known and fixed.

N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

Learning based

- The function *r* in the loss is

$$r(I_a, I_b) = \frac{\alpha}{2}(1 - \text{SSIM}(I_a, I_b)) + (1-\alpha)||I_a - I_b||_1$$

where SSIM is the Structural Similarity Index. *r* is based on the **brightness constancy assumption** (BCA), that can be violated by changes in illumination.

- To better guarantee to satisfy the BCA, **brightness transformation parameter** are learned and used to adapt the illumination of I_t with $I_{t'}$ extending the loss as

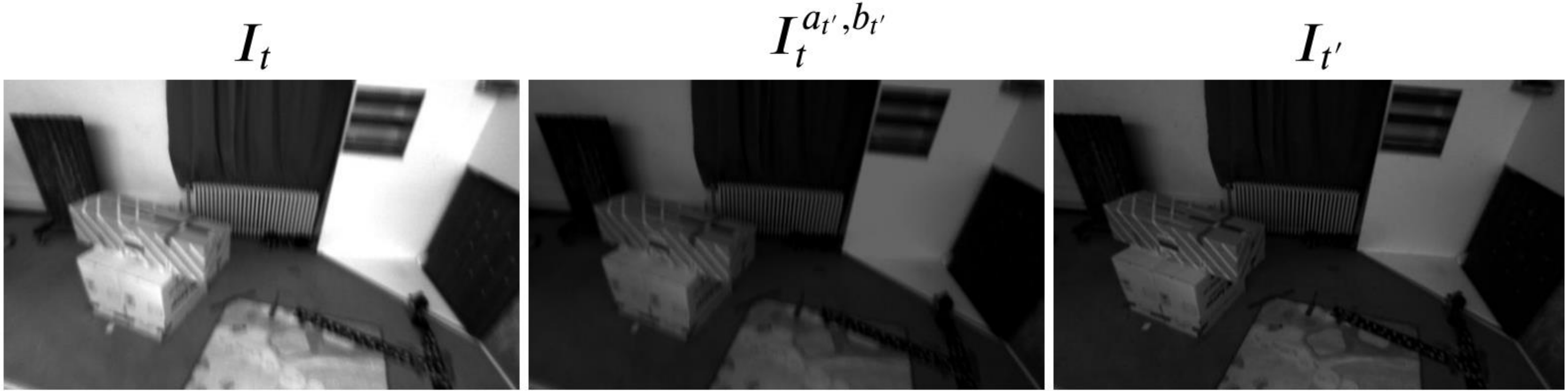
$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \min_{t'} r(I_t^{a_{t'}, b_{t'}}, I_{t' \to t})$$

where $I_t^{a_{t'}, b_{t'}} = a_{t \to t'} I_t + b_{t \to t'}$

N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

Learning based

- Example result of brightness adaptation

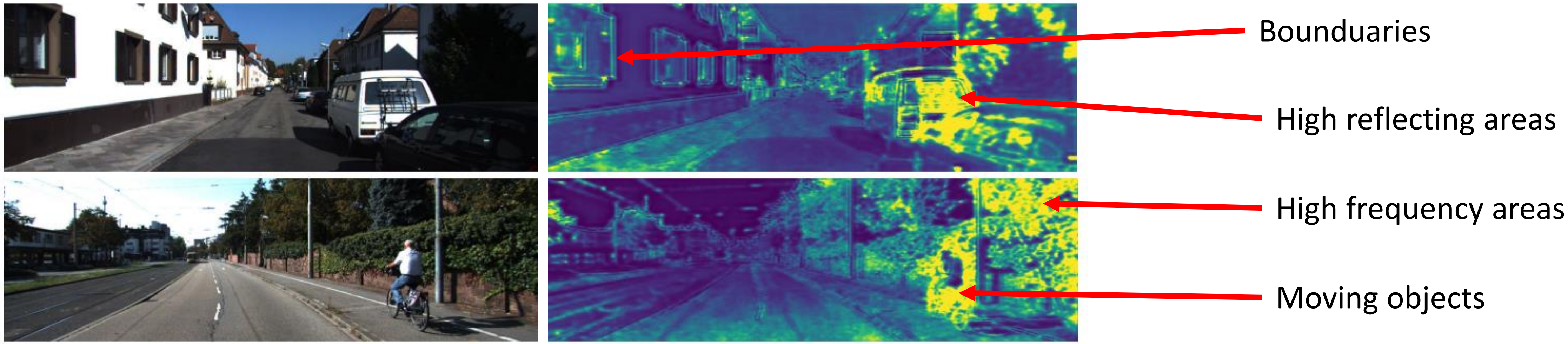


N. Yang, et al., “D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry” CVPR 2020.

Learning based

- To further improve the robustness of the system w.r.t. noisy data, the loss function is expanded to include an uncertainty map

$$L_{self} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \frac{\min_{t'} r(I_t^{a_{t'}, b_{t'}}, I_{t' \to t})}{\Sigma_t}$$



N. Yang, et al., "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry" CVPR 2020.

Learning based

- Finally, in order to avoid degenerate solution, a regularization term is introduced, considering both the brightness parameters and the uncertainty map

$$L_{total} = \frac{1}{|V|} \sum_{\mathbf{p} \in V} \frac{\min_{t'} r(I_t^{a_{t'}, b_{t'}}, I_{t' \to t})}{\Sigma_t} + \log \Sigma_t + \sum_{t'} (a_{t'} - 1)^2 + b_{t'}^2$$

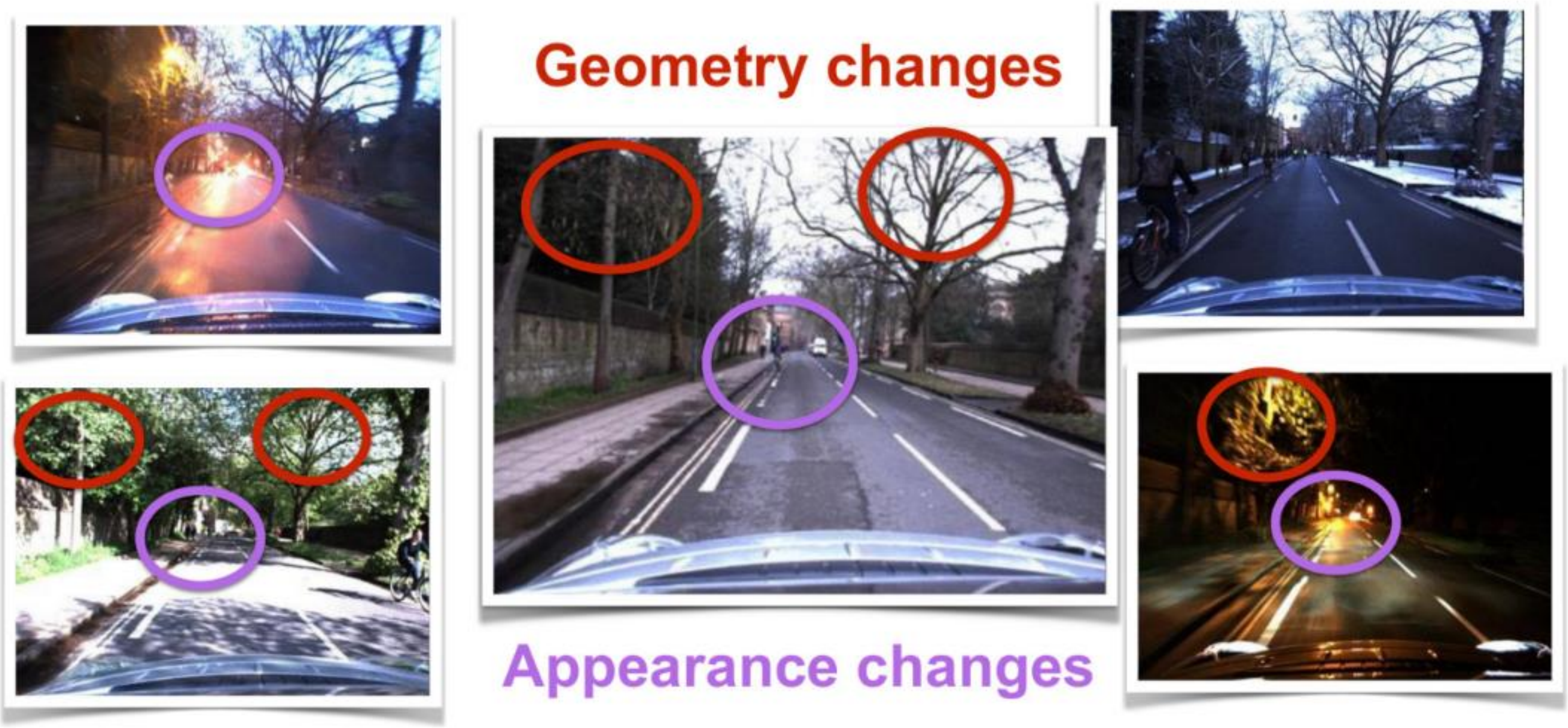
N. Yang, et al., “D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry” CVPR 2020.

Localization

- Suppose to have a database of maps with associated features
- **Goal**: localize a vehicle on the known maps using images
- At localization time we should try to find features extracted from the current image in the known map

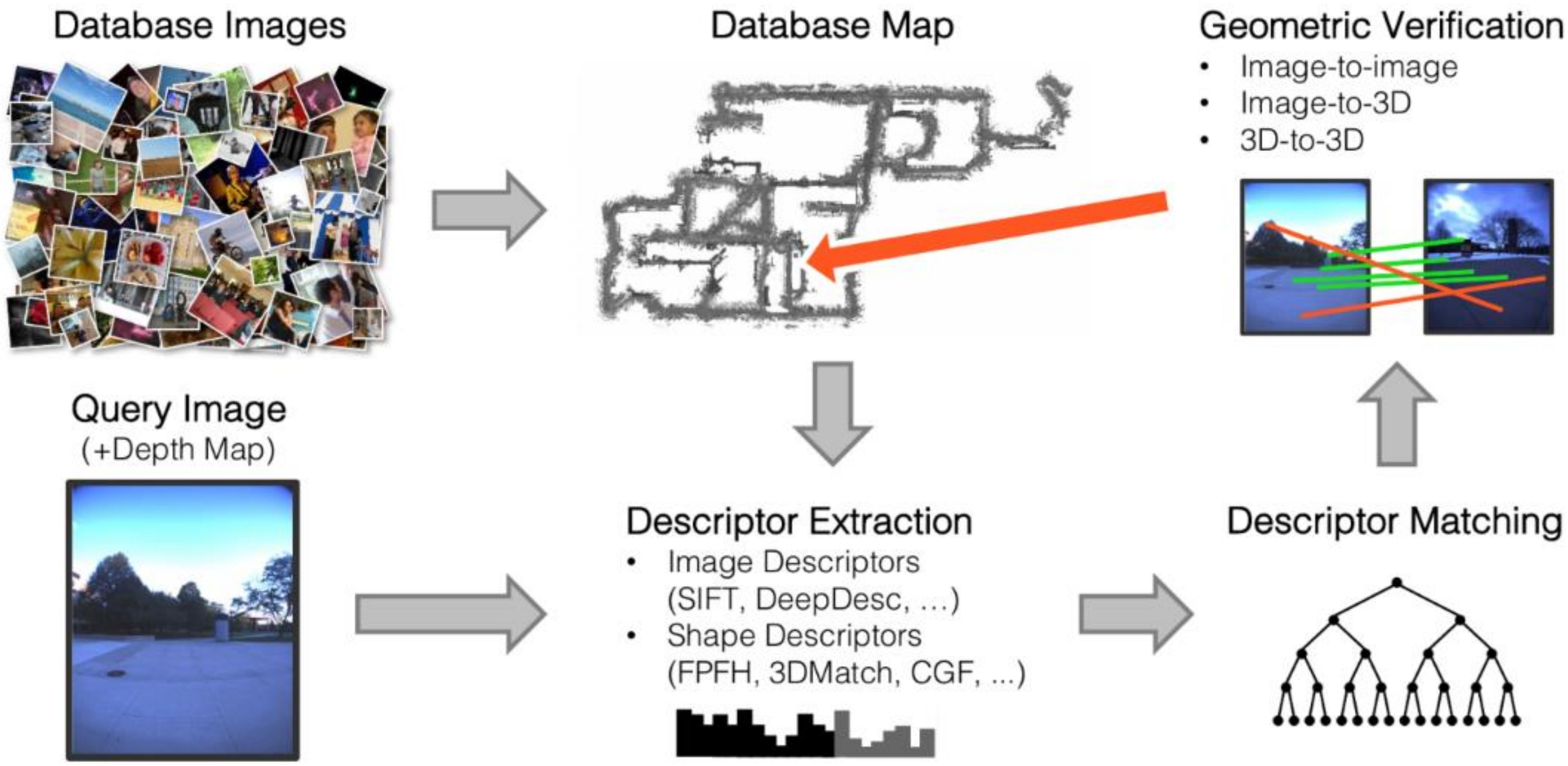
Localization

- Obviously, there are some challenges



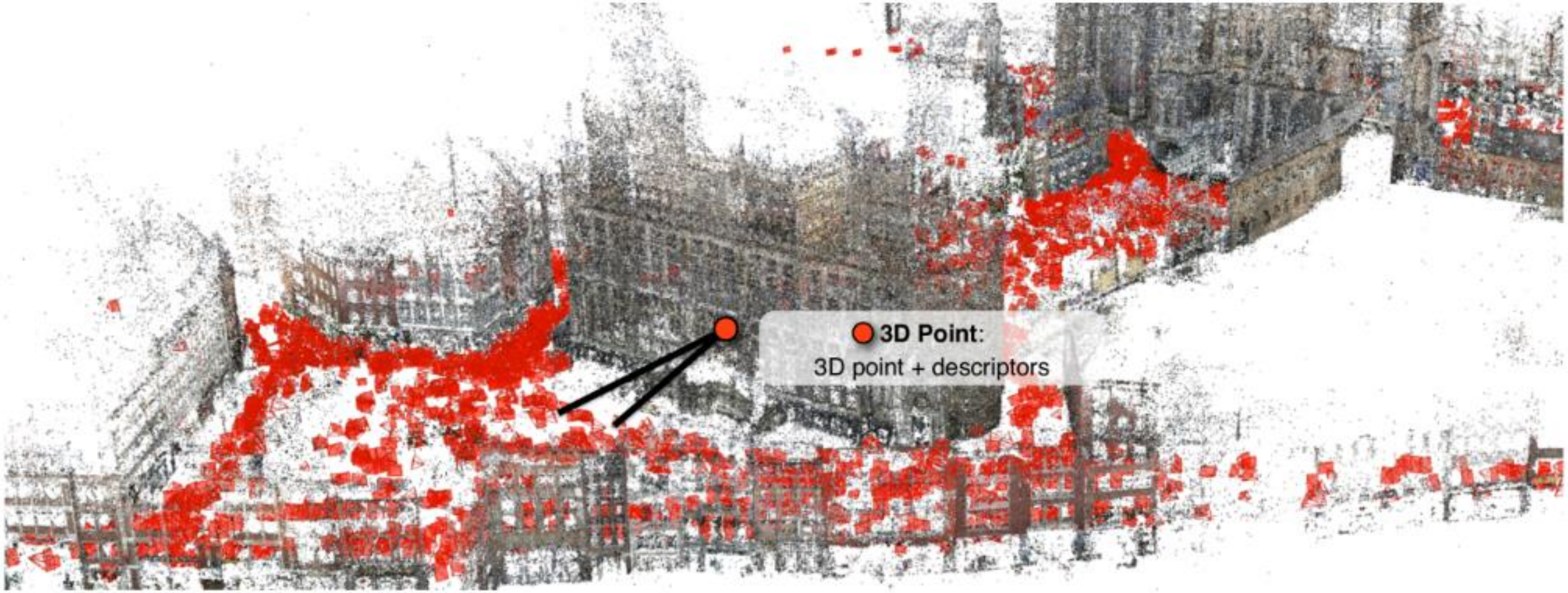
Slide from A. Geiger

Localization



Slide from T. Sattler

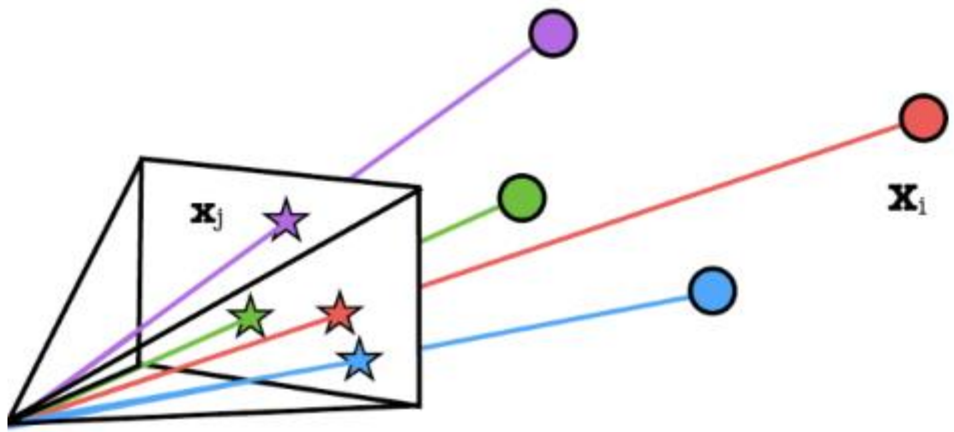
Localization



- In practice, we have a 3D sparse point cloud (SLAM, SfM)
- Each 3D point have one or more associated descriptor
- We have to find matches among the 3D point descriptors and those extracted from the input image

Slide from T. Sattler

Localization



- Once the 2D/3D correspondences are known, the pose of the input image can be estimated by minimizing

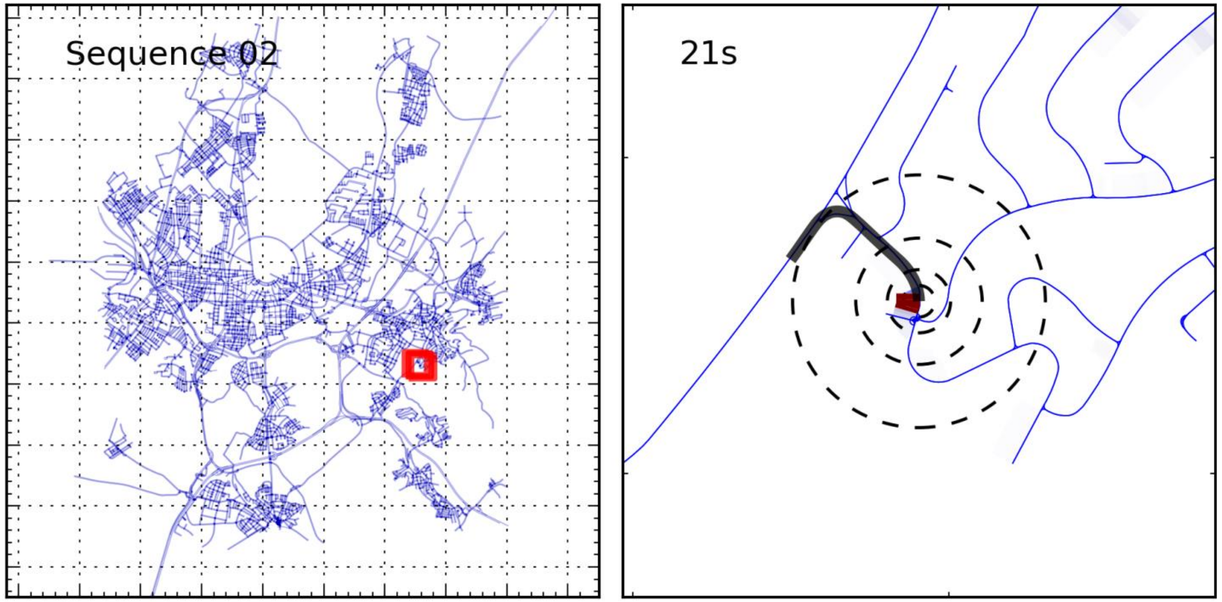
$$\sum_{i=1}^{N} ||\mathbf{x}_i - \pi(\mathbf{X}_i; \mathbf{r}, \mathbf{t})||^2$$

Slide from T. Sattler

Map-based Localization

- A different way to localize a vehicle

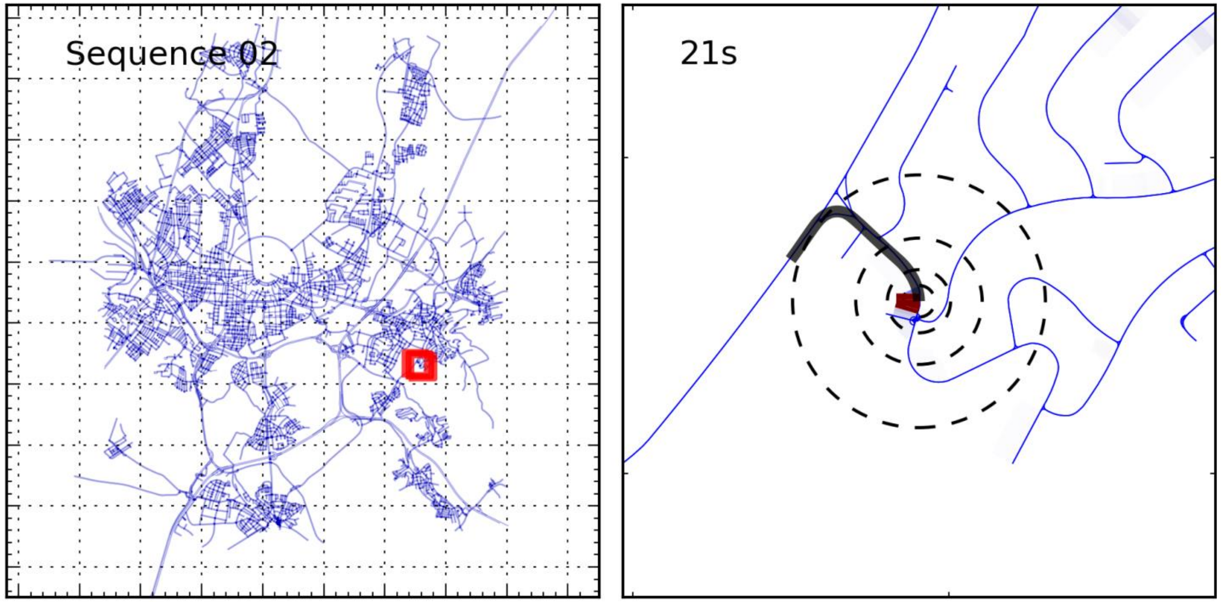
- The system tries to match the trajectory estimated by visual odometry with a street map (from OSM)



Brubaker, Geiger and Urtasun: Map-Based Probabilistic Visual Self-Localization. PAMI, 2016.

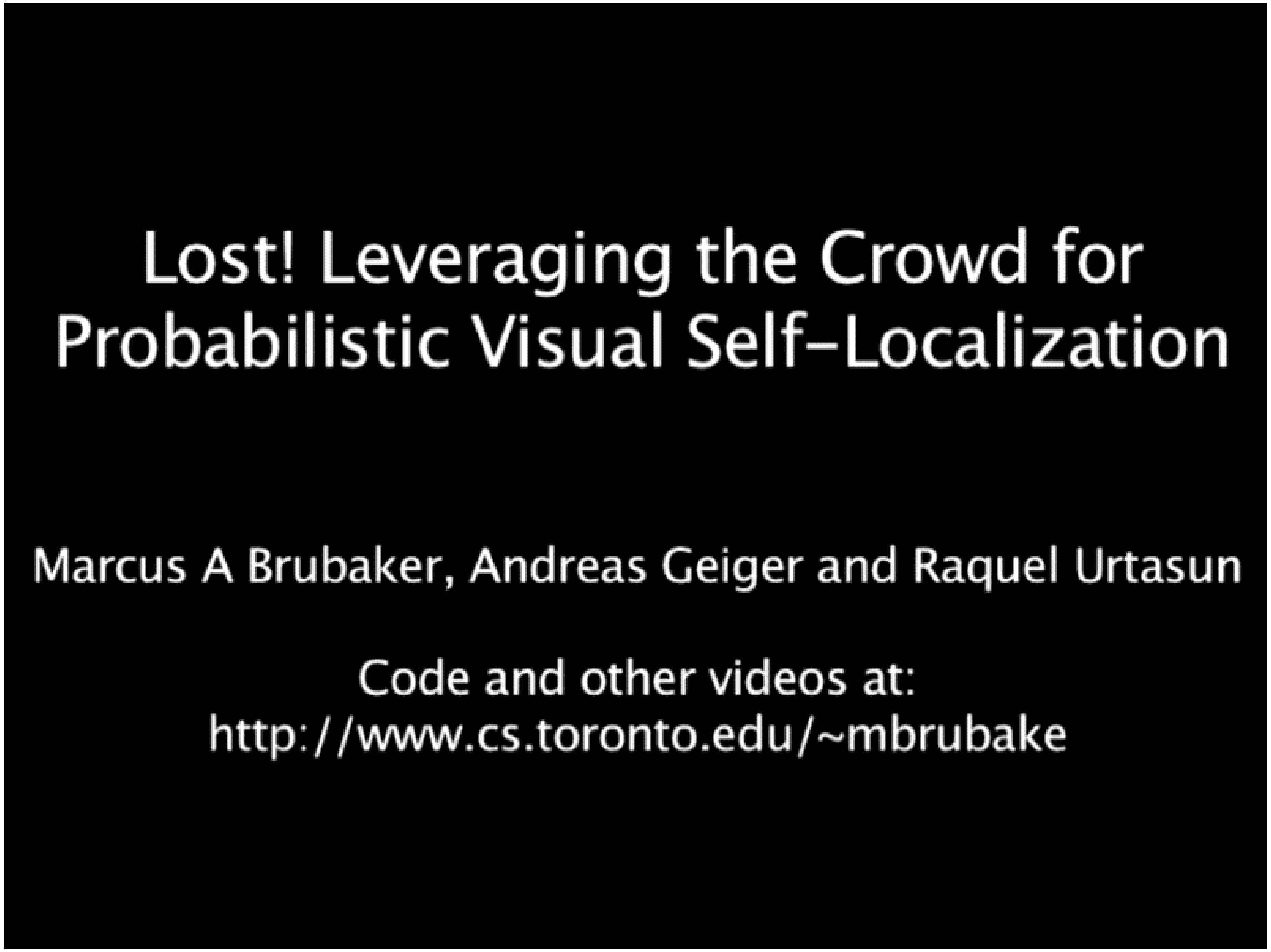
Map-based Localization

- **Idea**: to exploit the characteristics of a trajectory (e.g., straight segment length, curves, etc.) to find a matching pattern in a 2D street map



Brubaker, Geiger and Urtasun: Map-Based Probabilistic Visual Self-Localization. PAMI, 2016.

Map-based Localization



Brubaker, Geiger and Urtasun: Map-Based Probabilistic Visual Self-Localization. PAMI, 2016.