If you tell me what you look at, I could tell you where you are - Some state of the art pose estimation methods in computer vision

Myriam Servières

If you tell me what you look at, I could tell you where you are

Some state of the art pose estimation methods in computer vision

Myriam Servières
École Centrale Nantes, AAU-CRENAU, IRSTV

École Centrale Nantes, AAU-CRENAU, IRSTV

IPIN 2018

Nantes - 24 september 2018

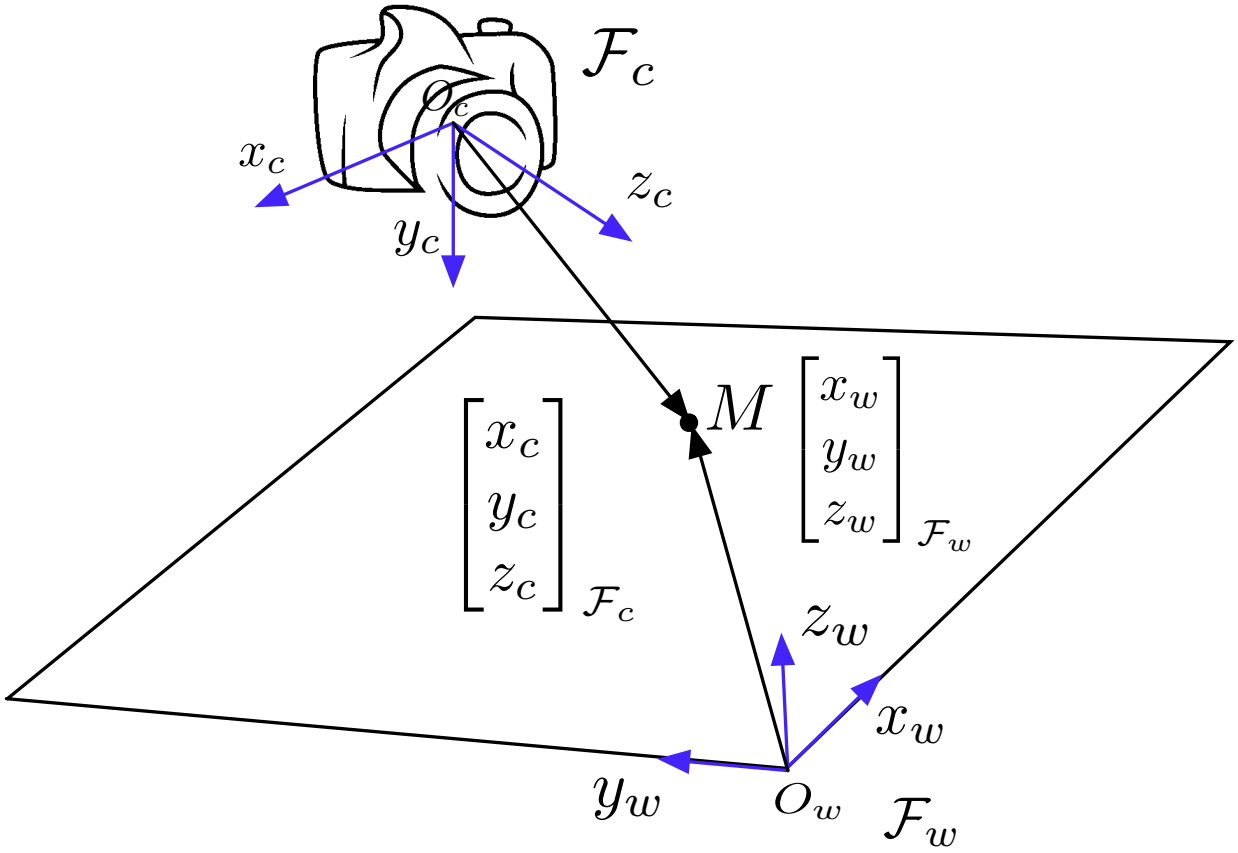

Cité des Congrès
NOVOTEL
NOVOTEL


Gare de Nantes
Parking gare de Nantes Nord - EFFIA
La Compagnie du Café Théâtre
Spectacles comiques &...
Parking Gare Sud 1
Capgemini Technology Services
La Tour LU
le lieu unique
Square Élisa-Mercœur
Canal Saint-Félix
L'Erdre
Marguerite Station Gare SNCF Sud
Rue Marcel Paul
Mail Pablo Picasso
Allée Baco
Rue Crucy
Carrefour City
Théâtre du Sphinx
Allée de l'Indre
Rue de l'Indre
La Cité des Congrès de Nantes
Espace événementiel...
Stade Marcel Saupin
John Doe Escape Game Nantes
Square du Lait de Mai
Rue de Mayence
Passage Berthault
Rue Perrault
Rue Baron
Rue Fouré
Rue de Rieux
Chaussée de la Madeleine
Avenue Jean Claude Bonduelle
Bras de la Madeleine
Pont Willy Brandt
Racing Club Nantais
Rue Perelle
Rue Marmontel
Quai André Morice
La Loire
Boulevard Louis Barthou
Rue André Tardieu
Centre Hospitalier Vétérinaire Atlantia
Quai Moncousu
Bras de la Madeleine
Lycée Nelson Mar


Harmonie Atlantique
Segafredo
VELTINS


Parc des Capucins
Rue Général Bedeau
Rue Noire
Rue Haute Roche
Rue Molac
Rue d'Iéna
Rue Russeil
Rue de Bel air
Rue Paul Bellamy
Rue de la Distillerie
L'Erdre
Quai Henri Barbusse
Rue Pitre Chevalier
Rue Colonel Boutin
Rue Henri Cochard
Allée de la Reine Margot
Rue Préfet Bonnefoy
Rue Adolphe Moitié
Rue Châteaubriand
Quai de Versailles
Quai Ceineray
Rue Tournefort
Rue du Poitou
Rue d'Auvours
Place Viarme
Rue Fredureau
Lycée Talensac
Rue Yves Bodiguel
Marché de Talensac
Square Maquis-de-Saffré
Croix à la mémoire du général Charette
Monument of Charles de Gaulle
Parking Cathé
Musée d'arts de
Domino's Pizza Nantes Centre
Rue Sarrazin
Rue Porte Neuve
Passage Louis Lévesque
Square Etienne Destranges
Rue d'Erlon
Rue de la Bastille
Square Faustin Hélie
Hôtel de Ville
Cathédrale Saint-Pierre-et-Saint-Paul
Radisson Blu Hotel, Nantes
Le Nid
GO SPORT NANTES
Square Amiral Halgand

Camera Pose



Camera pose

Camera position and orientation relative to a fixed coordinate system

Definitions

Camera pose

Camera position and orientation relative to a fixed coordinate system

Definitions

Camera pose

Camera position and orientation relative to a fixed coordinate system

Computer Vision *(CV)*

Interdisciplinary field that study and develop techniques to enable a computer system or artificial intelligence system to analyze and understand visual data obtained using cameras. Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images.

Why do we need the pose for?

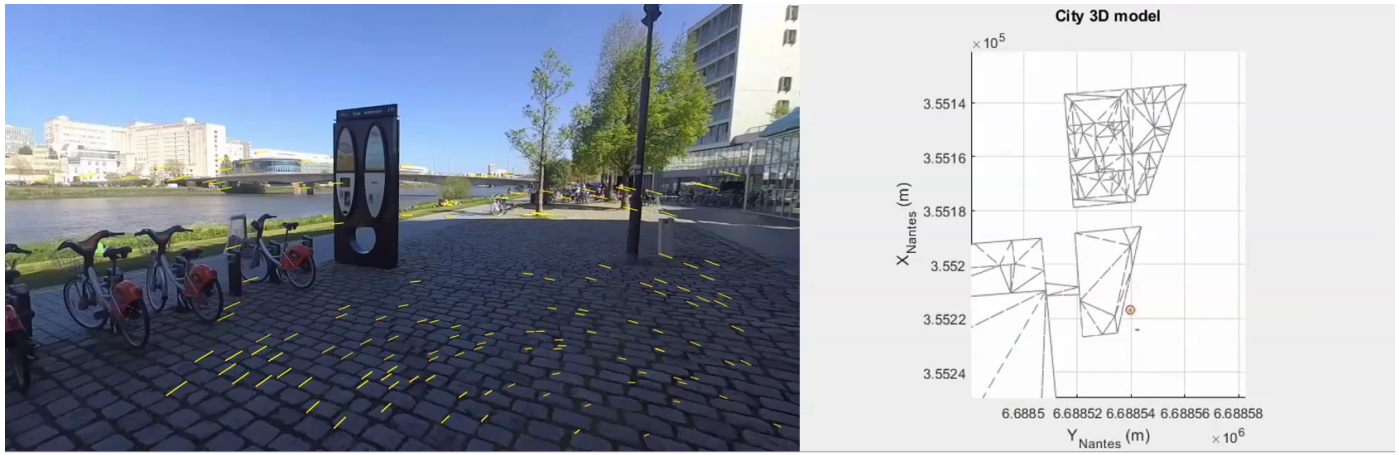
- Retrieving information from image

work from [ASR17]

- Knowing a displacement only from a video
- Augmented Reality
- Reconstructing a 3D model
- ...

Why do we need the pose for?

- Retrieving information from image
- Knowing a displacement only from a video



work from [ASR18]

- Augmented Reality
- Reconstructing a 3D model
- ...

Why do we need the pose for?

- Retrieving information from image
- Knowing a displacement only from a video
- Augmented Reality

work from [ASR17]

- Reconstructing a 3D model
- . . .

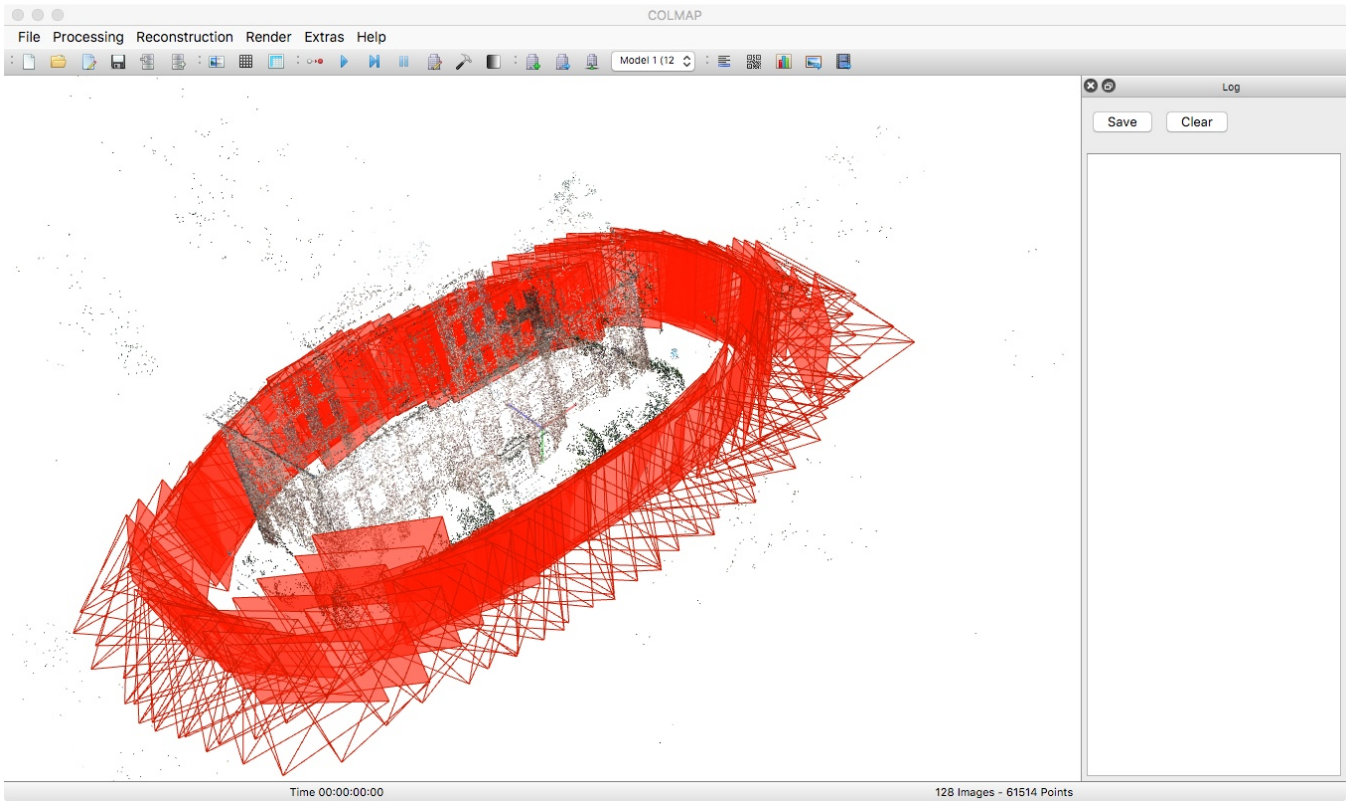
Why do we need the pose for?

- Retrieving information from image
- Knowing a displacement only from a video
- Augmented Reality
- Reconstructing a 3D model

- ...

Why do we need the pose for?

- Retrieving information from image
- Knowing a displacement only from a video
- Augmented Reality
- Reconstructing a 3D model



3D model reconstructed with COLMAP with an example dataset [SF16, SZPF16]

▸ https://demuc.de/colmap/#tutorial

Why do we need the pose for?

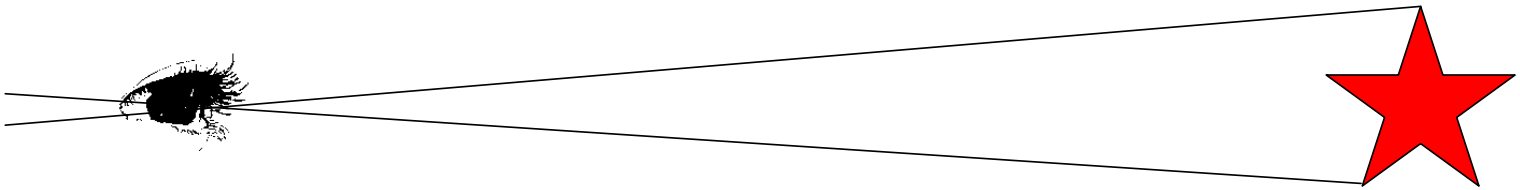
- Retrieving information from image
- Knowing a displacement only from a video
- Augmented Reality
- Reconstructing a 3D model
- ...

Outline

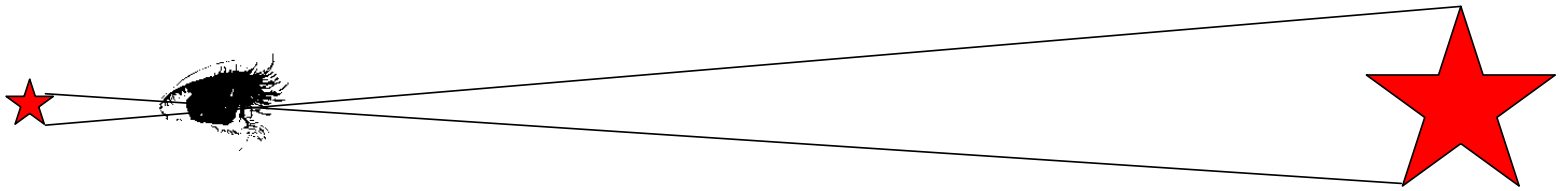
Outline

Outline

Pinhole Camera Model



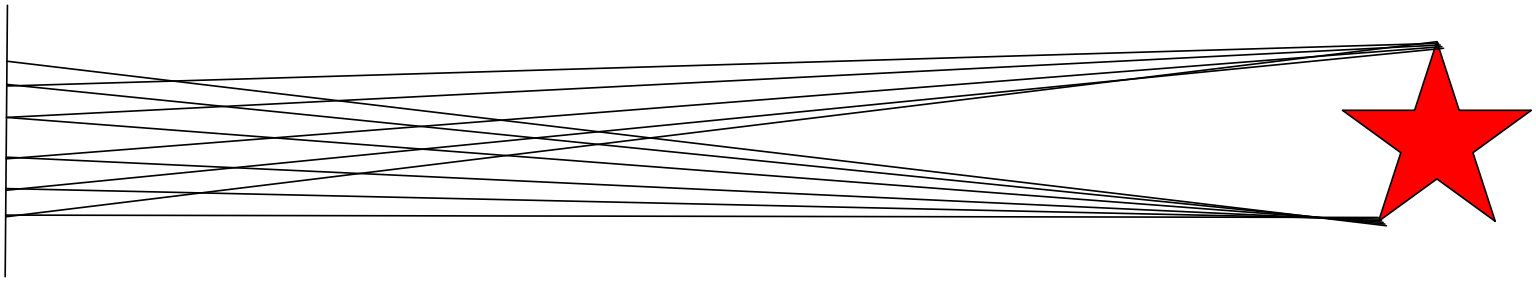
Pinhole Camera Model



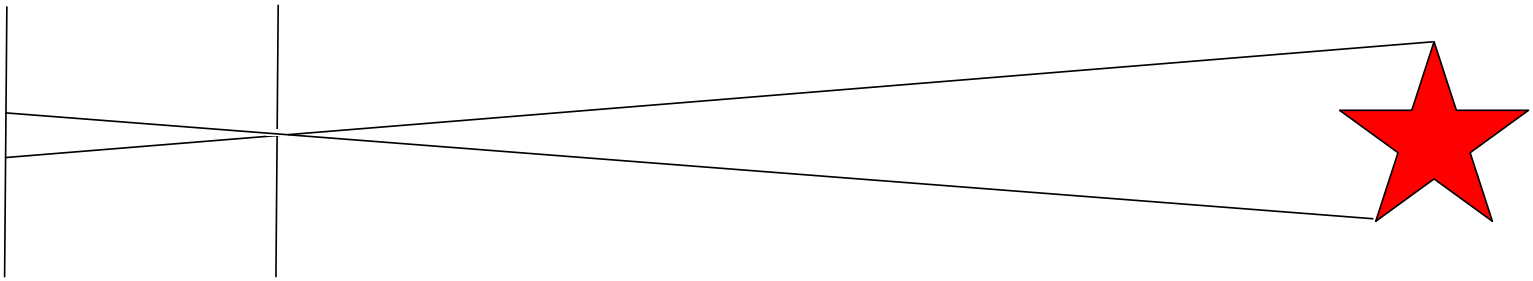
Pinhole Camera Model

imager

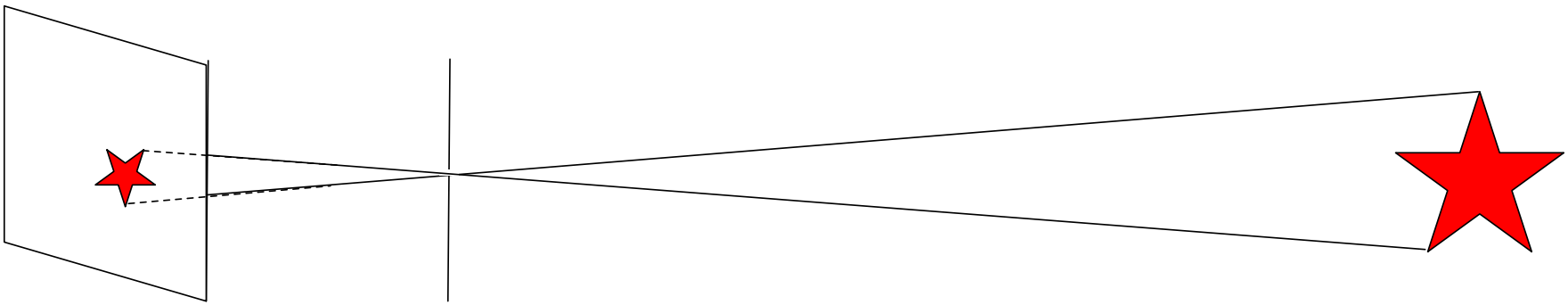
Pinhole Camera Model



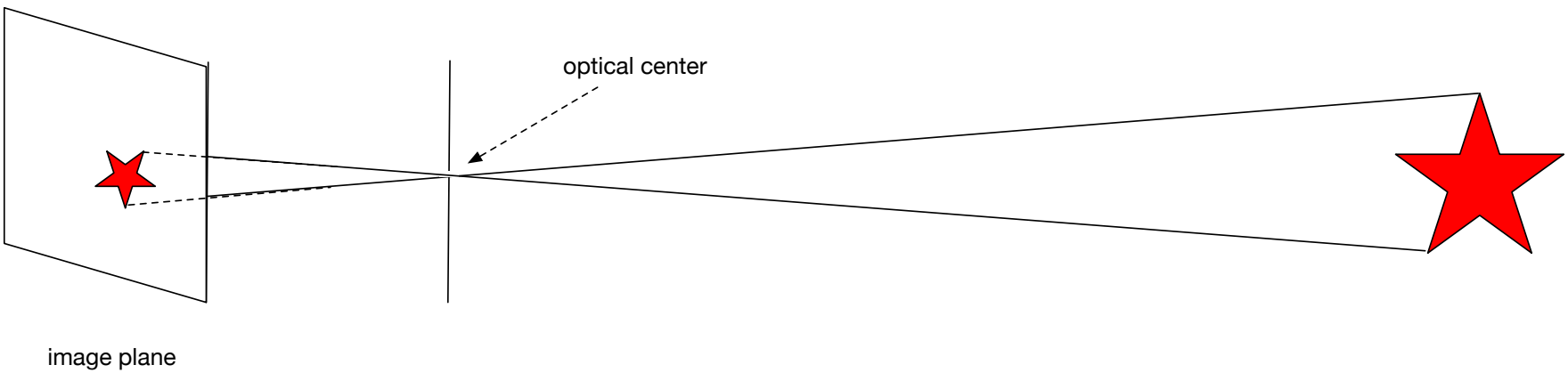
Pinhole Camera Model



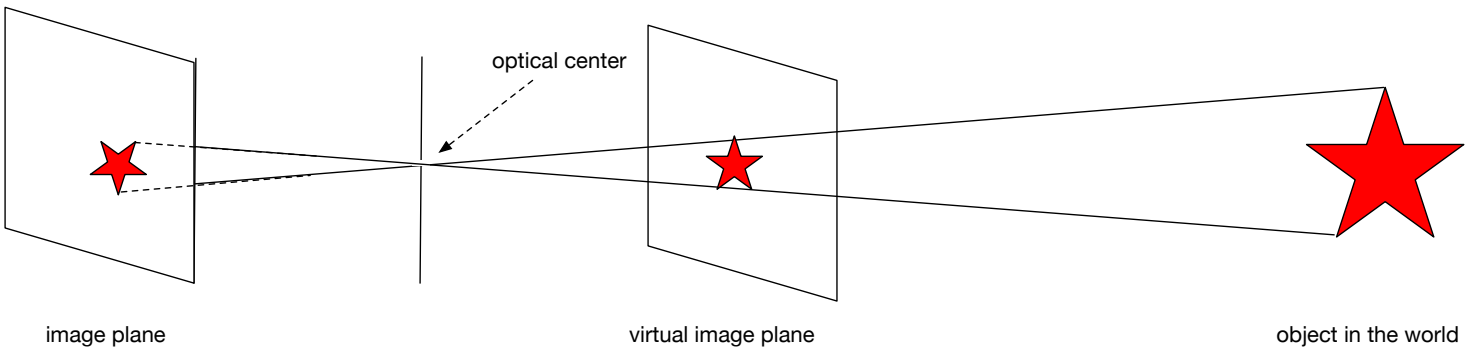
Pinhole Camera Model



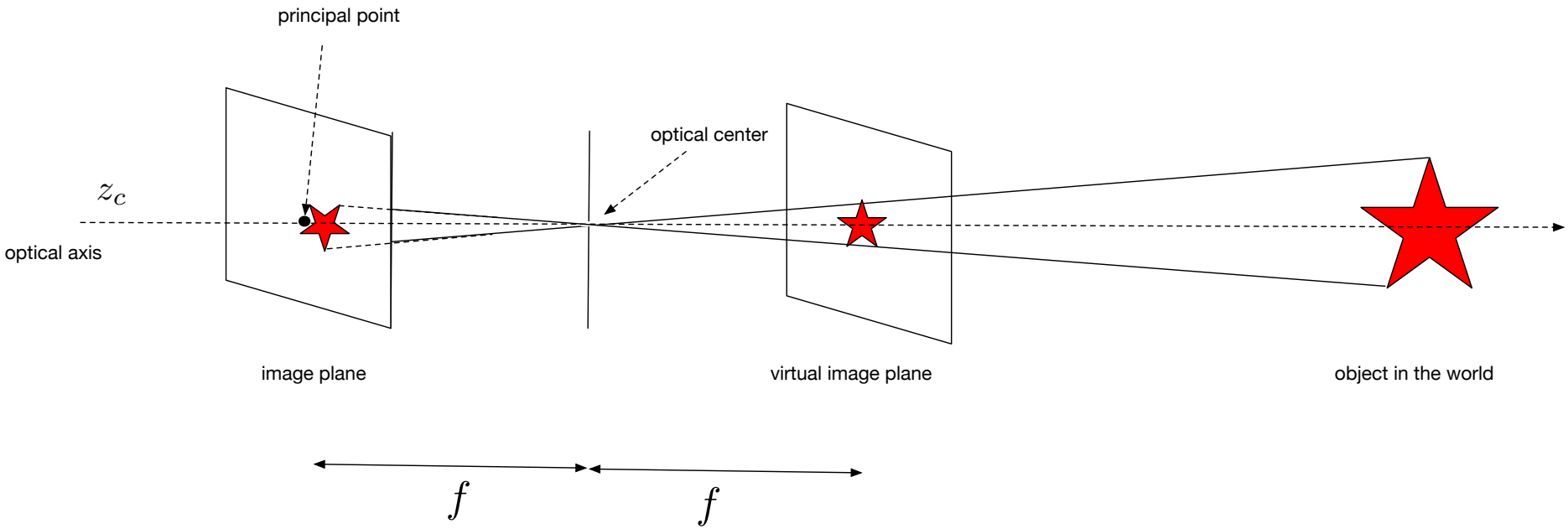
Pinhole Camera Model



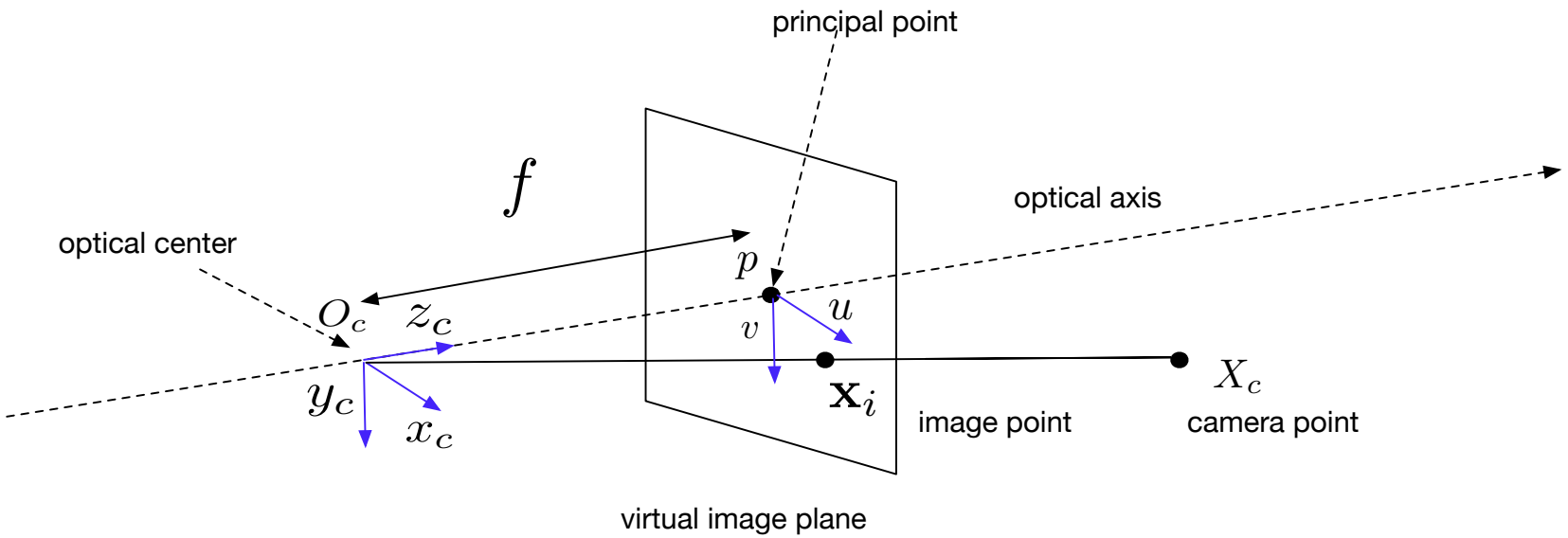
Pinhole Camera Model



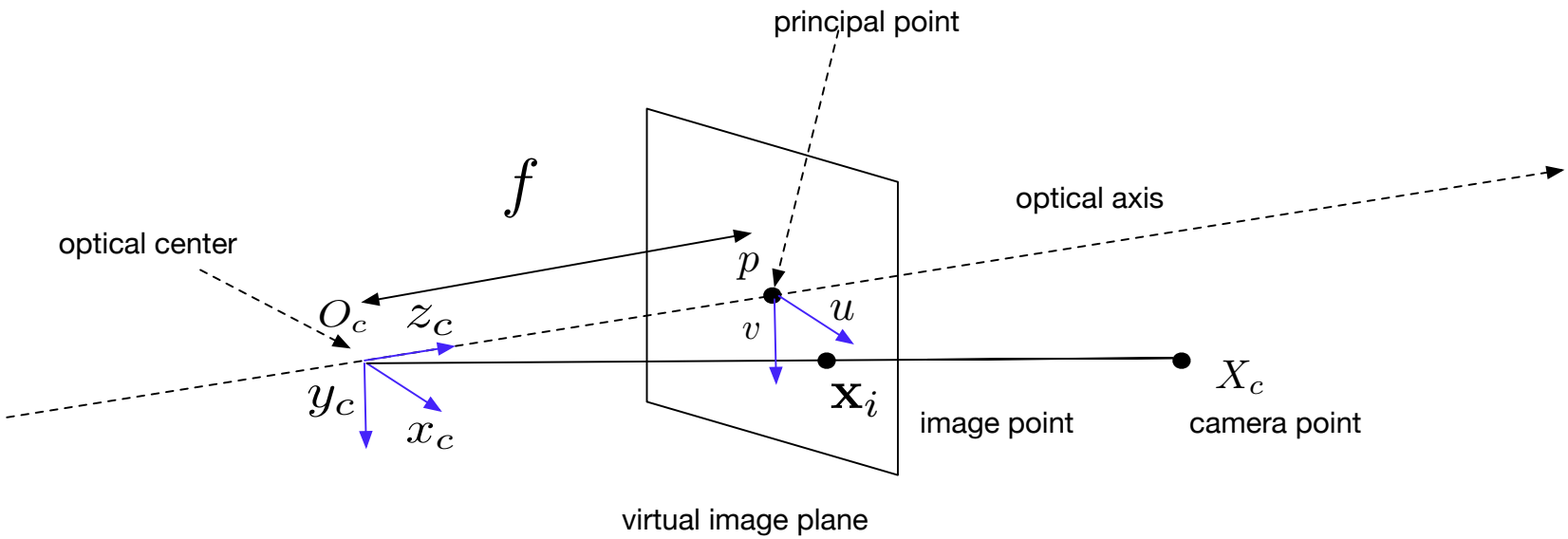
Pinhole Camera Model



Pinhole Camera Model



Pinhole Camera Model

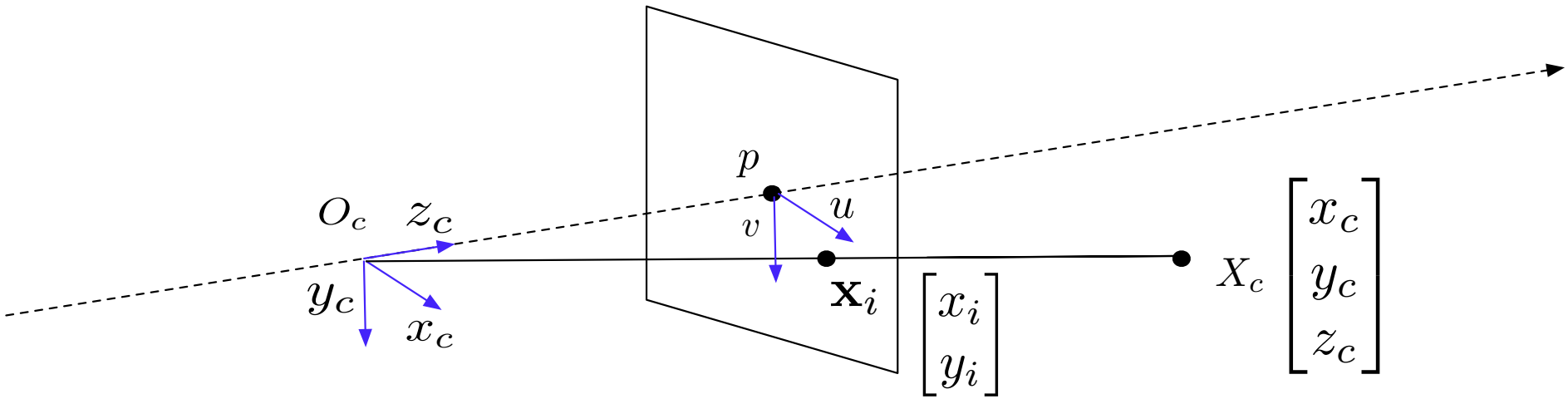


- Model of the camera's geometry
- Distortion model of the lens

Outline

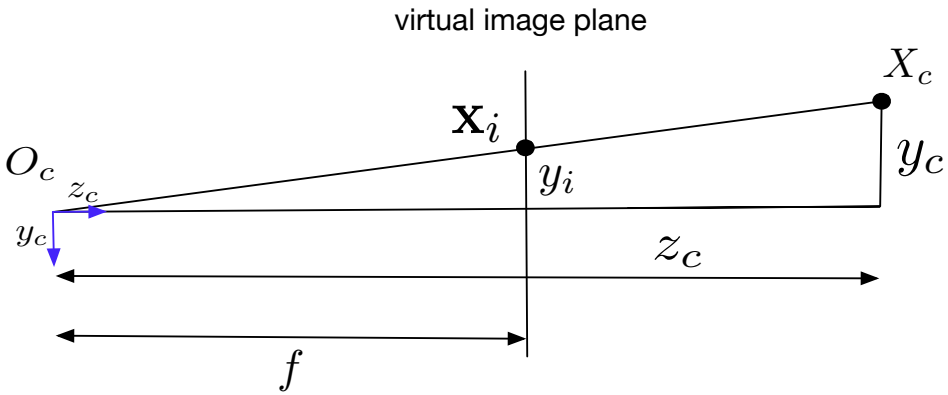
Outline

Link between imaged point and point on image

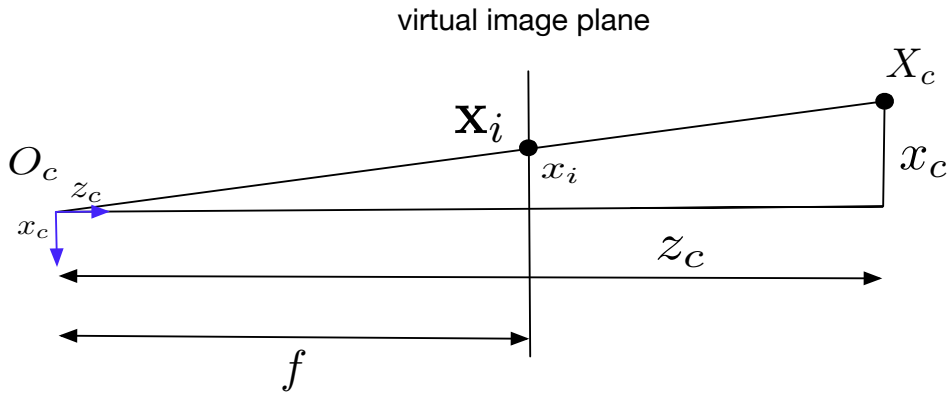


Perspective projection

Link between imaged point and point on image

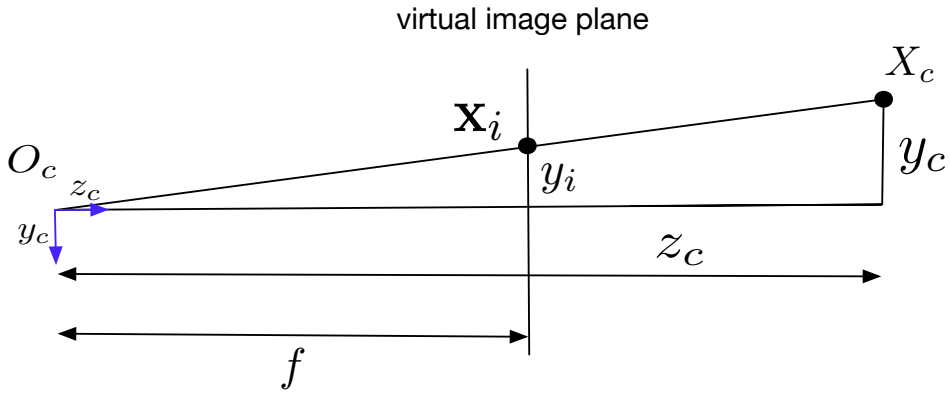


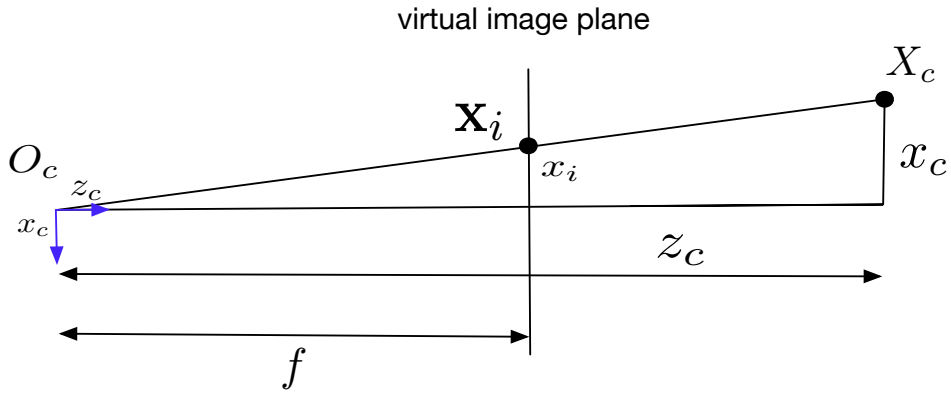
$$y_i = f\frac{y_c}{z_c}$$



$$x_i = f\frac{x_c}{z_c}$$

Link between imaged point and point on image

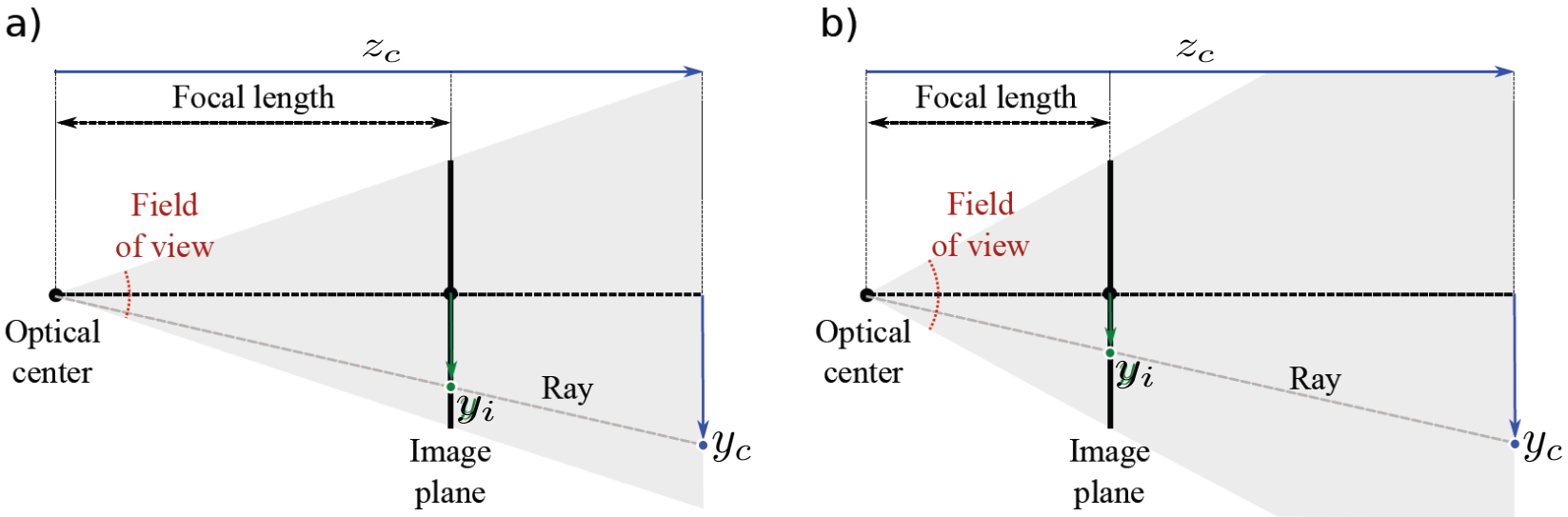




$$y_i = f\frac{y_c}{z_c} \qquad x_i = f\frac{x_c}{z_c}$$

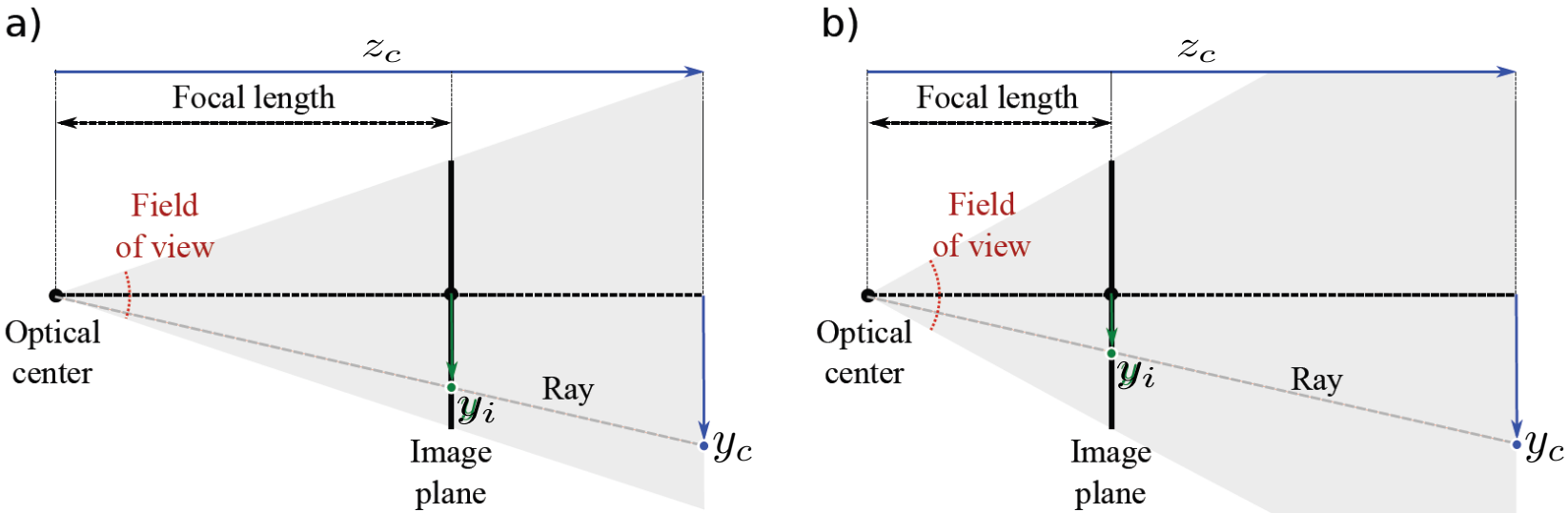
x_i, y_i, x_c, y_c and z_c are measured in the same real-world units (e.g. mm)

Focal length and photoreceptor spacing



[slightly adapted from [Pri12]]

Focal length and photoreceptor spacing

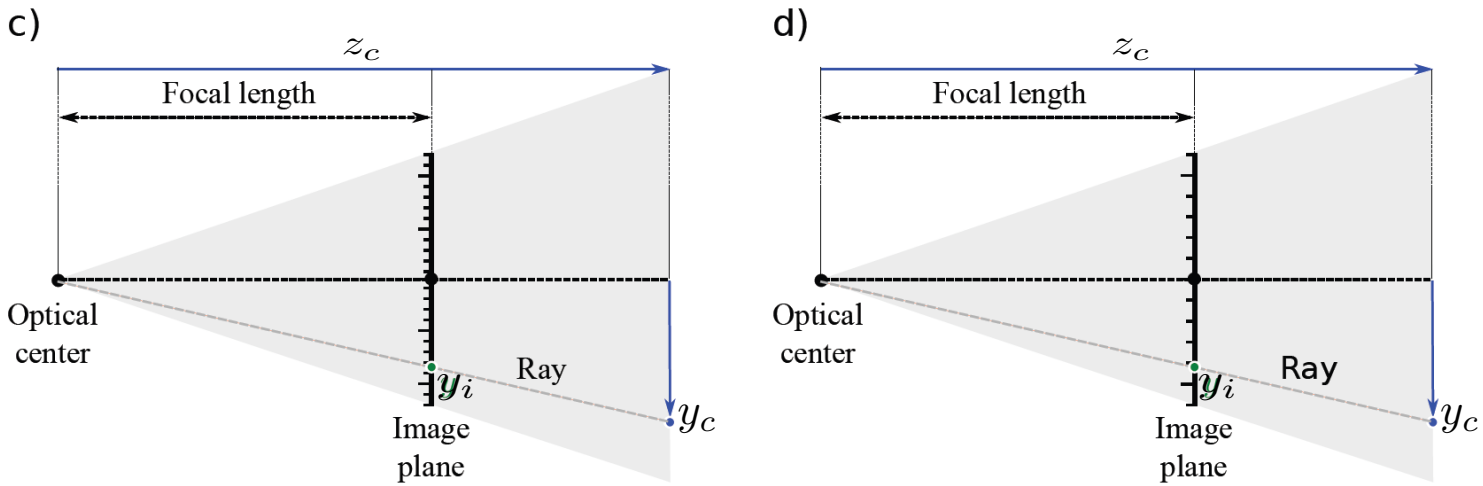


[slightly adapted from [Pri12]]

Camera field of view

Total angular range that is imaged (usually different in the x- and y-directions)

Focal length and photoreceptor spacing



[slightly adapted from [Pri12]]

Unit in pixel on the image

Rectangular pixels

- two different focal lengths : f_x and f_y

Unit in pixel on the image

Rectangular pixels

- two different focal lengths : f_x and f_y
- $f_x = f.s_x$ and $f_y = f.s_y$ with
  - f the physical focal length in the chosen real-world unit (e.g. mm)
  - s_x (resp. s_y) the size of the individual imager elements along x (resp. y) in px/chosen real-world unit (e.g. px/mm)

Unit in pixel on the image

Rectangular pixels

- two different focal lengths : f_x and f_y
- $f_x = f.s_x$ and $f_y = f.s_y$ with
  - f the physical focal length in the chosen real-world unit (e.g. mm)
  - s_x (resp. s_y) the size of the individual imager elements along x (resp. y) in px/chosen real-world unit (e.g. px/mm)
- Only f_x and f_y can be derived by the calibration process

Unit in pixel on the image

Rectangular pixels

- two different focal lengths : f_x and f_y
- $f_x = f.s_x$ and $f_y = f.s_y$ with
  - f the physical focal length in the chosen real-world unit (e.g. mm)
  - s_x (resp. s_y) the size of the individual imager elements along x (resp. y) in px/chosen real-world unit (e.g. px/mm)
- Only f_x and f_y can be derived by the calibration process

$$u = f_x \frac{x_c}{z_c}$$

$$v = f_y \frac{y_c}{z_c}$$

Representation in homogeneous coordinates

- $X_c = [x_c, y_c, z_c]^T$ correspond to $\mathbf{x}_{i_{pix}} = [u, v]^T = [f_x \frac{x_c}{z_c}, f_y \frac{y_c}{z_c}]^T$

Representation in homogeneous coordinates

- $X_c = [x_c, y_c, z_c]^T$ correspond to $\mathbf{x}_{i_{pix}} = [u, v]^T = [f_x \frac{x_c}{z_c}, f_y \frac{y_c}{z_c}]^T$
- non-linear operation $(/z_c) \rightarrow$ perspective projection

Representation in homogeneous coordinates

- $X_c = [x_c, y_c, z_c]^T$ correspond to $\mathbf{x}_{i_{pix}} = [u, v]^T = [f_x \frac{x_c}{z_c}, f_y \frac{y_c}{z_c}]^T$
- non-linear operation $(/z_c) \rightarrow$ perspective projection
- Use homogeneous coordinates to get a linear form (add one dimension).

$$\begin{bmatrix} u \\ v \end{bmatrix} \rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \rightarrow \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

Representation in homogeneous coordinates

- $X_c = [x_c, y_c, z_c]^T$ correspond to $\mathbf{x}_{i_{pix}} = [u, v]^T = [f_x \frac{x_c}{z_c}, f_y \frac{y_c}{z_c}]^T$
- non-linear operation $(/z_c) \rightarrow$ perspective projection
- Use homogeneous coordinates to get a linear form (add one dimension).

$$\begin{bmatrix} u \\ v \end{bmatrix} \rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \rightarrow \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

- conversion from homogeneous coordinates

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} \rightarrow \begin{bmatrix} u/w \\ v/w \end{bmatrix} \text{ and } \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}$$

Representation in homogeneous coordinates

- Using homogeneous coordinates

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \frac{x_c}{z_c} \\ f_y \frac{y_c}{z_c} \end{bmatrix}$$

can be expressed in a matrix form

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Representation in homogeneous coordinates

- Using homogeneous coordinates

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \frac{x_c}{z_c} \\ f_y \frac{y_c}{z_c} \end{bmatrix}$$

can be expressed in a matrix form

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \quad \text{or } \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

with λ a scale factor

Offset and skew parameters

- Principal point p is not at $[0, 0]^T$ in $\mathcal{F}_i$

Offset and skew parameters

- Principal point p is not at $[0, 0]^T$ in $\mathcal{F}_i$
- Point $[0, 0]^T$ is at upper-left corner and $p = [u_0, v_0]^T$

Offset and skew parameters

- Principal point p is not at $[0, 0]^T$ in $\mathcal{F}_i$
- Point $[0, 0]^T$ is at upper-left corner and $p = [u_0, v_0]^T$

$$u = f_x \frac{x_c}{z_c} + u_0$$

$$v = f_y \frac{y_c}{z_c} + v_0$$

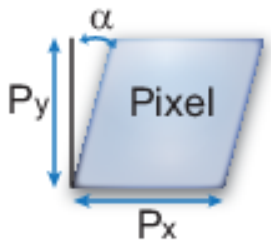
Offset and skew parameters

- Principal point p is not at $[0,0]^T$ in $\mathcal{F}_i$
- Point $[0,0]^T$ is at upper-left corner and $p = [u_0, v_0]^T$

$$u = f_x \frac{x_c}{z_c} + u_0$$

$$v = f_y \frac{y_c}{z_c} + v_0$$

- Pixels may have a skew factor s



©Mathworks source

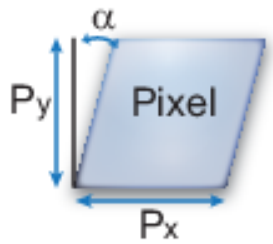
Offset and skew parameters

- Principal point p is not at $[0,0]^T$ in $\mathcal{F}_i$
- Point $[0,0]^T$ is at upper-left corner and $p = [u_0, v_0]^T$

$$u = f_x \frac{x_c}{z_c} + u_0$$

$$v = f_y \frac{y_c}{z_c} + v_0$$

- Pixels may have a skew factor s



©Mathworks source

$$u = f_x \frac{x_c + s.y_c}{z_c} + u_0$$

$$v = f_y \frac{y_c}{z_c} + v_0$$

Offset and skew parameters

- Full pinhole camera model in a matrix form

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

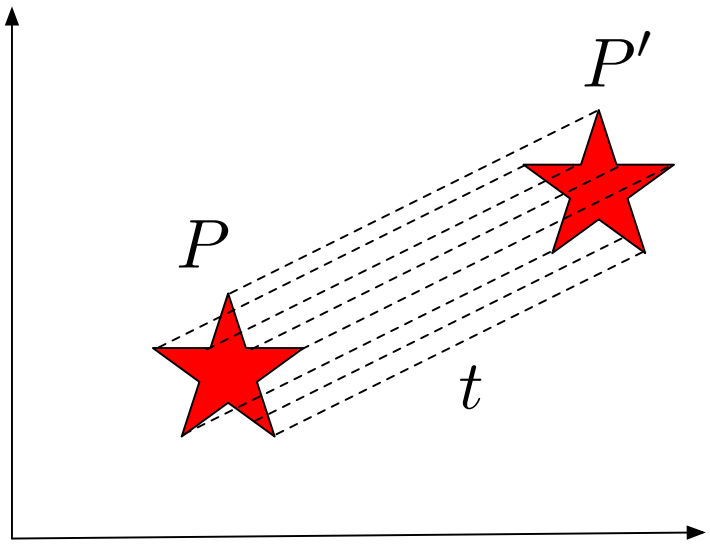
(linear!)

Intrinsic parameters matrix

$$\mathbf{K} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Outline

2D translation



$$P = [x, y]^T \rightarrow [x, y, 1]^T$$
$$t = [t_x, t_y]^T \rightarrow [t_x, t_y, 1]^T$$

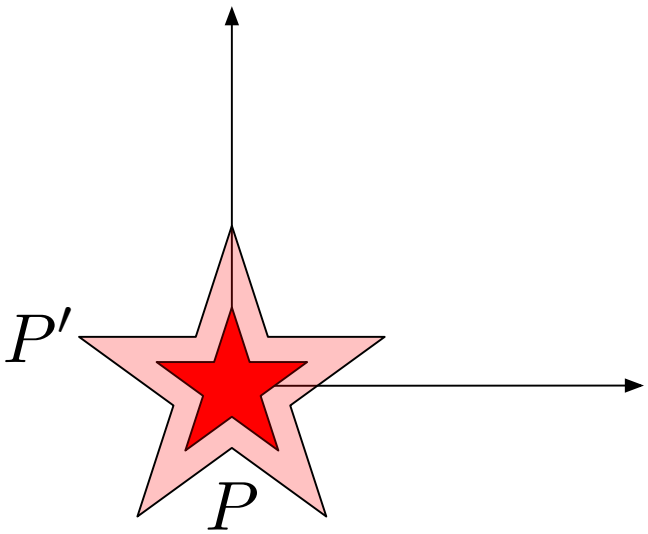
$$P' = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

$$P = \mathbf{T}.P$$

2 D.o.F

2D scale change

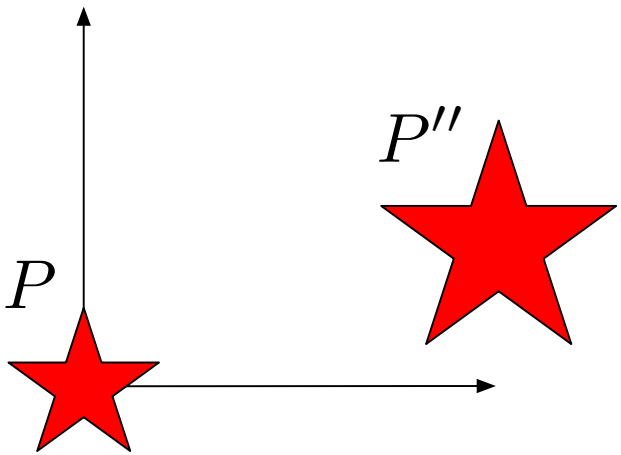


$$P = [x, y]^T \rightarrow [x, y, 1]^T$$
$$P' = [s_x.x, s_y.y]^T \rightarrow [s_x.x, s_y.y, 1]^T$$

$$P' = \begin{bmatrix} s_x.x \\ s_y.y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P' = \begin{bmatrix} \mathbf{s} & 0 \\ 0 & 1 \end{bmatrix} P = \mathbf{S}.P$$

2D scale change and translation



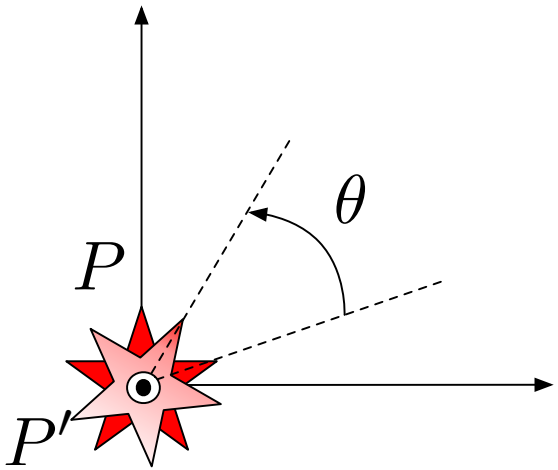
$$P'' = \mathbf{T}.P' = \mathbf{T}.\mathbf{S}.P$$

note : $\mathbf{T}.\mathbf{S}.P \neq \mathbf{S}.\mathbf{T}.P$

$$\mathbf{T}.\mathbf{S}.P = \begin{bmatrix} s_x.x + t_x \\ s_y.y + t_y \\ 1 \end{bmatrix} \text{ and}$$

$$\mathbf{S}.\mathbf{T}.P = \begin{bmatrix} s_x.x + s_x.t_x \\ s_y.y + s_y.t_y \\ 1 \end{bmatrix}$$

2D rotation



$$P' = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

1 D.o.F.

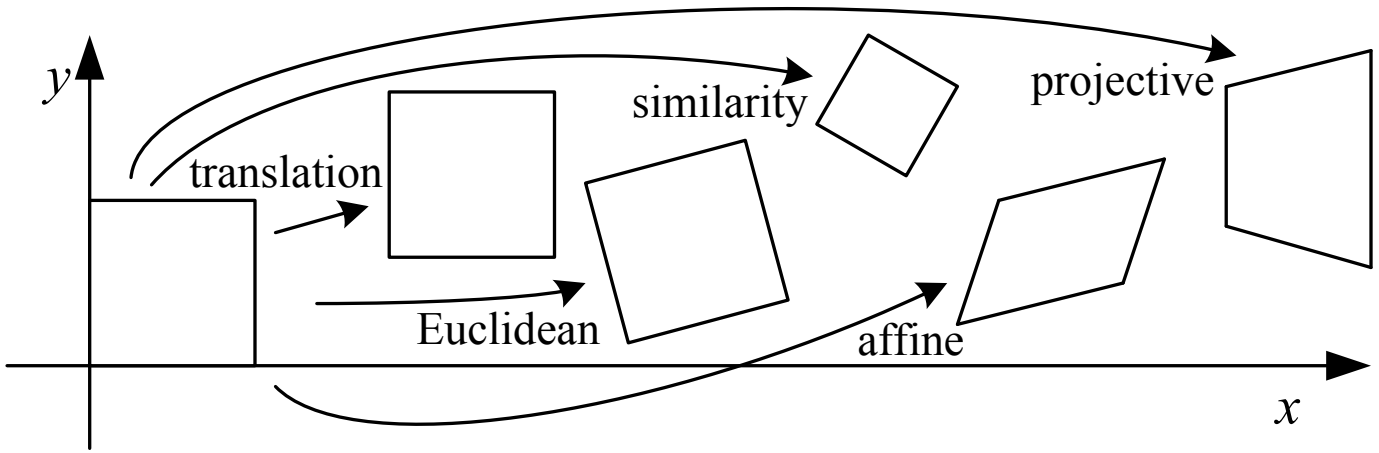
2D similarity

- Similarity = scale (with $s_x = s_y$)+ rotation + translation

$$P' = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

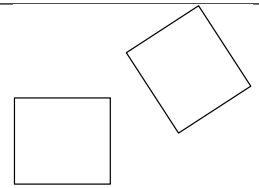
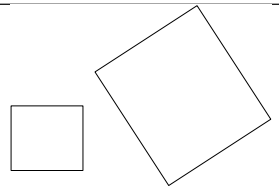
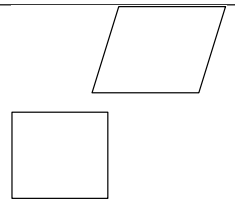
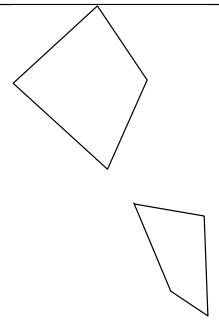
$$P' = \begin{bmatrix} \mathbf{RS} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D transformation summary



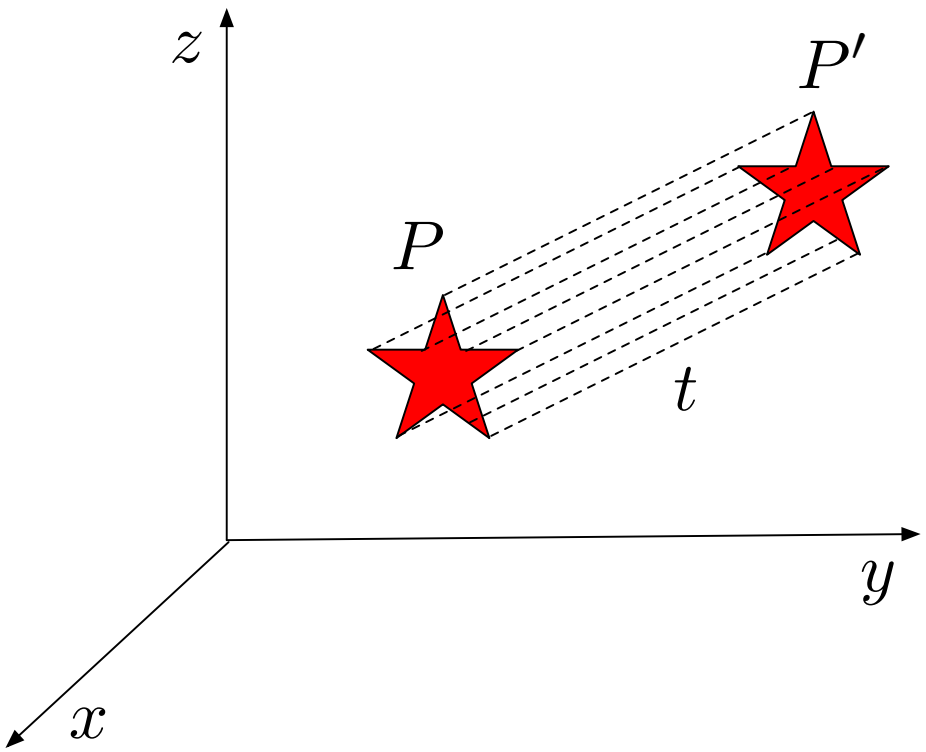
Extracted from [Sze10].

2D transformation summary

type	D.o.F.	matrix	transformed square	invariants
Euclidean	3	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		lengths, angles, parallelism, straight lines
Similarity	4	$\begin{bmatrix} s.r_{11} & s.r_{12} & t_x \\ s.r_{21} & s.r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		angles, parallelism, straight lines
Affine	6	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		parallelism, straight lines
Projective	8	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		straight lines

Adapted from [Sze10, HZ04]

3D translation



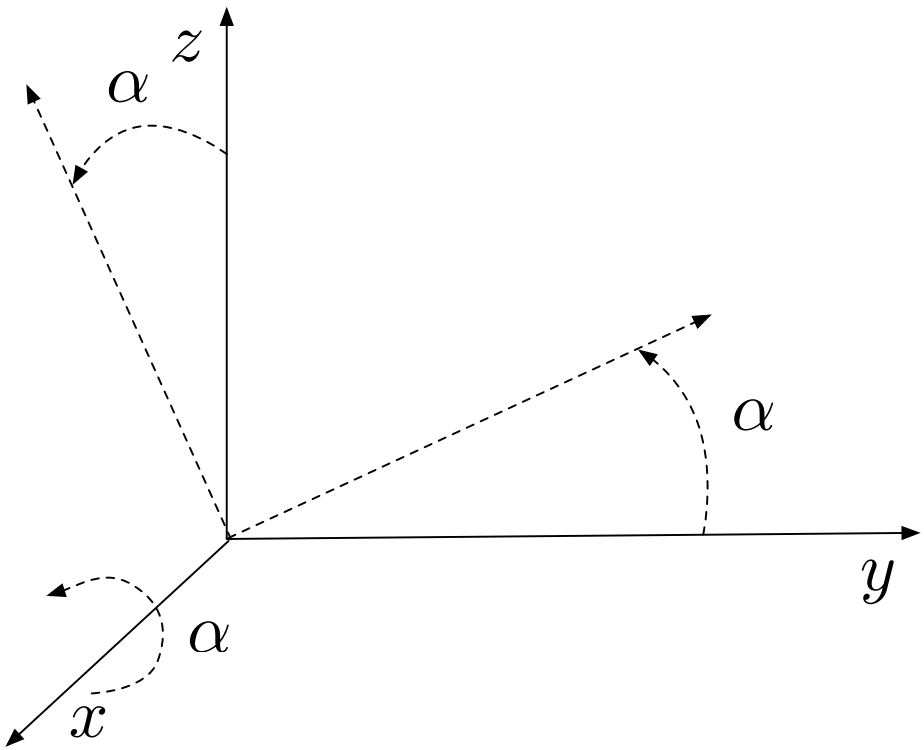
$$P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ and } t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$P' = \begin{bmatrix} 0 & \mathbf{t} \\ 0 & 1 \end{bmatrix}_{4\times 4} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3 D.o.F

3D rotation

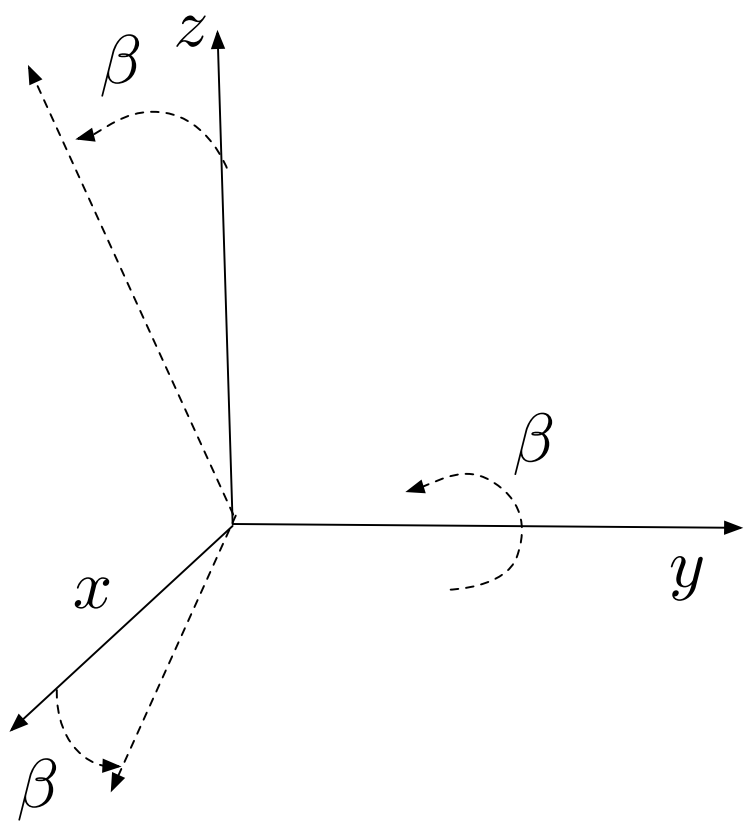
Counter-clockwise rotation around coordinate axes



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

3D rotation

Counter-clockwise rotation around coordinate axes

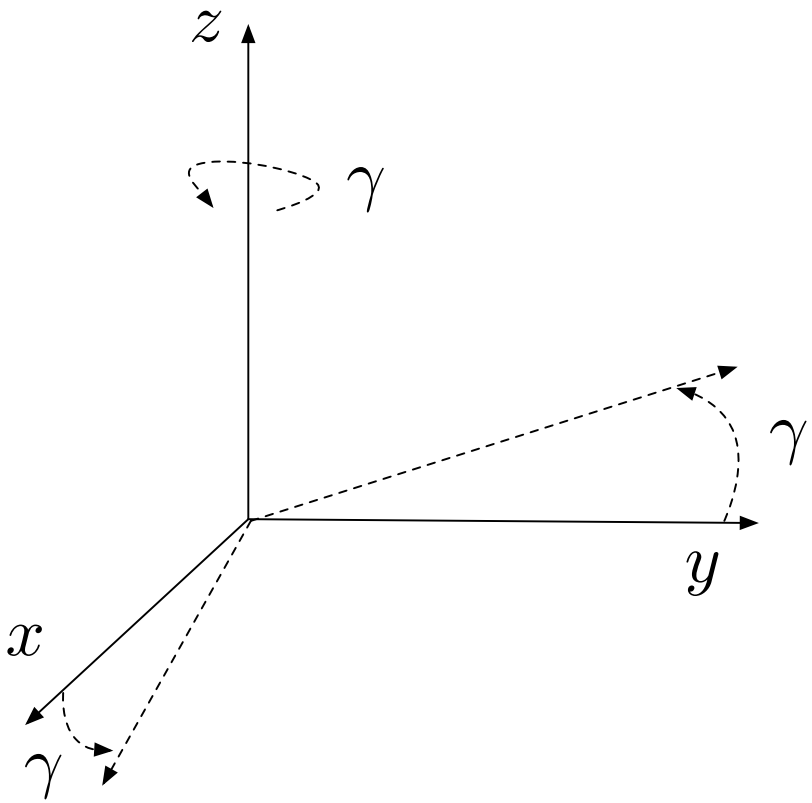


$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

3D rotation

Counter-clockwise rotation around coordinate axes



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3D rotation

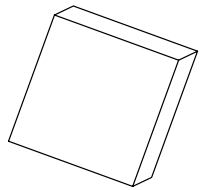
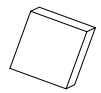
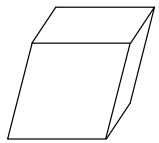
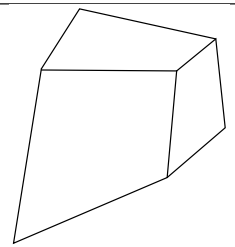
$$P' = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & 1 \end{bmatrix}_{4\times 4} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ with } \mathbf{R} = R_z(\gamma)R_y(\beta)R_x(\alpha) \text{ or}$$

$$\mathbf{R} = \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix}$$

$$\text{Simplified notation : } \mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\text{Reminder : } R^T R = RR^T = I$$

3D transformation summary

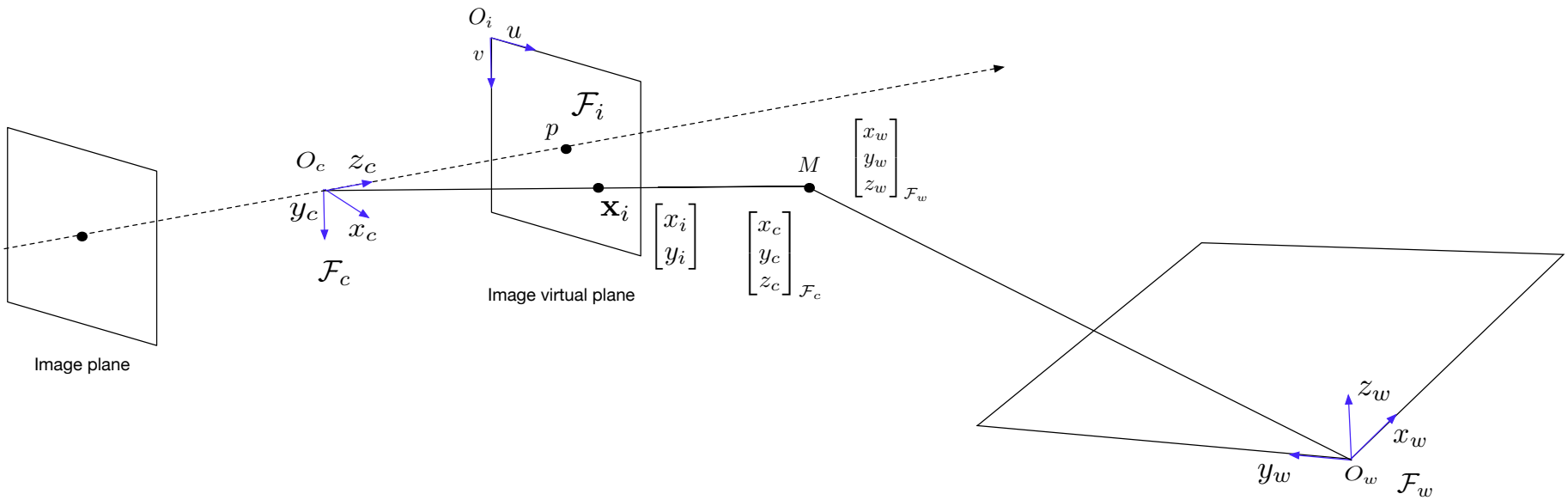
type	D.o.F.	matrix	transformed cube	invariants
Euclidean	6	$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$		lengths, angles, parallelism, straight lines
Similarity	7	$\begin{bmatrix} s.r_{11} & s.r_{12} & s.r_{13} & t_x \\ s.r_{21} & s.r_{22} & s.r_{23} & t_y \\ s.r_{31} & s.r_{32} & s.r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$		angles, parallelism, straight lines
Affine	12	$\begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$		parallelism, straight lines
Projective	15	$\begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$		straight lines

Adapted from [Sze10, HZ04]

Outline

Extrinsic parameters

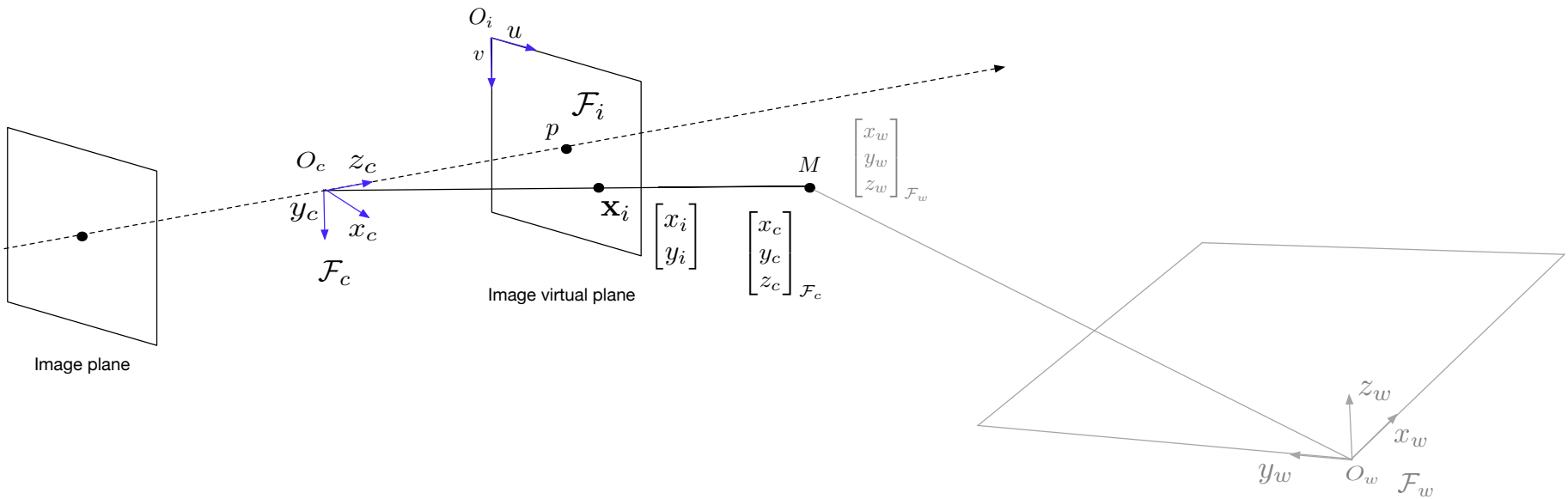
What we want:



$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \rightarrow \begin{bmatrix} u \\ v \end{bmatrix}$$

Extrinsic parameters

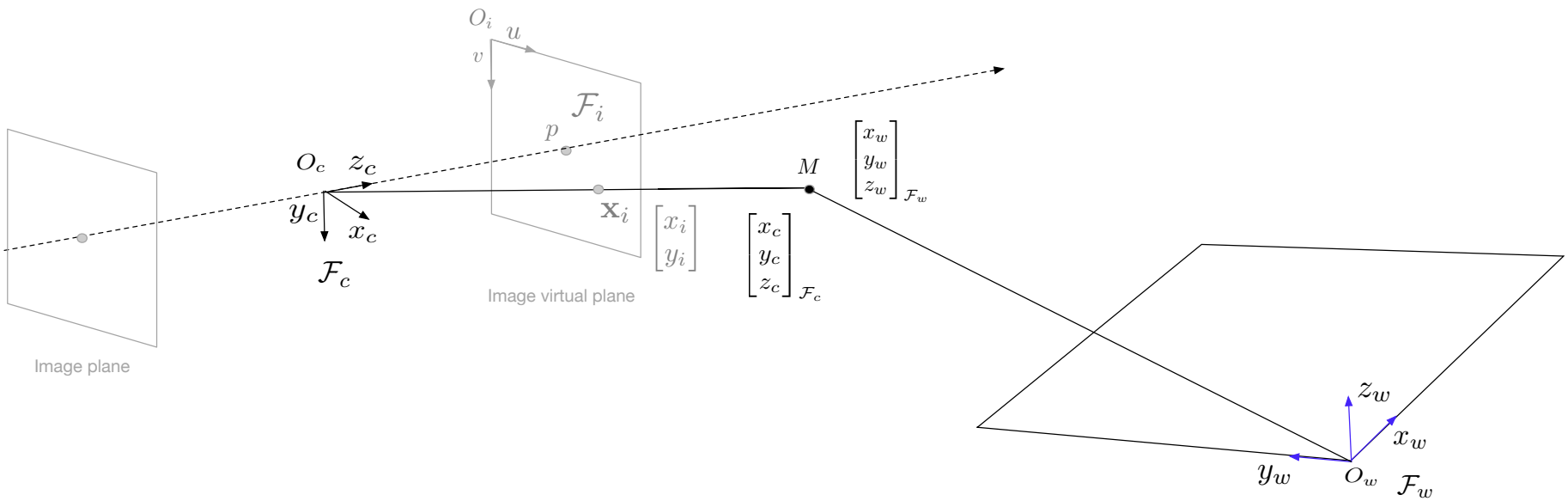
What we have:



$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

Extrinsic parameters

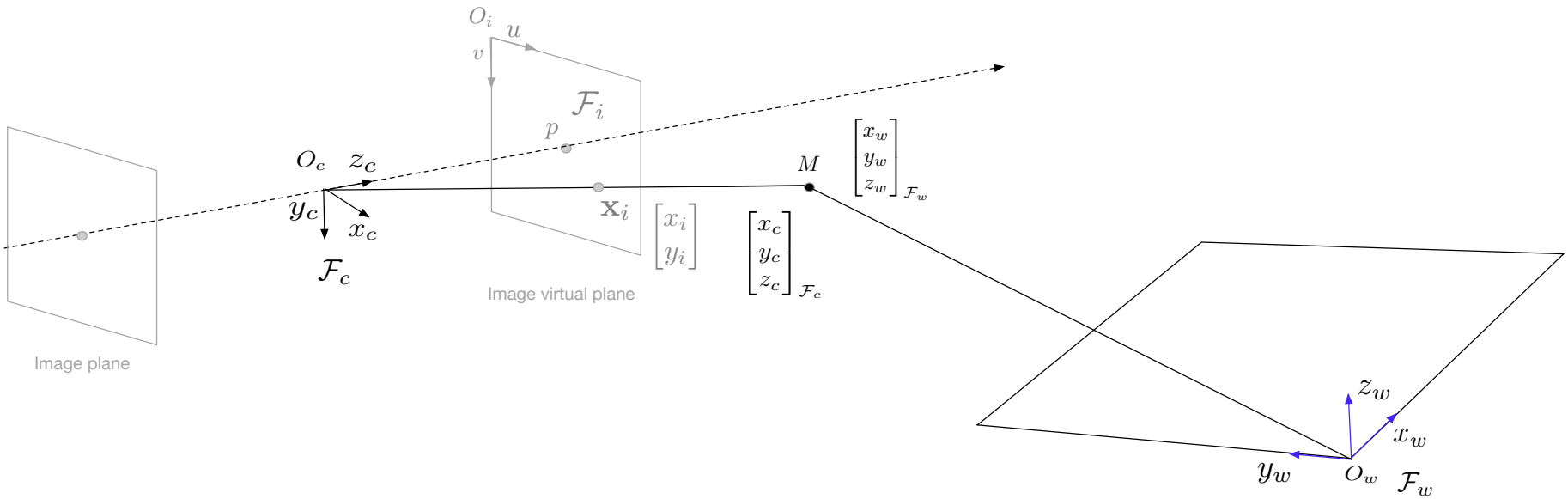
What is missing:



Position and orientation of the camera frame $\mathcal{F}_c$ in the world frame $\mathcal{F}_w$.

Extrinsic parameters

What is missing:

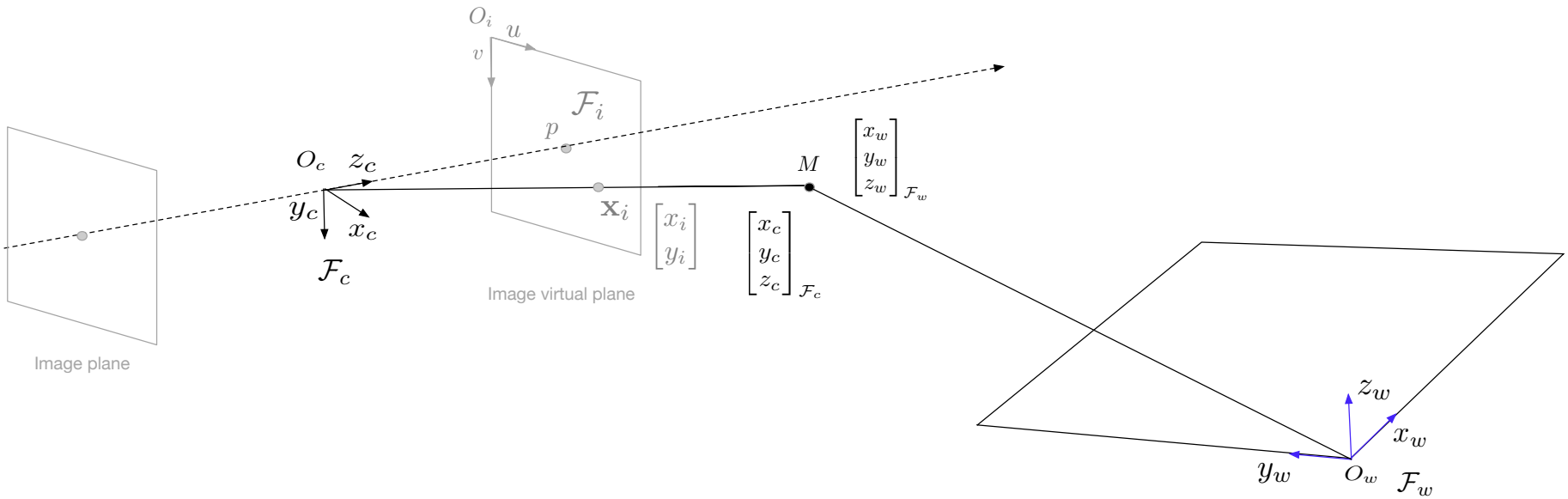


Position and orientation of the camera frame $\mathcal{F}_c$ in the world frame $\mathcal{F}_w$.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Extrinsic parameters

What is missing:



Position and orientation of the camera frame $\mathcal{F}_c$ in the world frame $\mathcal{F}_w$.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \text{ with } \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \text{ extrinsic parameters matrix.}$$

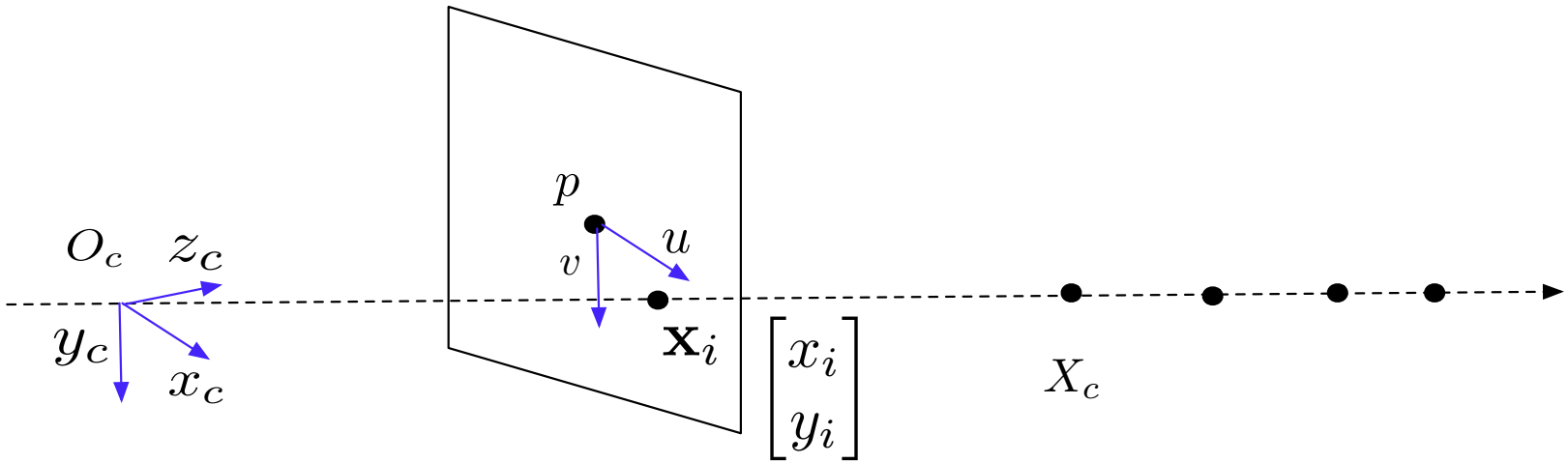
Outline

Camera matrix

Finally we get :

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

Camera matrix



All the 3D points on the same line passing through $\mathbf{x}_i$ have the same image point.

Camera matrix

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

- λ is a scale factor.
- $\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ is the camera matrix.
- $\mathbf{K}$ 5 D.o.F. and $\mathbf{R}$ 3 D.o.F. , $\mathbf{t}$ 3 D.o.F.
  $\Rightarrow$ 11 D.o.F. in total ($+1$ for the scale)

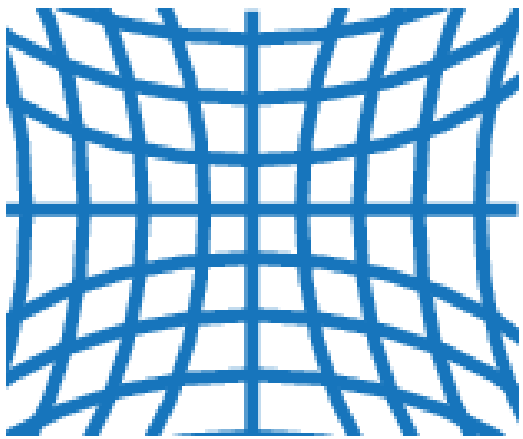
Outline

Lens distortions

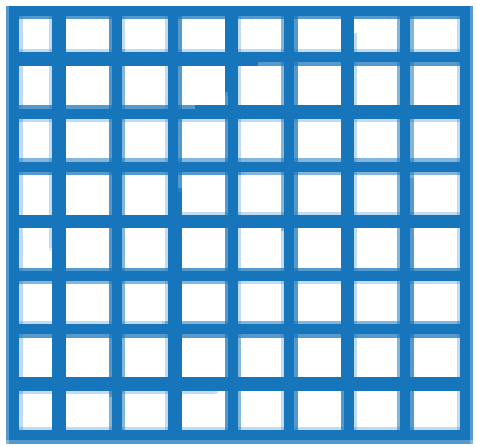
- Real-world cameras are not pinhole: use of lenses
- Lenses introduce distortions:
  - Radial distortions
  - Tangential distortions

©Photo credits Nicolas Antigny

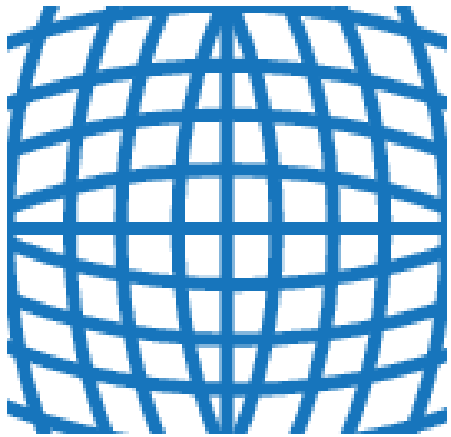
Lens distortions - Radial distortion



Negative radial distortion
"pincushion"



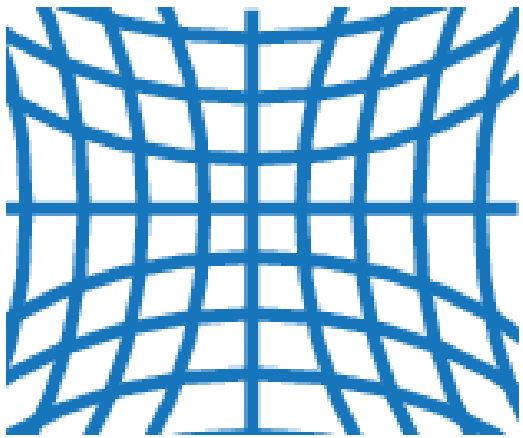
No distortion



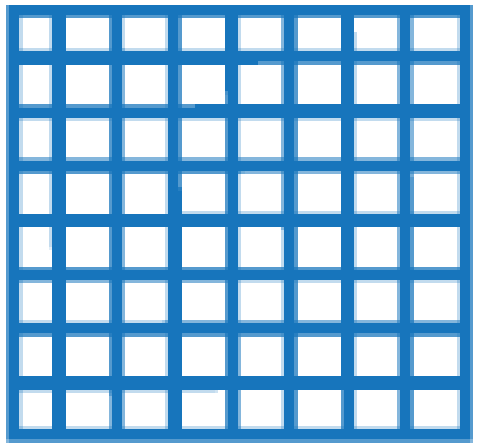
Positive radial distortion
"barrel"

©Mathworks source

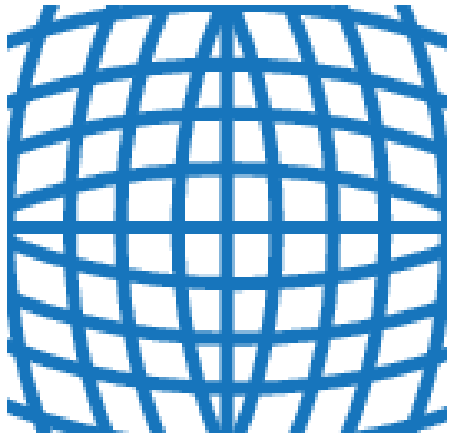
Lens distortions - Radial distortion



Negative radial distortion
"pincushion"



No distortion



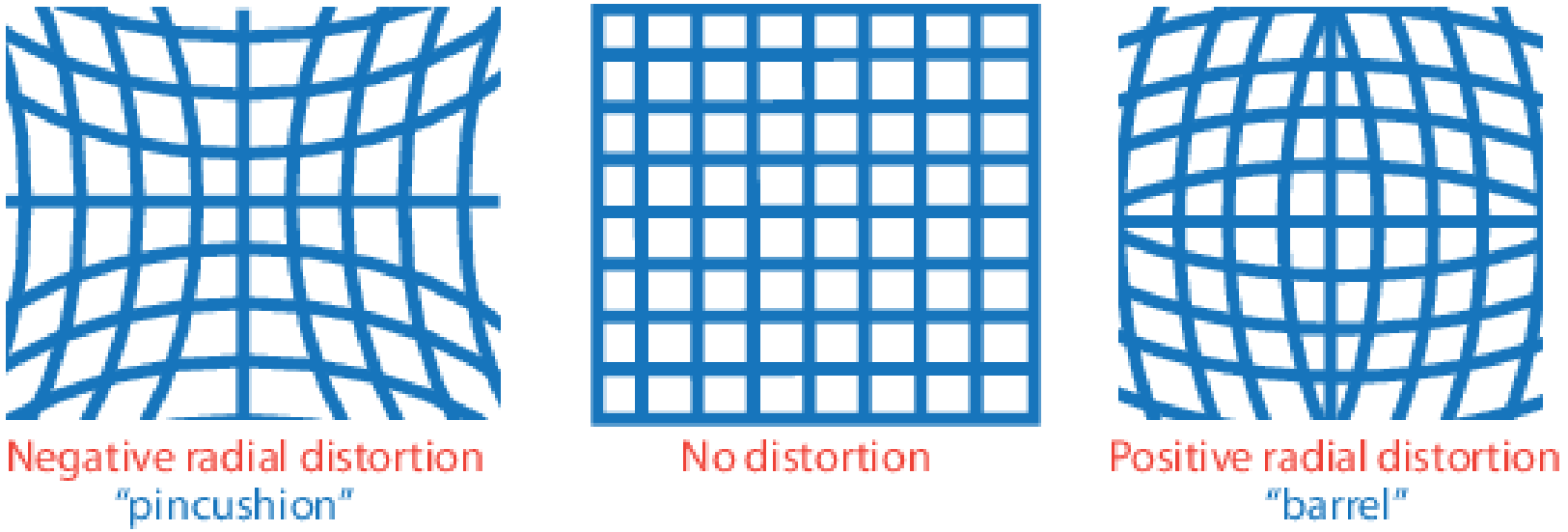
Positive radial distortion
"barrel"

©Mathworks source

$$x_{final} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$
$$y_{final} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

with k_1, k_2, k_3 distortion coefficients and r distance to optical center

Lens distortions - Radial distortion

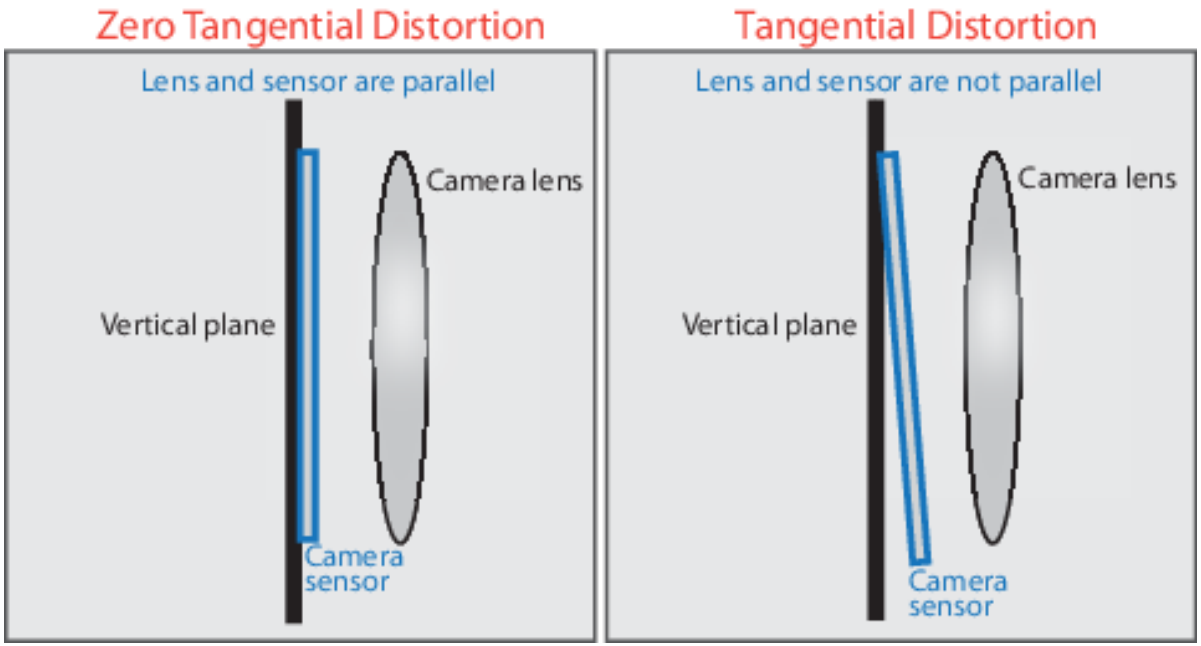


$$x_{final} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$
$$y_{final} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

with k_1, k_2, k_3 distortion coefficients and r distance to optical center

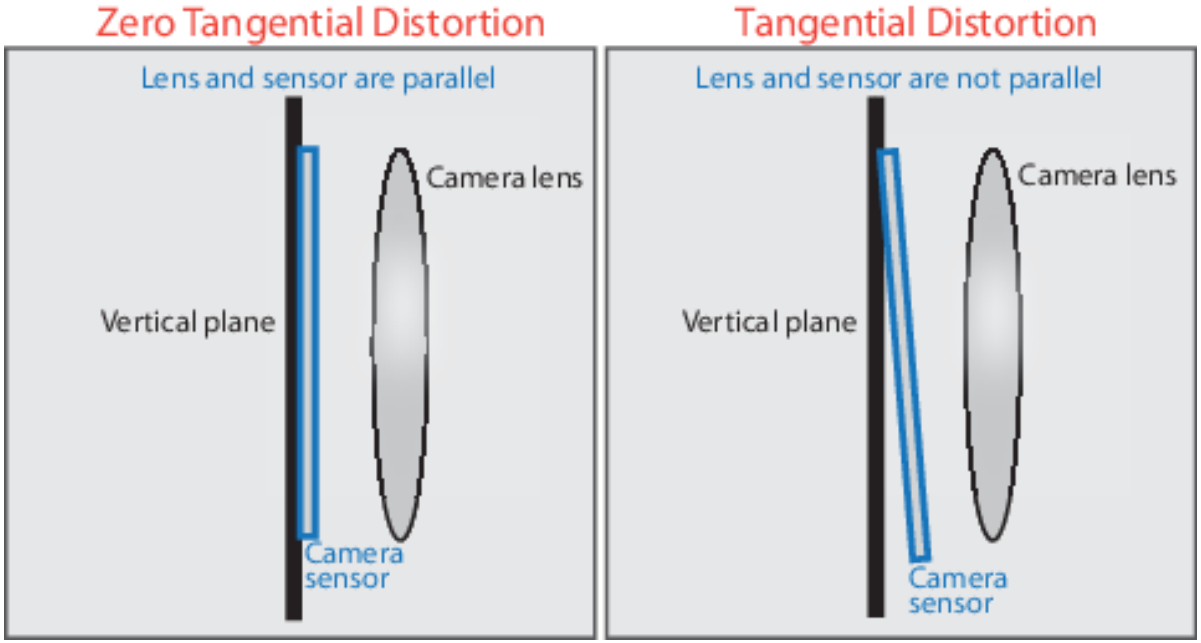
Note: This distortion is implemented after perspective projection but before the effect of the intrinsic parameters so the warping is relative to the optical axis and not the origin of the pixel coordinate system.

Lens distortions - Tangential distortions



©Mathworks source

Lens distortions - Tangential distortions



©Mathworks source

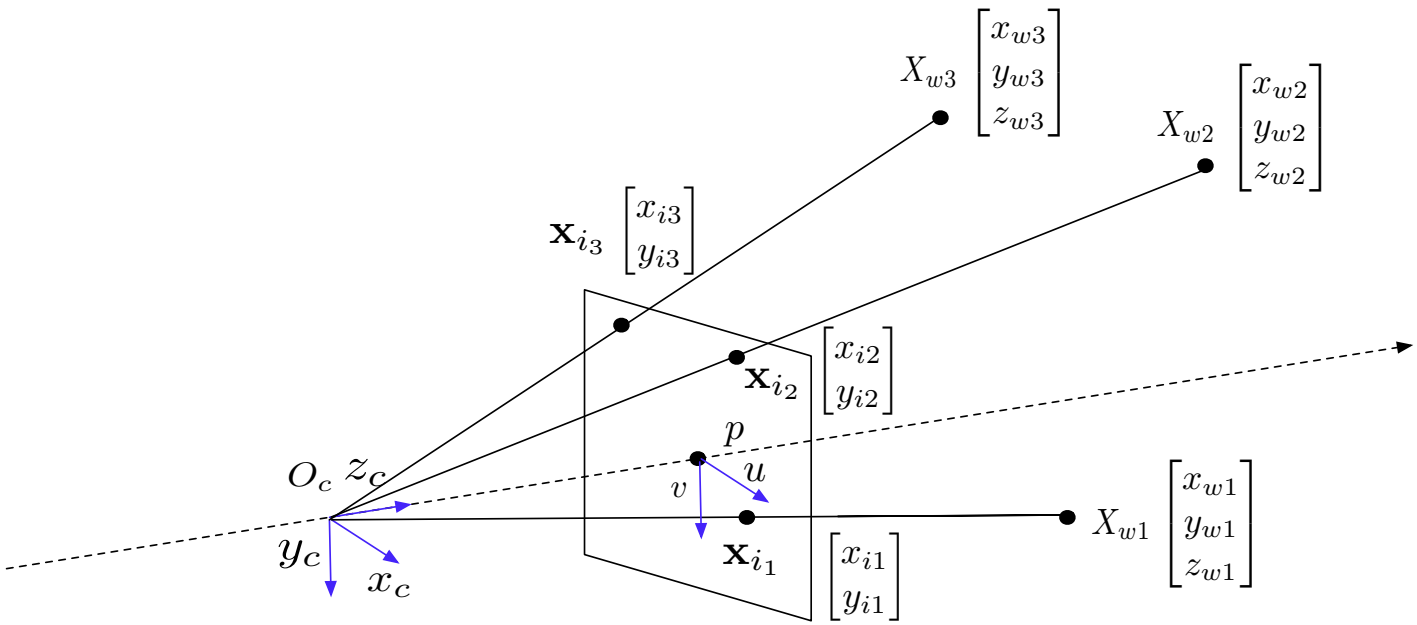
$$x_{final} = x + [2p_1y + p_2(r^2 + 2x^2)]$$
$$y_{final} = y + [p_1(r^2 + 2y^2) + 2p_2x]$$

with p_1, p_2 tangential distortion parameters.

Outline

Outline

Linear approach principle

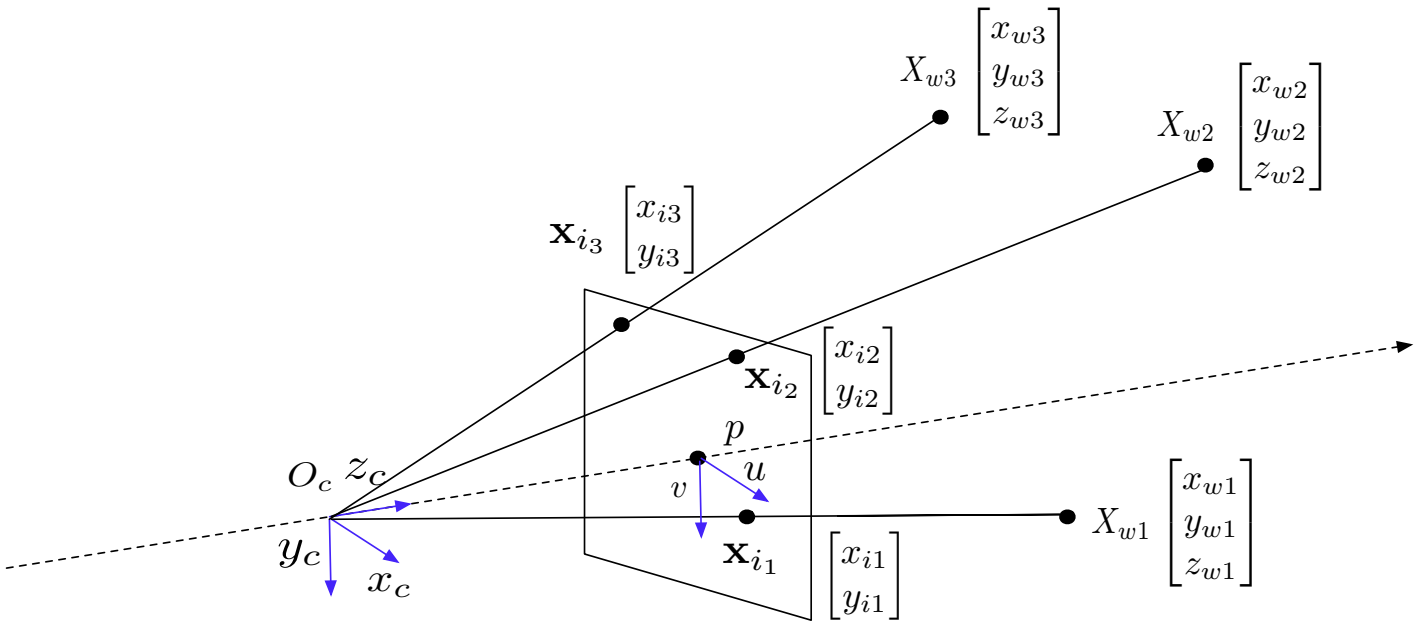


Knowing 3D points X_{w_i} and their corresponding points in the image $\mathbf{x}_{i_i}$

P-n-P: Perspective-*n*-point

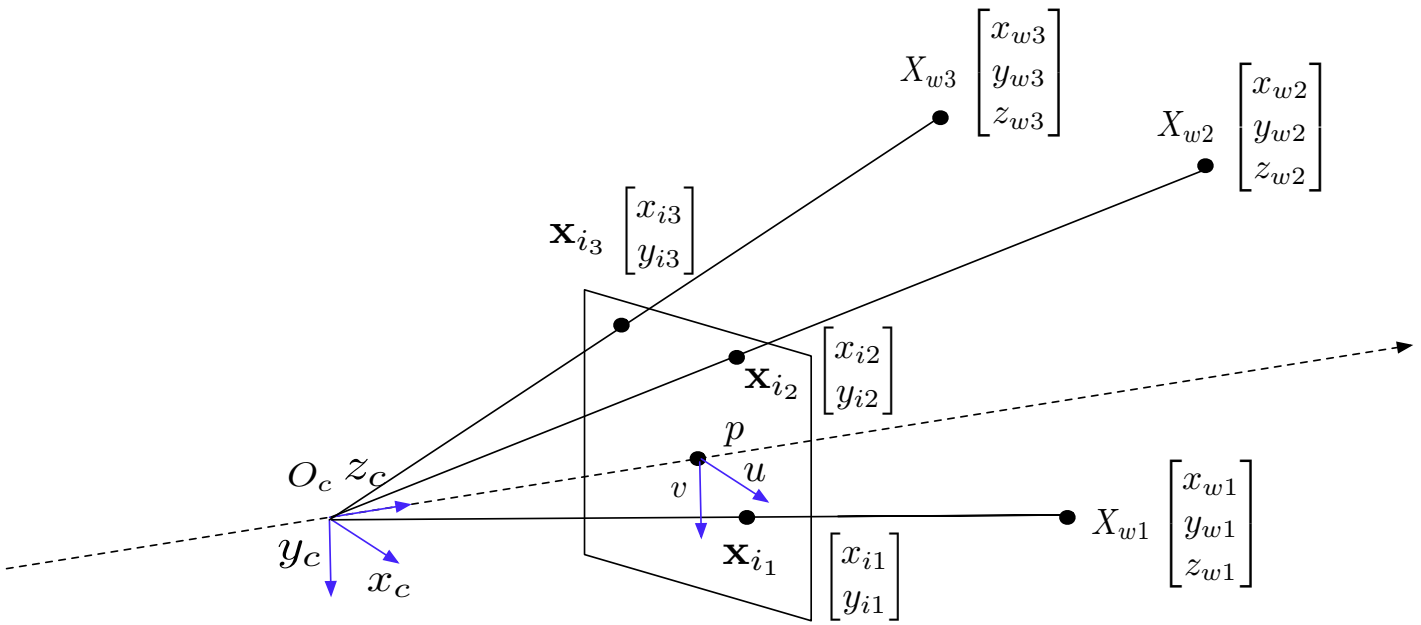
- Estimate **P**
- Extract **K**, **R** and **t** from **P**

Linear approach principle



How many corresponding points are needed?

Linear approach principle



How many corresponding points are needed?

- $\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{I}_{3\times3} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$ as 12 entries and 11 D.o.F. (ignoring the scale)
- each correspondence point leads to 2 eq.
  $\Rightarrow$ 5 correspondences + 1 eq. are needed: 6 correspondence points

Direct Linear Transform: $\mathbf{x}_i \leftrightarrow X_w$

- Considering $\mathbf{P}_{3\times4}$ and $\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ a point in the image

  corresponding to a 3D point $X_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$

Direct Linear Transform: $\mathbf{x}_i \leftrightarrow X_w$

- Considering $\mathbf{P}_{3\times4}$ and $\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ a point in the image corresponding to a 3D point $X_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$
- $\mathbf{x}_i = \mathbf{P}X_w$ for all corresponding points

Direct Linear Transform: $\mathbf{x}_i \leftrightarrow X_w$

- Considering $\mathbf{P}_{3\times4}$ and $\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ a point in the image

  corresponding to a 3D point $X_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$
- $\mathbf{x}_i = \mathbf{P}X_w$ for all corresponding points
- We have the linear relationship:

$$\mathbf{x}_i \times \mathbf{P}X_w = 0 \text{ (with } \times \text{ cross product)}$$

Outline

Direct Linear Transform: $\mathbf{x}_i \times \mathbf{P} X_w = 0$

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

- writing $\mathbf{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$ with P_i a 4-vector of the i-th row of $\mathbf{P}$

Direct Linear Transform: $\mathbf{x}_i \times \mathbf{P}X_w = 0$

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

- writing $\mathbf{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$ with P_i a 4-vector of the i-th row of $\mathbf{P}$
- then $\mathbf{P}X_w = \begin{bmatrix} P_1^T \cdot X_w \\ P_2^T \cdot X_w \\ P_3^T \cdot X_w \end{bmatrix}$

Direct Linear Transform: $\mathbf{x}_i \times \mathbf{P}X_w = 0$

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

- writing $\mathbf{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$ with P_i a 4-vector of the i-th row of $\mathbf{P}$
- then $\mathbf{P}X_w = \begin{bmatrix} P_1^T \cdot X_w \\ P_2^T \cdot X_w \\ P_3^T \cdot X_w \end{bmatrix}$
- and $\mathbf{x}_i \times \mathbf{P}X_w = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \times \begin{bmatrix} P_1^T \cdot X_w \\ P_2^T \cdot X_w \\ P_3^T \cdot X_w \end{bmatrix}$

Direct Linear Transform: $\mathbf{x}_i \times \mathbf{P}X_w = 0$

- $\mathbf{x}_i \times \mathbf{P}X_w = 0$

Direct Linear Transform: $\mathbf{x}_i \times \mathbf{P} X_w = 0$

- $\mathbf{x}_i \times \mathbf{P} X_w = 0$
- Developing it:

$$\begin{bmatrix} x_i P_2^T X_w - y_i P_1^T X_w \\ -x_i P_3^T X_w + z_i P_1^T X_w \\ y_i P_3^T X_w - z_i P_2^T X_w \end{bmatrix} = 0$$

Direct Linear Transform: $\mathbf{x}_i \times \mathbf{P} X_w = 0$

- $\mathbf{x}_i \times \mathbf{P} X_w = 0$
- Developing it:
$$\begin{bmatrix} x_i P_2^T X_w - y_i P_1^T X_w \\ -x_i P_3^T X_w + z_i P_1^T X_w \\ y_i P_3^T X_w - z_i P_2^T X_w \end{bmatrix} = 0$$
- As $P_i^T X_w = X_w^T P_i$:
$$\begin{bmatrix} x_i X_w^T P_2 - y_i X_w^T P_1 \\ -x_i X_w^T P_3 + z_i X_w^T P_1 \\ y_i X_w^T P_3 - z_i X_w^T P_2 \end{bmatrix} = 0$$

Direct Linear Transform: $\mathbf{x}_i \times \mathbf{P}X_w = 0$

- $\mathbf{x}_i \times \mathbf{P}X_w = 0$
- Developing it:
$$\begin{bmatrix} x_i P_2^T X_w - y_i P_1^T X_w \\ -x_i P_3^T X_w + z_i P_1^T X_w \\ y_i P_3^T X_w - z_i P_2^T X_w \end{bmatrix} = 0$$
- As $P_i^T X_w = X_w^T P_i$:
$$\begin{bmatrix} x_i X_w^T P_2 - y_i X_w^T P_1 \\ -x_i X_w^T P_3 + z_i X_w^T P_1 \\ y_i X_w^T P_3 - z_i X_w^T P_2 \end{bmatrix} = 0$$
- and writing in a matrix form:
$$\begin{bmatrix} 0 & -z_i X_w^T & y_i X_w^T \\ z_i X_w^T & 0 & -x_i X_w^T \\ -y_i X_w^T & x_i X_w^T & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$

Direct Linear Transform: determination of **P**

- the 3 rows are linearly dependent

$$\begin{bmatrix} 0 & -z_i X_w^T & y_i X_w^T \\ z_i X_w^T & 0 & -x_i X_w^T \\ -y_i X_w^T & x_i X_w^T & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$

Direct Linear Transform: determination of **P**

- the 3 rows are linearly dependent

$$\begin{bmatrix} 0 & -z_i X_w^T & y_i X_w^T \\ z_i X_w^T & 0 & -x_i X_w^T \\ -y_i X_w^T & x_i X_w^T & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$

- So we choose only the 2 first ones

$$\begin{bmatrix} 0 & -z_i X_w^T & y_i X_w^T \\ z_i X_w^T & 0 & -x_i X_w^T \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$

Direct Linear Transform: determination of **P**

- the 3 rows are linearly dependent

$$\begin{bmatrix} 0 & -z_i X_w^T & y_i X_w^T \\ z_i X_w^T & 0 & -x_i X_w^T \\ -y_i X_w^T & x_i X_w^T & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$

- So we choose only the 2 first ones

$$\begin{bmatrix} 0 & -z_i X_w^T & y_i X_w^T \\ z_i X_w^T & 0 & -x_i X_w^T \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$

- this will be written
  $A_i \mathbf{p} = 0$ with $\mathbf{p}$ a 12-vector made up of the entries of $\mathbf{P}$.

Direct Linear Transform: determination of **P**

- the 3 rows are linearly dependent
$$\begin{bmatrix} 0 & -z_i X_w^T & y_i X_w^T \\ z_i X_w^T & 0 & -x_i X_w^T \\ -y_i X_w^T & x_i X_w^T & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$
- So we choose only the 2 first ones
$$\begin{bmatrix} 0 & -z_i X_w^T & y_i X_w^T \\ z_i X_w^T & 0 & -x_i X_w^T \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0$$
- this will be written
$A_i \mathbf{p} = 0$ with $\mathbf{p}$ a 12-vector made up of the entries of $\mathbf{P}$.
- note: The eq. hold for any homogeneous representation of $\mathbf{x}_i$.
Choosing $z_i = 1$, then $\begin{bmatrix} x_i \\ y_i \end{bmatrix}$ are coordinate on the image

Direct Linear Transform: determination of **P**

- Staking all eq. for each correspondence point leads to: $A\mathbf{p} = 0$ with A matrix of eq. coefficients built from the matrix rows A_i and $\mathbf{p}$ and 12-vector made up of the entries of the matrix $\mathbf{P}$

Direct Linear Transform: determination of **P**

- Staking all eq. for each correspondence point leads to:
  $A\mathbf{p} = 0$ with A matrix of eq. coefficients built from the matrix rows A_i and $\mathbf{p}$ and 12-vector made up of the entries of the matrix $\mathbf{P}$
- Obtain the SVD of A. The unit singular vector corresponding to the smallest singular value is the solution of $\mathbf{p}$.
  Specifically, if $A = UDV^T$ with D diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{p}$ is the last column of V.

Direct Linear Transform: determination of **P**

- Staking all eq. for each correspondence point leads to: $A\mathbf{p} = 0$ with A matrix of eq. coefficients built from the matrix rows A_i and $\mathbf{p}$ and 12-vector made up of the entries of the matrix $\mathbf{P}$
- Obtain the SVD of A. The unit singular vector corresponding to the smallest singular value is the solution of $\mathbf{p}$.
  Specifically, if $A = UDV^T$ with D diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{p}$ is the last column of V.
- $\mathbf{P}$ is then given by $\mathbf{p}$

[DLT algorithm from [HZ04]].

Direct Linear Transform: determination of **P**

- Minimal solution
  Given 6 correspondences, the solution is exact.
  The solution is obtained solving $A\mathbf{p} = 0$ where A is 11×12.
  In general A will have rank 11, and the solution vector $\mathbf{p}$ is the 1-dimensional right null-space of A.

Direct Linear Transform: determination of **P**

- Minimal solution
  Given 6 correspondences, the solution is exact.
  The solution is obtained solving $A\mathbf{p} = 0$ where A is 11×12.
  In general A will have rank 11, and the solution vector $\mathbf{p}$ is the 1-dimensional right null-space of A.
- Over-determined solution
  If data are not exact (noise) and $n \geq 6 \rightarrow$ not an exact solution to $A\mathbf{b} = 0$.
  Then the estimation of **P** may be obtained by minimizing an algebraic error with the normalization constraint $||\mathbf{p}|| = 1$

[From [HZ04]].

DLT: points normalization

- DLT is not invariant to similarity transformations (details in [HZ04])
  $\Rightarrow$ apply a normalizing transformation to the data before applying the DLT algorithm

DLT: points normalization

- DLT is not invariant to similarity transformations (details in [HZ04])
  $\Rightarrow$ apply a normalizing transformation to the data before applying the DLT algorithm
- Normalization for 2D points: Isotropic scaling
  - points are translated so that their centroid is at the origin.
  - points are then scaled so that their root-mean-square distance from the origin is equal to $\sqrt{2}$

DLT: points normalization

- DLT is not invariant to similarity transformations (details in [HZ04])
  $\Rightarrow$ apply a normalizing transformation to the data before applying the DLT algorithm
- Normalization for 2D points: Isotropic scaling
  - points are translated so that their centroid is at the origin.
  - points are then scaled so that their root-mean-square distance from the origin is equal to $\sqrt{2}$
- Normalization for 3D points - case with small variations in points depth
  - centroid of the points is translated to the origin
  - coordinates of the points are scaled so that their root-mean-square distance from the origin is $\sqrt{3}$

Direct Linear Transform complete algorithm [HZ04]

Objective:

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the camera projection matrix $\mathbf{P}$ such as $\mathbf{x}_i = \mathbf{P}X_w$

Algorithm

1. **Normalization of $\mathbf{x}_i$**: Compute a similarity transformation T, consisting of a translation and scaling, that takes points $\mathbf{x}_i$ to a new set of points $\tilde{\mathbf{x}}_i$ such that the centroid of the points $\tilde{\mathbf{x}}_i$ is the coordinate origin $[0, 0]^T$, and their average distance from the origin is $\sqrt{2}$.

Direct Linear Transform complete algorithm [HZ04]

Objective:

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the camera projection matrix $\mathbf{P}$ such as $\mathbf{x}_i = \mathbf{P}X_w$

Algorithm

1. **Normalization of $\mathbf{x}_i$**: Compute a similarity transformation T
2. **Normalization of X_w**: Compute a similarity transformation U, consisting of a translation and scaling, that takes points X_w to a new set of points $\tilde{X}_w$ such that the centroid of the points $\tilde{X}_w$ is the coordinate origin $[0, 0, 0]^T$, and their average distance from the origin is $\sqrt{3}$.

Direct Linear Transform complete algorithm [HZ04]

Objective:

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the camera projection matrix $\mathbf{P}$ such as $\mathbf{x}_i = \mathbf{P}X_w$

Algorithm

1. **Normalization of $\mathbf{x}_i$**: Compute a similarity transformation T
2. **Normalization of X_w**: Compute a similarity transformation U

Direct Linear Transform complete algorithm [HZ04]

Objective:

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the camera projection matrix $\mathbf{P}$ such as $\mathbf{x}_i = \mathbf{P}X_w$

Algorithm

1. **Normalization of $\mathbf{x}_i$**: Compute a similarity transformation T
2. **Normalization of X_w**: Compute a similarity transformation U
3. **DLT**:
   1. For each correspondence $\tilde{X}_w \leftrightarrow \tilde{\mathbf{x}}_i$ compute the matrix A_i. Only the first two rows need be used in general.
   2. Form the $2n \times 12$ matrix A by stacking the equations generated by each correspondence $\tilde{X}_w \leftrightarrow \tilde{\mathbf{x}}_i$.
   3. Write $\mathbf{p}$ for the vector containing the entries of the matrix $\tilde{\mathbf{P}}$. A solution of $A\mathbf{p} = 0$, subject to $||p|| = 1$, is obtained from the unit singular vector of A corresponding to the smallest singular value.

Direct Linear Transform complete algorithm [HZ04]

Objective:

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the camera projection matrix $\mathbf{P}$ such as $\mathbf{x}_i = \mathbf{P}X_w$

Algorithm

1. **Normalization of $\mathbf{x}_i$**: Compute a similarity transformation T
2. **Normalization of X_w**: Compute a similarity transformation U
3. **DLT**: Compute $\mathbf{p}$ from A then get $\tilde{\mathbf{P}}$
4. **Denormalization**. The camera matrix for the original (unnormalized) coordinates is obtained from $\tilde{\mathbf{P}}$ as :

$$\mathbf{P} = T^{-1}\tilde{\mathbf{P}}U$$

Direct Linear Transform complete algorithm [HZ04]

Objective:

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the camera projection matrix $\mathbf{P}$ such as $\mathbf{x}_i = \mathbf{P}X_w$

Algorithm

1. **Normalization of $\mathbf{x}_i$**: Compute a similarity transformation T
2. **Normalization of X_w**: Compute a similarity transformation U
3. **DLT**: Compute $\mathbf{p}$ from A then get $\tilde{\mathbf{P}}$
4. **Denormalization**. The camera matrix for the original (unnormalized) coordinates is obtained from $\tilde{\mathbf{P}}$ as :

$$\mathbf{P} = T^{-1}\tilde{\mathbf{P}}U$$

Note: an implementation for leaning purposes can be found at

- http://people.rennes.inria.fr/Eric.Marchand/pose-estimation/tutorial-pose-dlt-opencv.html

Outline

Extraction of the extrinsic and intrinsic parameters

Having $\mathbf{P}$:

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{I}_{3\times 3} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

We can write:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

Then:

$$\mathbf{P} = \begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

[From [Mar13]]

Extraction of the extrinsic and intrinsic parameters

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} =$$

$$\begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r31 & f_y r_{22} + v_0 r32 & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

leads to:

- $r_{3i} = \frac{p_{3i}}{||p_{3i}||}$ for $i = 1, 2, 3$
- $t_3 = p_{34}$

Case $s = 0$: get intrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

[From [Mar13]]

Case $s = 0$: get intrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$u_0 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

[From [Mar13]]

Case $s = 0$: get intrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$u_0 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

$$v_0 = \begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

[From [Mar13]]

Case $s = 0$: get intrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$u_0 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

$$f_x = \left\| \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} - u_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right\|$$

$$v_0 = \begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

[From [Mar13]]

Case $s = 0$: get intrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$u_0 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

$$f_x = \left\| \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} - u_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right\|$$

$$v_0 = \begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

$$f_y = \left\| \begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} - v_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right\|$$

[From [Mar13]]

Case $s = 0$: get extrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

[From [Mar13]]

Case $s = 0$: get extrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$\begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \end{bmatrix} = \left[\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} - u_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_x}$$

[From [Mar13]]

Case $s = 0$: get extrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$\begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \end{bmatrix} = \left[\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} - u_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_x}$$

$$\begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix} = \left[\begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} - v_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_y}$$

[From [Mar13]]

Case $s = 0$: get extrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$\begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \end{bmatrix} = \left[\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} - u_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_x}$$

$$\begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix} = \left[\begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} - v_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_y}$$

$$t_1 = \frac{p_{14} - u_0 \cdot t_3}{f_x}$$

[From [Mar13]]

Case $s = 0$: get extrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$\begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \end{bmatrix} = \left[\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} - u_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_x}$$

$$\begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix} = \left[\begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} - v_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_y}$$

$$t_1 = \frac{p_{14} - u_0 \cdot t_3}{f_x}$$

$$t_2 = \frac{p_{24} - v_0 \cdot t_3}{f_y}$$

[From [Mar13]]

Case $s = 0$: get extrinsic parameters

$$\begin{bmatrix} f_x r_{11} + u_0 r_{31} & f_x r_{12} + u_0 r_{32} & f_x r_{13} + u_0 r_{33} & f_x t_1 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r_{33} & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$\begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \end{bmatrix} = \left[\begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} - u_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_x}$$

$$\begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix} = \left[\begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} - v_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_y}$$

$$t_1 = \frac{p_{14} - u_0 \cdot t_3}{f_x}$$

$$t_2 = \frac{p_{24} - v_0 \cdot t_3}{f_y}$$

Notes: we must ensure $||r_i|| = 1$ with r_i i-th column of R and we are not sure that $r_3 = r_1 \times r_2$

[From [Mar13]]

Case $s \neq 0$

$$P = \begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$u_0 = \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

$$f_x = \left\| \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \end{bmatrix} - u_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right\|$$

$$v_0 = \begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

$$\begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix} = \left[\begin{bmatrix} p_{21} \\ p_{22} \\ p_{23} \end{bmatrix} - v_0 \cdot \begin{bmatrix} p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} \right] \cdot \frac{1}{f_y}$$

and we still have r_3 and t_3.

[From [Mar13]]

Case $s \neq 0$

$$P = \begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

[From [Mar13]]

Case $s \neq 0$

$$P = \begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$r_2 = \frac{r_2}{||r_2||}$$
$$r_1 = r_2 \times r_3$$

[From [Mar13]]

Case $s \neq 0$

$$P = \begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$r_2 = \frac{r_2}{||r_2||}$$
$$r_1 = r_2 \times r_3$$

$$t_2 = \frac{p_{24} - v_0 \cdot t_3}{f_y}$$
$$t_1 = \frac{p_{14} - u_0 \cdot t_3 - s \cdot t_2}{f_x}$$

[From [Mar13]]

Case $s \neq 0$

$$P = \begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$r_2 = \frac{r_2}{||r_2||}$$
$$r_1 = r_2 \times r_3$$

$$t_2 = \frac{p_{24} - v_0 \cdot t_3}{f_y}$$
$$t_1 = \frac{p_{14} - u_0 \cdot t_3 - s \cdot t_2}{f_x}$$

$$f_x r_{11} + s r_{21} = p_{11} - u_0 r_{31}$$
$$f_x r_{12} + s r_{22} = p_{12} - u_0 r_{32}$$

[From [Mar13]]

Case $s \neq 0$

$$P = \begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$r_2 = \frac{r_2}{||r_2||}$$
$$r_1 = r_2 \times r_3$$

$$t_2 = \frac{p_{24} - v_0 \cdot t_3}{f_y}$$
$$t_1 = \frac{p_{14} - u_0 \cdot t_3 - s \cdot t_2}{f_x}$$

$$\begin{bmatrix} r_{11} & r_{21} \\ r_{12} & r_{22} \end{bmatrix} \cdot \begin{bmatrix} f_x \\ s \end{bmatrix} = \begin{bmatrix} p_{11} - u_0 r_{31} \\ p_{12} - u_0 r_{32} \end{bmatrix}$$

[From [Mar13]]

Case $s \neq 0$

$$P = \begin{bmatrix} f_x r_{11} + s r_{21} + u_0 r_{31} & f_x r_{12} + s r_{22} + u_0 r_{32} & f_x r_{13} + s r_{23} + u_0 r_{33} & f_x t_1 + s t_2 + u_0 t_3 \\ f_y r_{21} + v_0 r_{31} & f_y r_{22} + v_0 r_{32} & f_y r_{23} + v_0 r33 & f_y t_2 + v_0 t_3 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$

$$r_2 = \frac{r_2}{||r_2||}$$
$$r_1 = r_2 \times r_3$$

$$t_2 = \frac{p_{24} - v_0 \cdot t_3}{f_y}$$
$$t_1 = \frac{p_{14} - u_0 \cdot t_3 - s \cdot t_2}{f_x}$$

$$\begin{bmatrix} f_x \\ s \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} \\ r_{12} & r_{22} \end{bmatrix}^{-1} \cdot \begin{bmatrix} p_{11} - u_0 r_{31} \\ p_{12} - u_0 r_{32} \end{bmatrix}$$

[From [Mar13]]

Outline

Non linear approach principle

Consider the projection of a 3D point $X_w = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$ onto a 2D point

$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ with $z_i = f$ (the focal length):

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \lambda_i \left[\mathbf{R} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \mathbf{t} \right]$$

with :

- $\mathbf{R}$ the rotation matrix
- $\mathbf{t}$ the translation vector
- λ_i a scale factor

Non linear approach principle

Suppressing the scale factor, we can write:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f\,\frac{r_{11}x_w + r_{12}y_w + r_{13}z_w + t_1}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_3} \\ f\,\frac{r_{21}x_w + r_{22}y_w + r_{23}z_w + t_2}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_3} \end{bmatrix}$$

Non linear approach principle

Suppressing the scale factor, we can write:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f\frac{r_{11}x_w+r_{12}y_w+r_{13}z_w+t_1}{r_{31}x_w+r_{32}y_w+r_{33}z_w+t_3} \\ f\frac{r_{21}x_w+r_{22}y_w+r_{23}z_w+t_2}{r_{31}x_w+r_{32}y_w+r_{33}z_w+t_3} \end{bmatrix}$$

Expressing $\begin{bmatrix} x_i \\ y_i \end{bmatrix}$ in pixels coordinates:

$$x_i = \frac{(u+e_x-u_0)}{s_x} - d_{0_x}$$
$$y_i = \frac{(v+e_y-v_0)}{s_y} - d_{0_y}$$

- e_x, e_y: measurement errors on x_i, y_i
- s_x, s_y: imager size along x and y
- d_{0_x}, d_{0_y}: distortions

[From [Mar13]]

Non linear approach principle

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f\frac{r_{11}x_w+r_{12}y_w+r_{13}z_w+t_1}{r_{31}x_w+r_{32}y_w+r_{33}z_w+t_3} \\ f\frac{r_{21}x_w+r_{22}y_w+r_{23}z_w+t_2}{r_{31}x_w+r_{32}y_w+r_{33}z_w+t_3} \end{bmatrix} = \begin{bmatrix} \frac{(u+e_x-u_0)}{s_x} - d_{0_x} \\ \frac{(v+e_y-v_0)}{s_y} - d_{0_y} \end{bmatrix}$$

We can express:

$$\begin{bmatrix} u+e_x \\ v+e_y \end{bmatrix} = \begin{bmatrix} P(\Phi) \\ Q(\Phi) \end{bmatrix}$$

with $\Phi = [u_0, v_0, k_1, k_2, k_3, p_1, p_2, f_x, f_y, t_1, t_2, t_3, \alpha, \beta, \gamma]^T$ a 15 parameters vector where $(\alpha, \beta, \gamma)^T$ parametrize the orientation.

[From [Mar13]]

Non linear approach principle

Then: $\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} P(\Phi) - u \\ Q(\Phi) - v \end{bmatrix} \Rightarrow V(\Phi) = \begin{bmatrix} e_x \\ e_y \end{bmatrix}$

We must find Φ which minimizes the reprojection error S:

- if 1 image and n correspondence points:

$$S = \sum_{i=1}^{n} (e_{x_i}^2 + e_{y_i}^2)$$

- if m image and n correspondence points on each image:

$$S = \sum_{j=1}^{m} \sum_{i=1}^{n} (e_{x_{ij}}^2 + e_{y_{ij}}^2)$$

[From [Mar13]]

Non linear approach principle

Then: $\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} P(\Phi) - u \\ Q(\Phi) - v \end{bmatrix} \Rightarrow V(\Phi) = \begin{bmatrix} e_x \\ e_y \end{bmatrix}$

We must find Φ which minimizes the reprojection error S:

- if 1 image and n correspondence points: $S = \sum_{i=1}^{n} (e_{x_i}^2 + e_{y_i}^2)$
- if m image and n correspondence points on each image: $S = \sum_{j=1}^{m} \sum_{i=1}^{n} (e_{x_{ij}}^2 + e_{y_{ij}}^2)$

Non-linear optimization problem!

That can be solved using algorithm as Gauss-Newton or Levenberg-Marquardt.

[From [Mar13]]

Non linear approach principle

Then: $\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} P(\Phi) - u \\ Q(\Phi) - v \end{bmatrix} \Rightarrow V(\Phi) = \begin{bmatrix} e_x \\ e_y \end{bmatrix}$

We must find Φ which minimizes the reprojection error S:

Note: An implementation for learning purposes of a pose estimation using Gauss-Newton can be found at

- http://people.rennes.inria.fr/Eric.Marchand/pose-estimation/tutorial-pose-gauss-newton-opencv.html

[From [Mar13]]

Outline

"Gold standard" algorithm [HZ04]

Objective

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the Maximum Likehood estimated of the camera projection matrix $\mathbf{P}$, i.e. the $\mathbf{P}$ which minimizes $\sum_i d(\mathbf{x}_i, \mathbf{P}X_w)$

"Gold standard" algorithm [HZ04]

Objective

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the Maximum Likehood estimated of the camera projection matrix $\mathbf{P}$, i.e. the $\mathbf{P}$ which minimizes $\sum_i d(\mathbf{x}_i, \mathbf{P}X_w)$
$\sum_i d(\mathbf{x}_i, \mathbf{P}X_w)$ is the geometric error in the image.
Minimizing geometric error require the use of iterative techniques (as Levenberg-Marquardt).
If the measurement error are Gaussian then the solution of

$$\min_{\mathbf{P}} \sum_i d(\mathbf{x}_i, \mathbf{P}X_w)$$

is the Maximum Likelihood estimate under $\mathbf{P}$.
DLT solution (or a minimal solution) is used as a starting point for the iterative minimization.

[From [HZ04]

"Gold standard" algorithm [HZ04]

Objective

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the Maximum Likehood estimated of the camera projection matrix $\mathbf{P}$, i.e. the $\mathbf{P}$ which minimizes $\sum_i d(\mathbf{x}_i, \mathbf{P}X_w)$

1. **Linear Solution** Compute an initial estimate of $\mathbf{P}$ using previous linear method.
   1. **Normalization** Use a similarity transformation T to normalize the image points $\tilde{\mathbf{x}}_{ii} = T\mathbf{x}_i$, and a second similarity transformation U to normalize the space points $\tilde{X}_w = UX_w$
   2. Apply **DLT** algorithm
2. **Minimize geometric error** Using the linear estimate as a starting point minimize the geometric error

$$\sum_i d(\tilde{\mathbf{x}}_i, \tilde{\mathbf{P}}\tilde{X}_w)$$

over $\tilde{P}$, using an iterative algorithm such as Levenberg-Marquardt.

"Gold standard" algorithm [HZ04]

Objective

Given $n \geq 6$ world to image point correspondences $X_w \leftrightarrow \mathbf{x}_i$, determine the Maximum Likehood estimated of the camera projection matrix $\mathbf{P}$, i.e. the $\mathbf{P}$ which minimizes $\sum_i d(\mathbf{x}_i, \mathbf{P}X_w)$

1. **Linear Solution** Compute an initial estimate of $\mathbf{P}$ using previous linear method.
2. **Minimize geometric error** Using the linear estimate as a starting point minimize the geometric error

$$\sum_i d(\tilde{\mathbf{x}}_i, \tilde{\mathbf{P}}\tilde{X}_w)$$

   over $\tilde{\mathbf{P}}$, using an iterative algorithm such as Levenberg-Marquardt.
3. **Denormalization** The camera matrix for the original (unnormalized) coordinates is obtained from $\tilde{\mathbf{P}}$ as

$$\mathbf{P} = T^{-1}\tilde{\mathbf{P}}U$$

Outline

References

Presented methods where proposed by Roberts [Rob63], Tsai [Tsa87], Lowe [Low85, Low91], Yuan [Yua89] and Zhang [Zha00] (among others).

Tools

- OpenCV ▸ https://opencv.org *(Open source)*
- OpenGV ▸ https://laurentkneip.github.io/opengv/index.html *(Open source)*
- ViSP ▸ https://visp.inria.fr *(Open source)*
- MATLAB Toolboxes ▸ https://fr.mathworks.com/solutions/image-video-processing.html or ▸ http://www.vision.caltech.edu/bouguetj/calib_doc/
- Omnidirectional Calibration Toolbox ▸ http://www.robots.ox.ac.uk/čmei/Toolbox.html#download

Pose estimation for augmented reality: a hand-on survey [MUS16] codes and explanations ▸ http://people.rennes.inria.fr/Eric.Marchand/pose-estimation/index.html

Calibration with a chessboard

Any appropriately characterized object could be used as a calibration object
$\Rightarrow$ practically: use of regular pattern as a chessboard

Rq.: The specific use of this calibration object and much of the calibration approach itself comes from [Zha00] and [SM99]

Demo: openCV calibration example code

- https://github.com/opencv/opencv/blob/master/samples/cpp/calibration.cpp

Calibration with a chessboard

Any appropriately characterized object could be used as a calibration object
$\Rightarrow$ practically: use of regular pattern as a chessboard

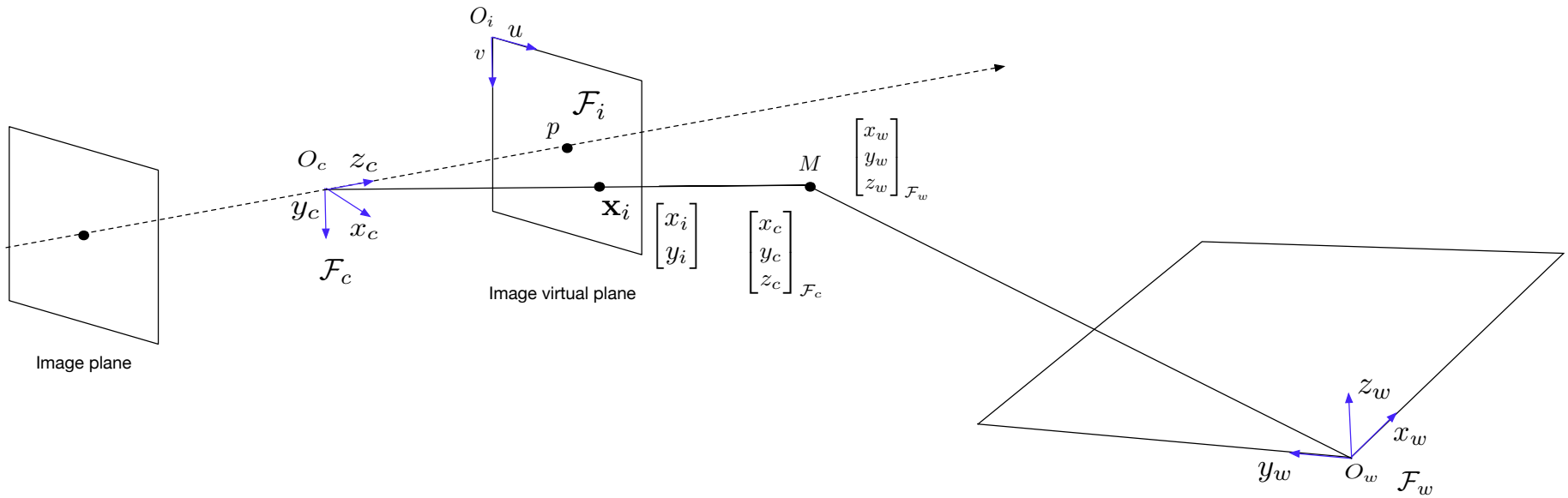
Rq.: The specific use of this calibration object and much of the calibration approach itself comes from [Zha00] and [SM99]

Demo: openCV calibration example code

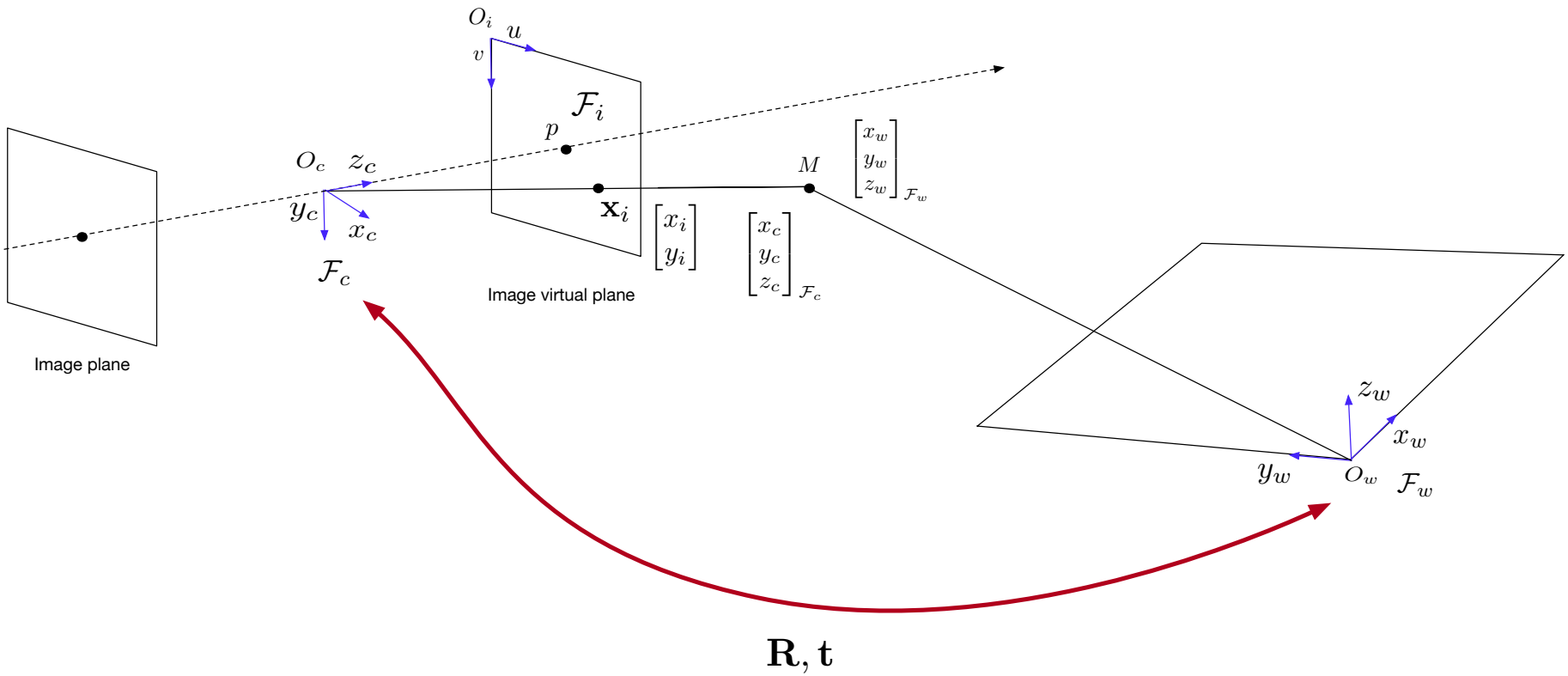
- https://github.com/opencv/opencv/blob/master/samples/cpp/calibration.cpp

In the yml file, you will find the camera matrix $\left[f_x, 0, u_0, 0, f_y, v_0, 0, 0, 1\right]$ and the distortion coefficients $[k_1, k_2, p_1, p_2, k_3]$

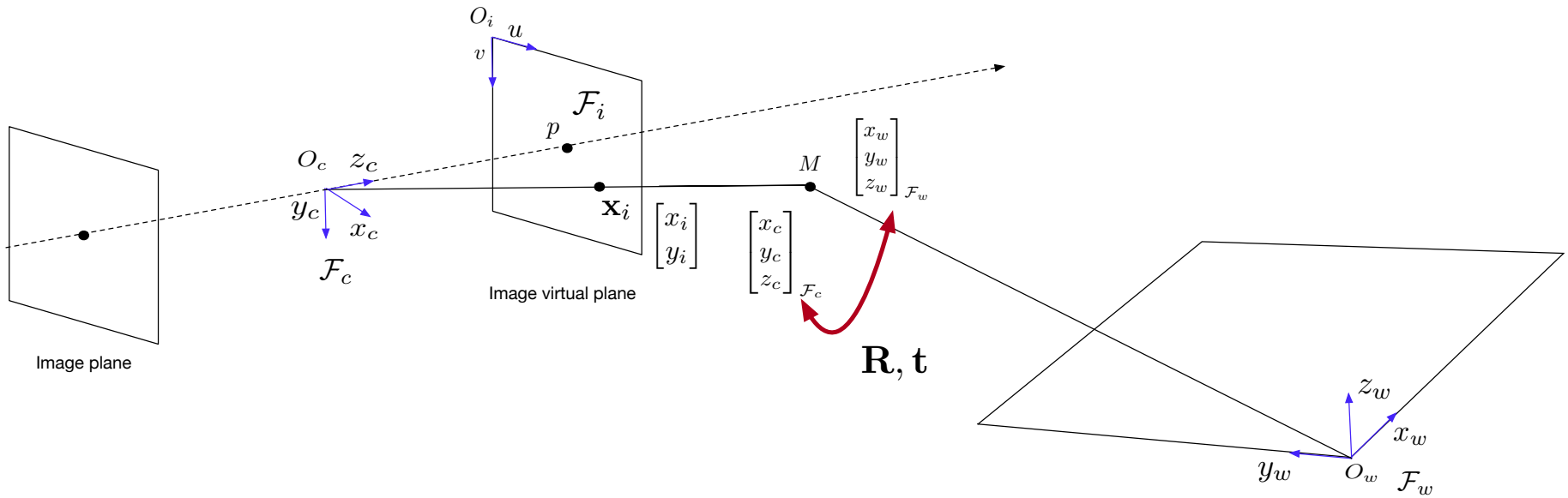
Summary on perspective projection



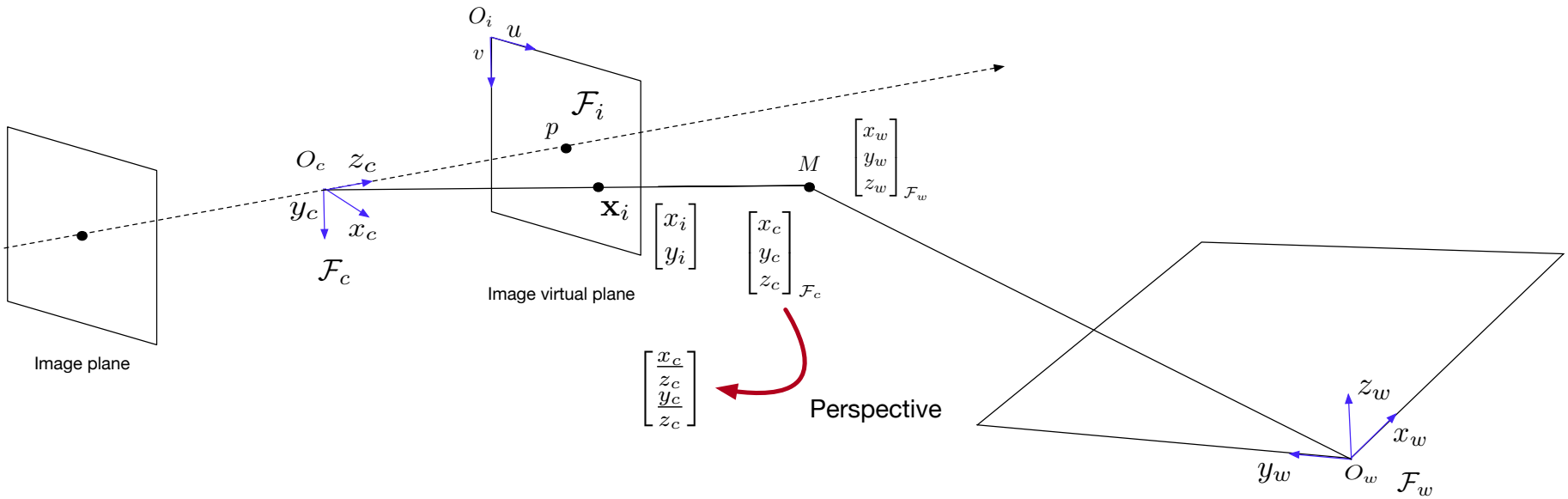
Summary on perspective projection



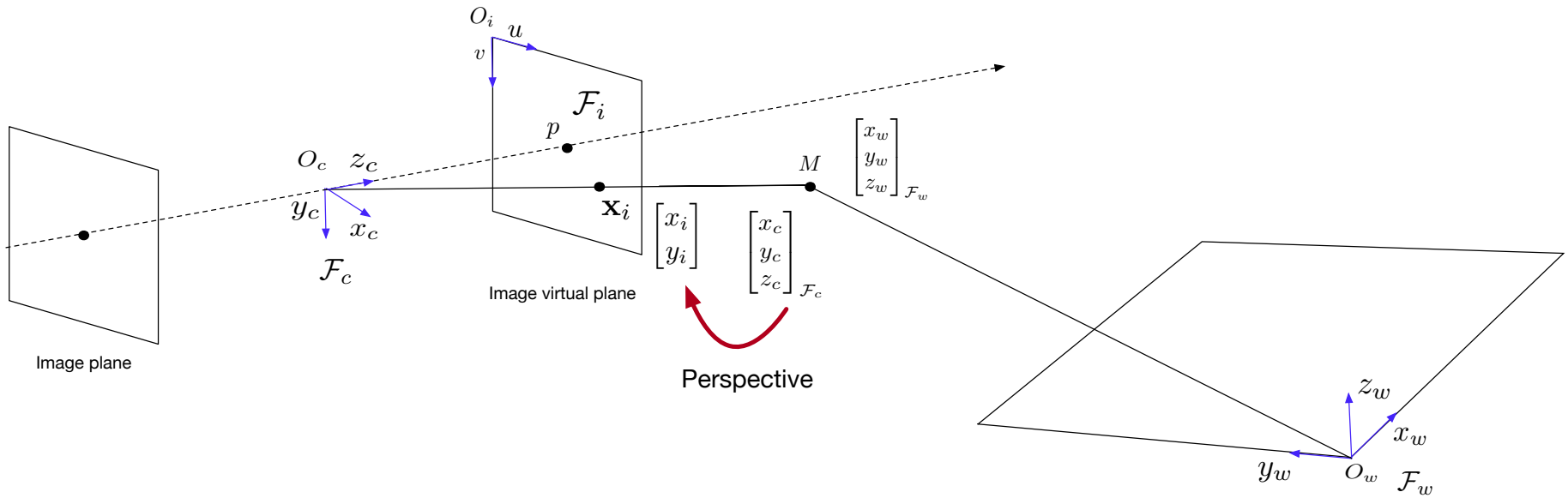
Summary on perspective projection



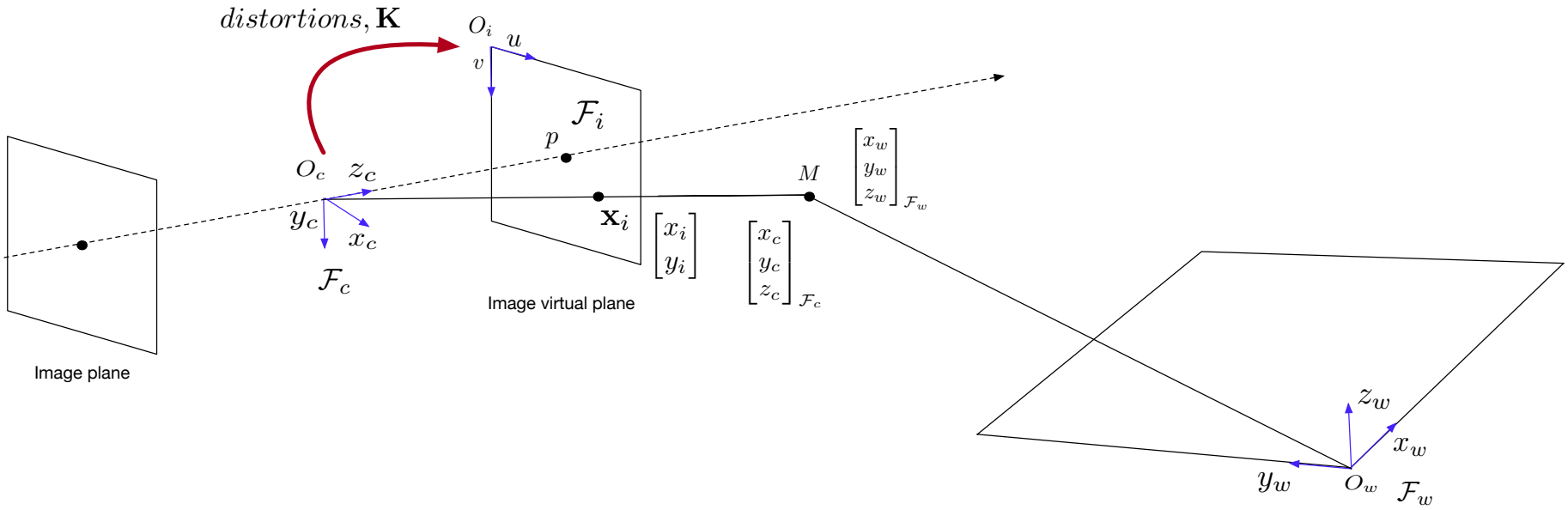
Summary on perspective projection



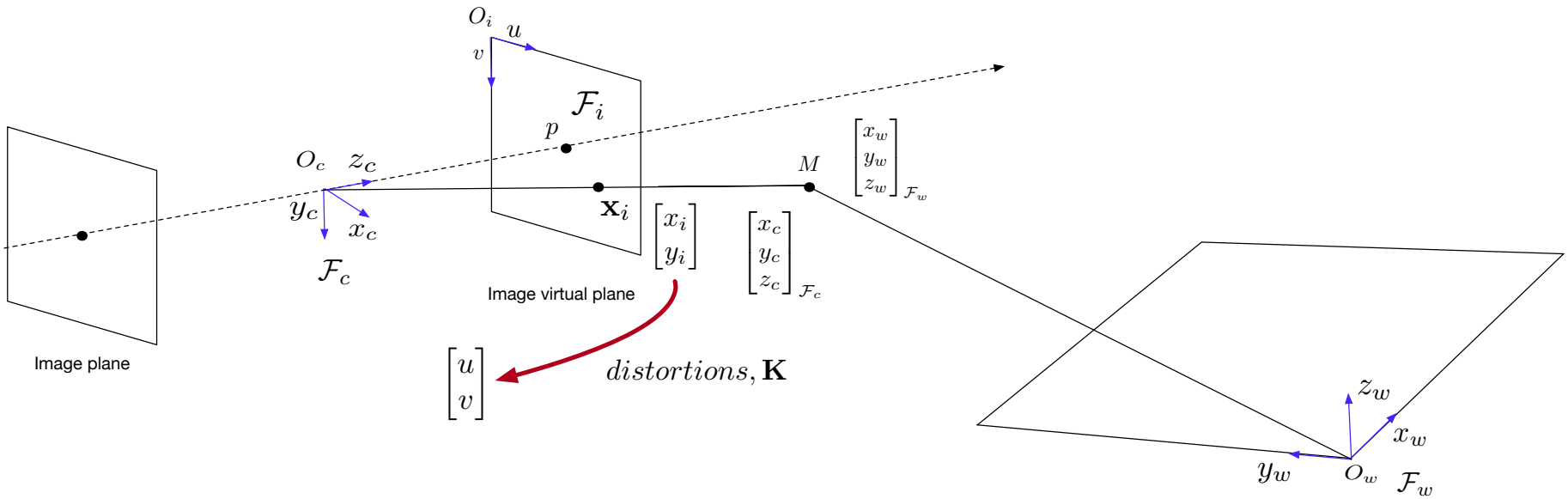
Summary on perspective projection



Summary on perspective projection



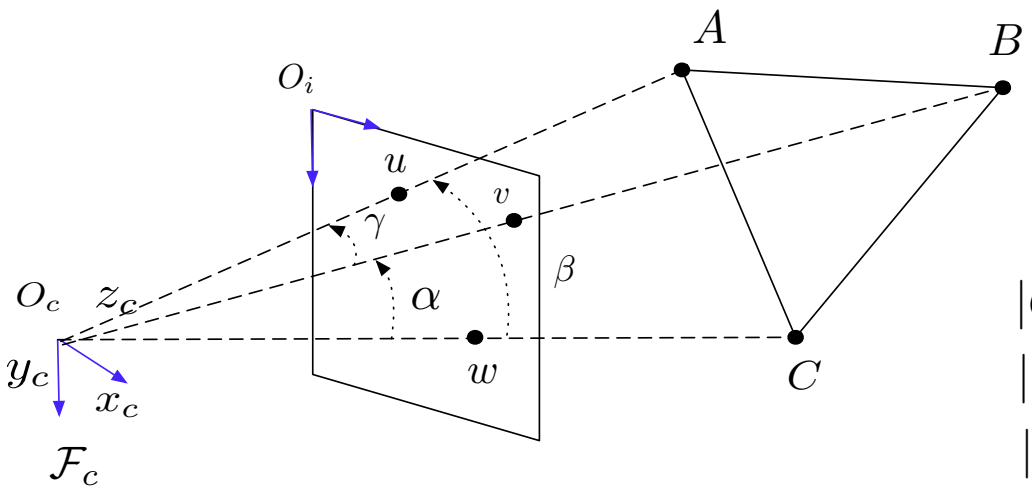
Summary on perspective projection



Outline

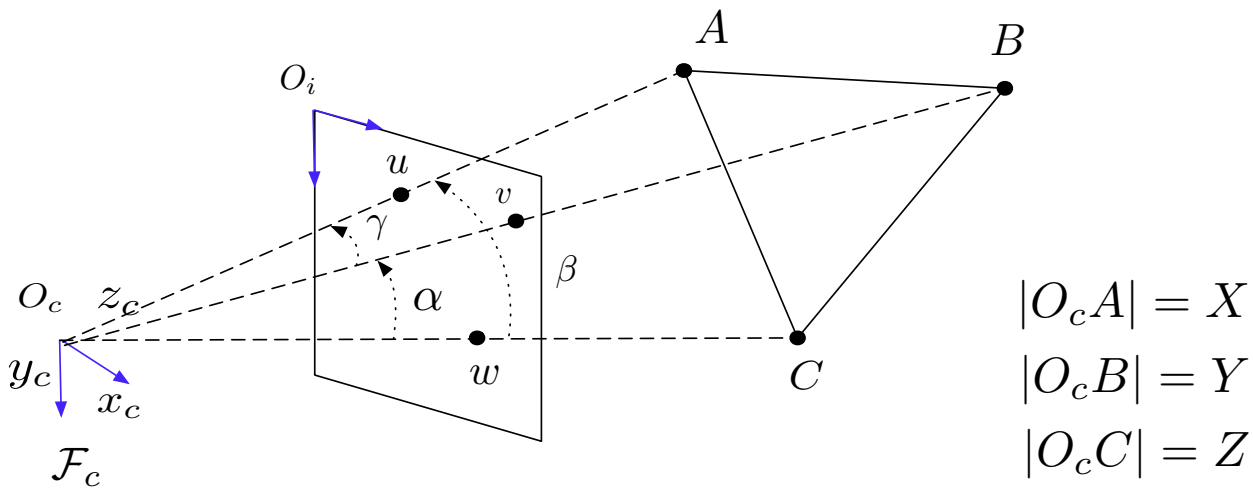
Outline

P3P : pose estimation with the smallest correspondence subset



$$|O_cA| = X$$
$$|O_cB| = Y$$
$$|O_cC| = Z$$

P3P : pose estimation with the smallest correspondence subset



Most of the P3P approaches rely on a 2 steps solution:

- Estimation of the unknown depth of each point (in $\mathcal{F}_c$)
- Estimating the rigid transformation that maps the coordinates expressed in $\mathcal{F}_c$ to the coordinates expressed in $\mathcal{F}_w$

P3P : pose estimation with the smallest correspondence subset

- Estimation of the unknown depth of each point (in $\mathcal{F}_c$):
  - Use of triangles O_cAB, O_cBC and O_cAC and the law of cosines:

$$Y^2 + Z^2 - 2YZ\cos\alpha - |BC|^2 = 0$$
$$Z^2 + X^2 - 2ZX\cos\beta - |AC|^2 = 0$$
$$X^2 + Y^2 - 2XY\cos\gamma - |AB|^2 = 0$$

  - Solve a fourth order polynomial equation (example in [GHTC03])
    $\rightarrow$ up to 4 four possible solutions
- Estimating the rigid transformation from $\mathcal{F}_c$ to $\mathcal{F}_w$

P3P : pose estimation with the smallest correspondence subset

- Estimation of the unknown depth of each point (in $\mathcal{F}_c$):
  - Use of triangles O_cAB, O_cBC and O_cAC and the law of cosines:

$$Y^2 + Z^2 - 2YZ\cos\alpha - |BC|^2 = 0$$
$$Z^2 + X^2 - 2ZX\cos\beta - |AC|^2 = 0$$
$$X^2 + Y^2 - 2XY\cos\gamma - |AB|^2 = 0$$

  - Solve a fourth order polynomial equation (example in [GHTC03])
    $\rightarrow$ up to 4 four possible solutions
- Estimating the rigid transformation from $\mathcal{F}_c$ to $\mathcal{F}_w$

Necessity to have a 4th point to disambiguate the results.

Outline

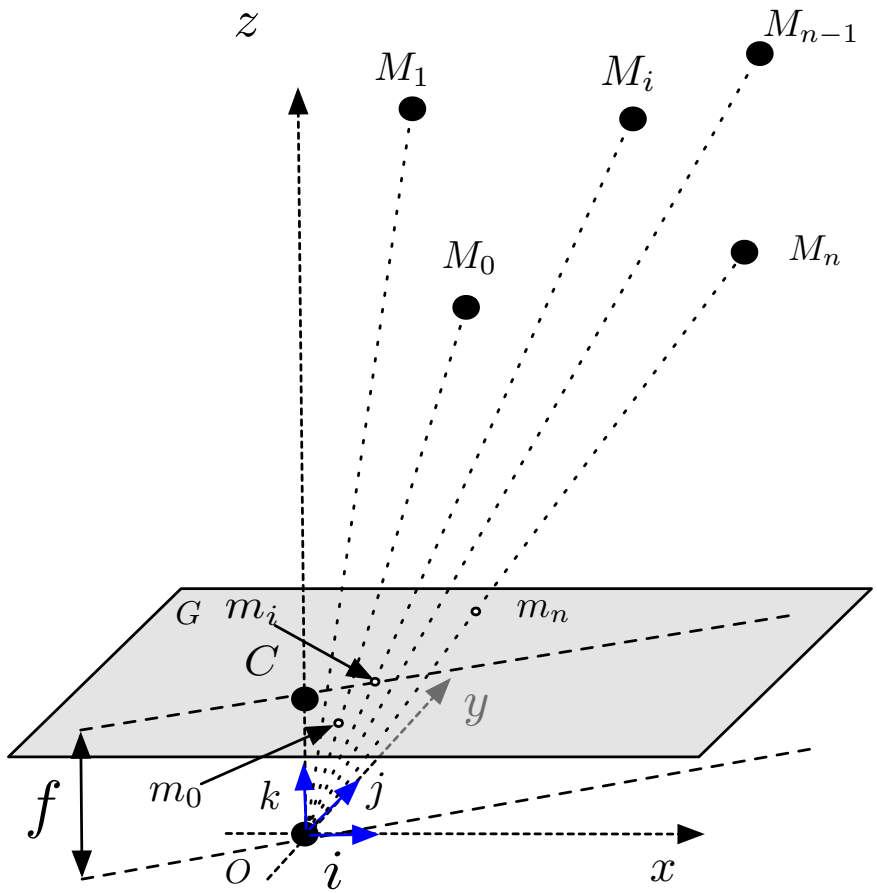
POS and POSIT [DD95]: overview of the problem

Notations:

- O center of projection
- $(\vec{i}, \vec{j}, \vec{k})$ the camera frame
- f focal length
- G the image plane
- C the central point

Hypothesis:

- pinhole camera model
- n non coplanar 3D points: M_0, M_1,... M_n
- m_0, m_1, ... m_n: perspective projection in image plane of the corresponding 3D points



POS and POSIT [DD95]: overview of the problem

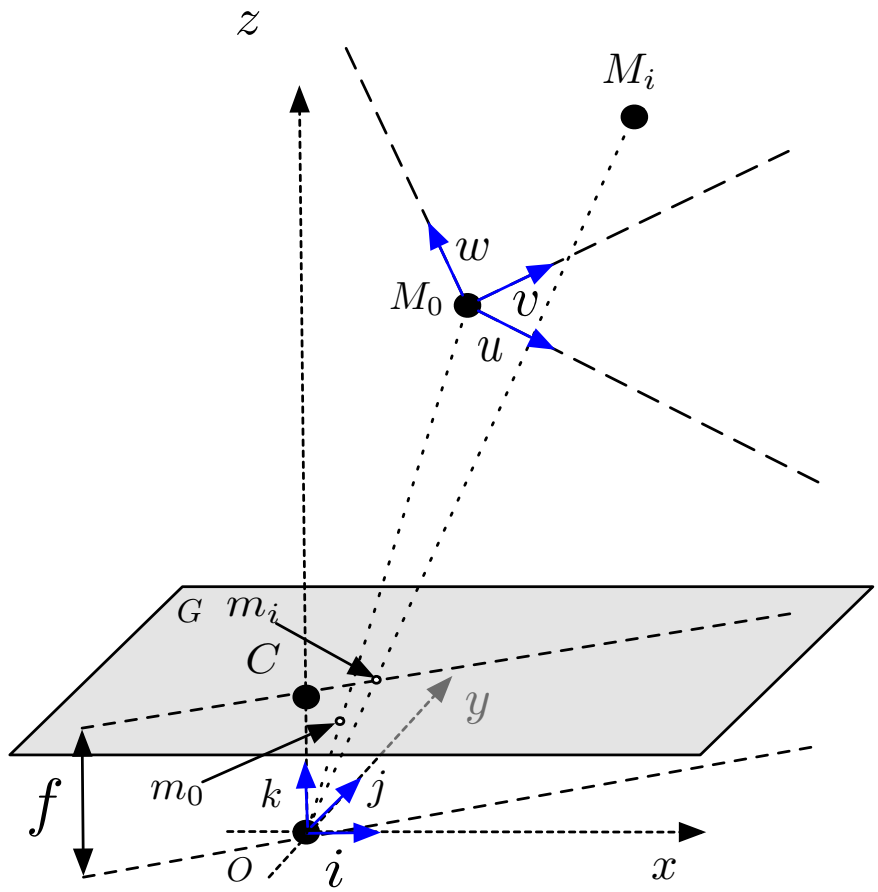
Notations:

- $\mathcal{F}_o$: object frame as M_0 origin with (u, v, w)
- $M_i = \begin{bmatrix} U_i \\ V_i \\ W_i \end{bmatrix}$ in $\mathcal{F}_o$
- $m_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ in the image plane

Hypothesis:

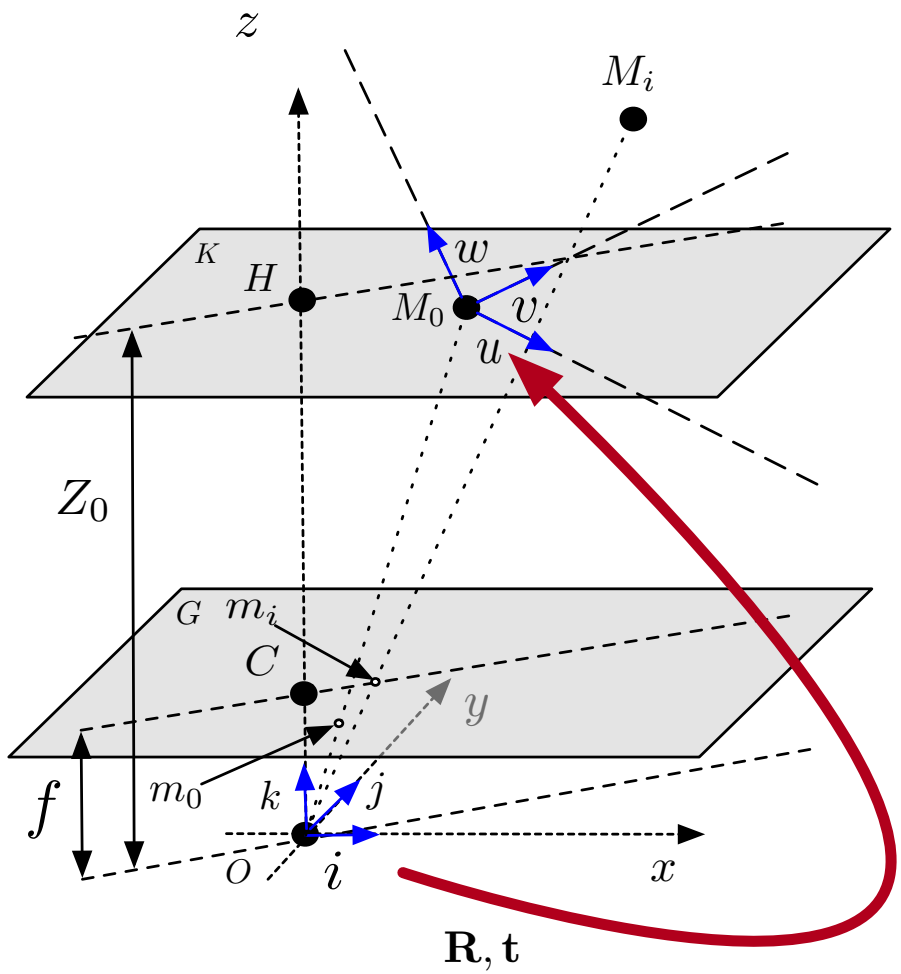
- the shape of the object is known $\Rightarrow$ M_i coordinates in $\mathcal{F}_o$ are known

We want to retrieve $[X_i, Y_i, Z_i]^T$ coordinates of M_i in $\mathcal{F}_c$



POS and POSIT [DD95]: overview of the problem

- K: plane parallel to image plane G containing M_0 at Z_0 from $\mathcal{F}_c$
- Problem: compute $\mathbf{R}, \mathbf{t}$

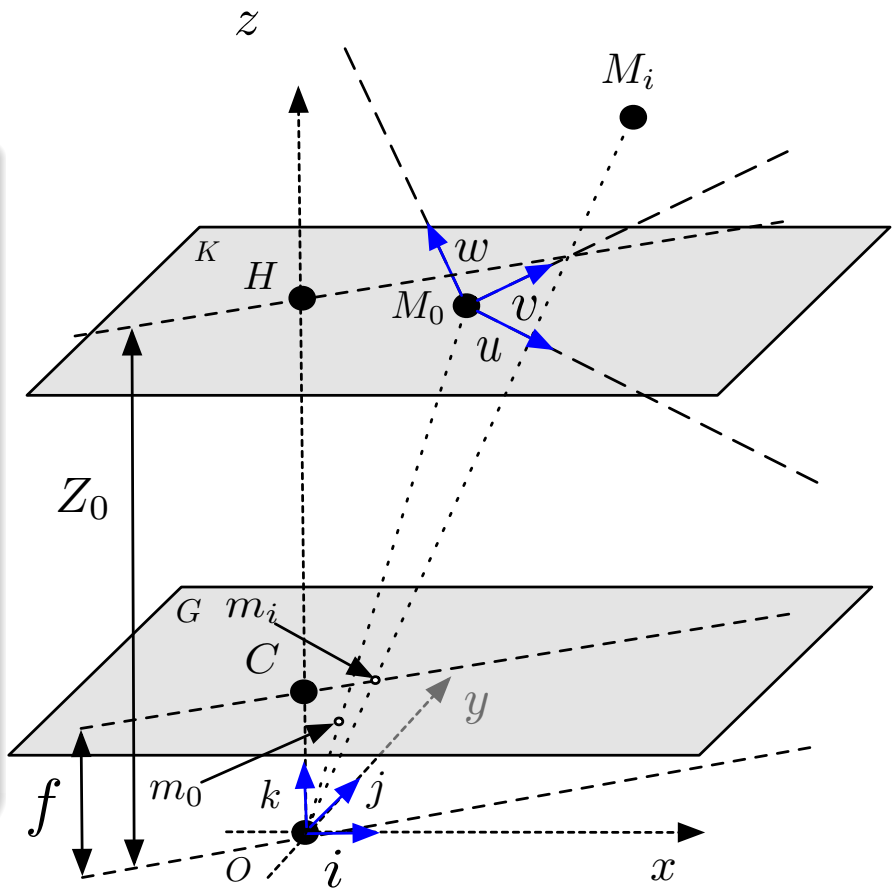


POS and POSIT [DD95]: Compute $\mathbf{R}, \mathbf{t}$

R

$$R = \begin{bmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{bmatrix} = \begin{bmatrix} \vec{i}^T \\ \vec{j}^T \\ \vec{k}^T \end{bmatrix}$$

only need to compute $\vec{i}, \vec{j}$ then $\vec{k} = \vec{i} \times \vec{j}$



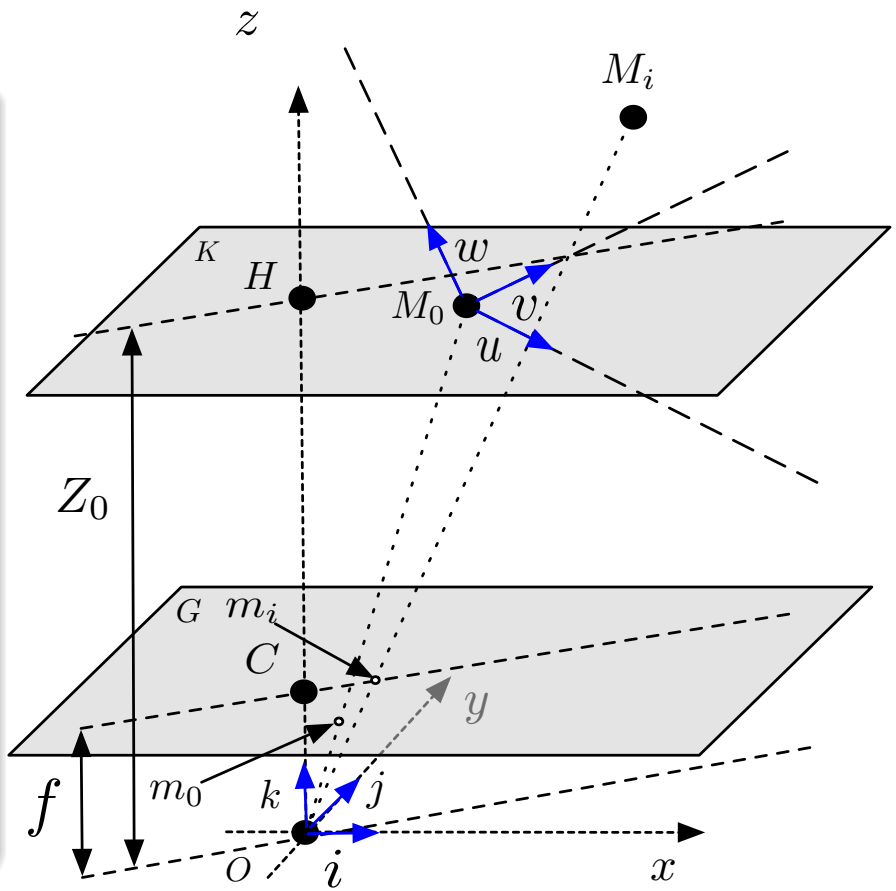
POS and POSIT [DD95]: Compute $\mathbf{R}, \mathbf{t}$

t

Given by $O\vec{M}_0$:

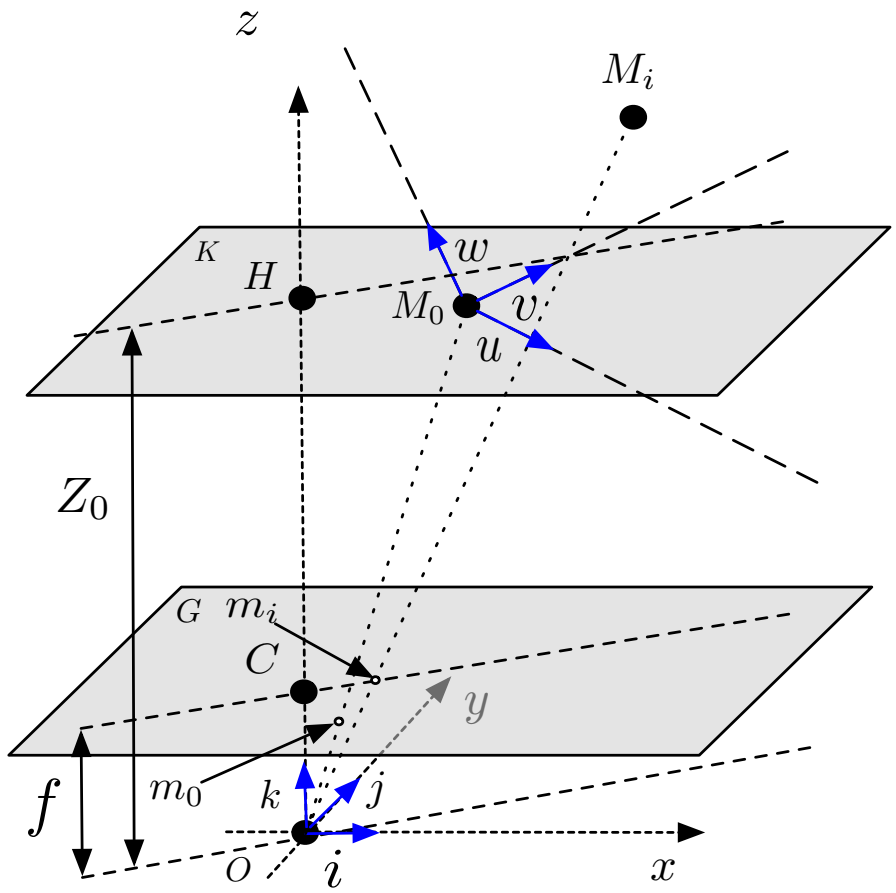
$$\mathbf{t} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

and O, m_0 and M_0 aligned:
$\mathbf{t} = \frac{Z_0}{f} O\vec{m}_0$
Then to get $\mathbf{t}$, only Z_0 is missing.



POS and POSIT [DD95]: Compute $\mathbf{R}, \mathbf{t}$

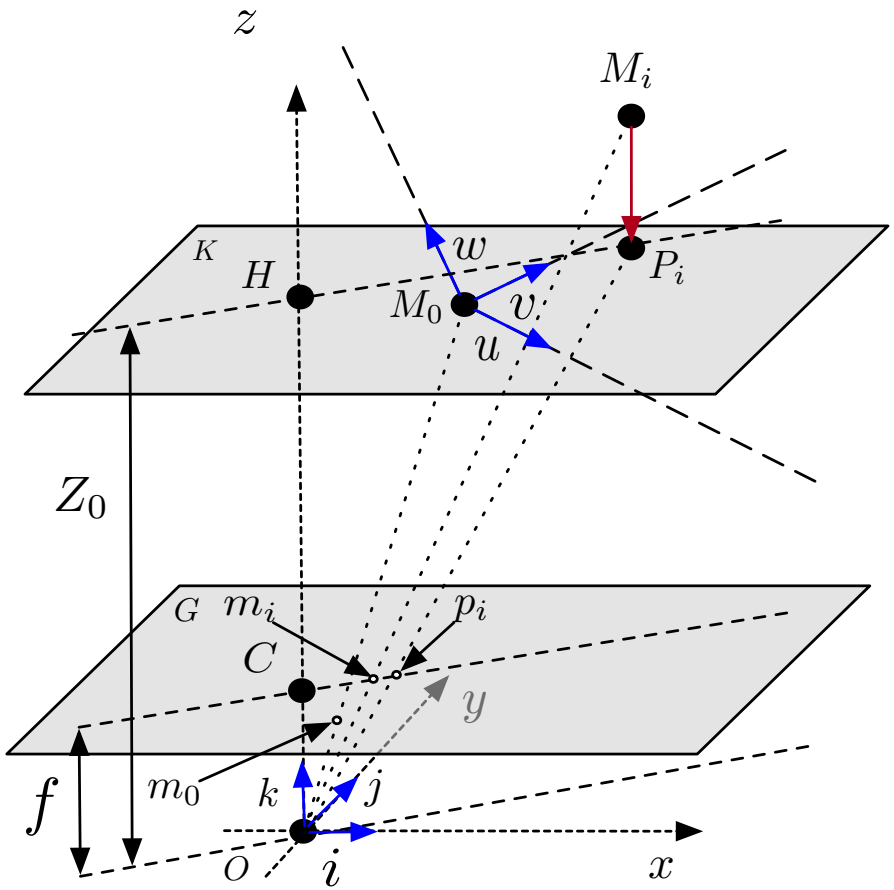
The object pose is fully defined once we have $\vec{i}, \vec{j}$ and Z_0.



POS and POSIT [DD95]: Compute $\mathbf{R}, \mathbf{t}$

- Scaled orthographic projection (SOP):
  - an approximation to *true* perspective projection
  - consider an object that we can consider all points with the same depth
- SOP vs. Perspective projection (PP):

SOP	PP
$M_i \Rightarrow P_i \Rightarrow p_i$	$M_i \Rightarrow m_i$
$p_i = (x'_i, y'_i)^T$	$m_i = (x_i, y_i)^T$
$p_i = (f\frac{X_i}{Z_0}, f\frac{Y_i}{Z_0})^T$	$m_i = (f\frac{X_i}{Z_i}, f\frac{Y_i}{Z_i})^T$



POS and POSIT [DD95]: Compute $\mathbf{R}, \mathbf{t}$

- SOP *scaling factor*: $s = \frac{f}{Z_0}$
- M_0 projects on m_0 with SOP and PP

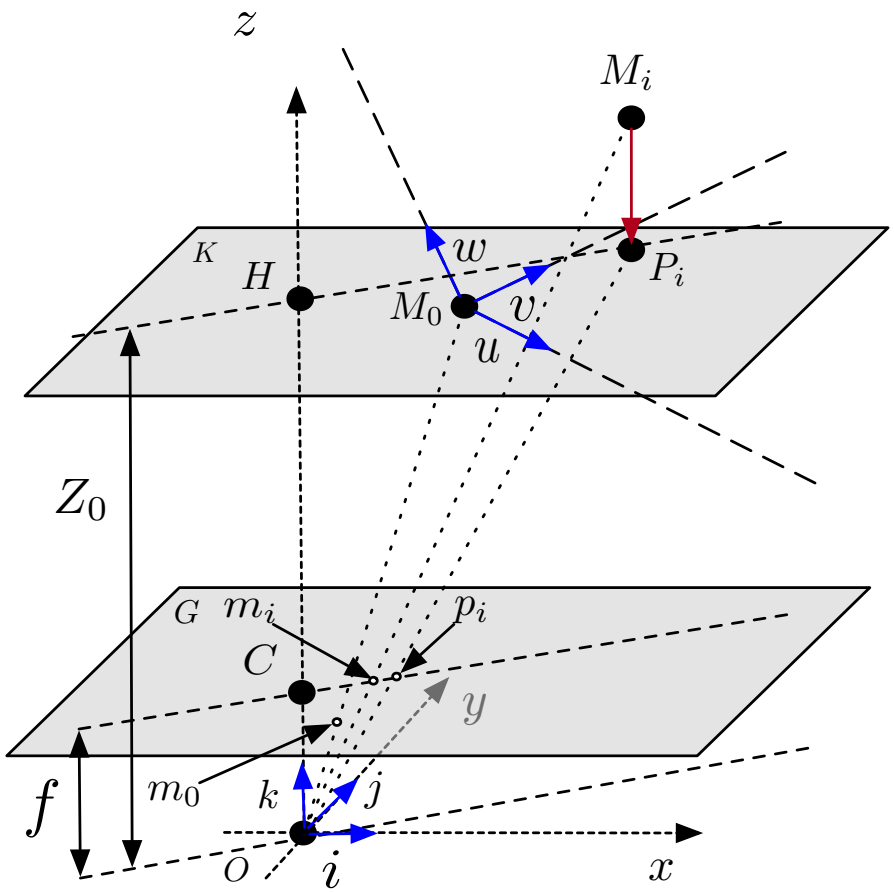
We can write:

$$x_i' = f\frac{X_i}{Z_0} = f\frac{X_i - X_0 + X_0}{Z_0}$$
$$= x_0 + s(X_i - X_0)$$

And deduce:

$$x_i' = x_0 + s(X_i - X_0)$$
$$y_i' = y_0 + s(Y_i - Y_0)$$

and: $\vec{m_0 p_i} = \frac{f}{Z_0}\vec{M_0 P_i}$



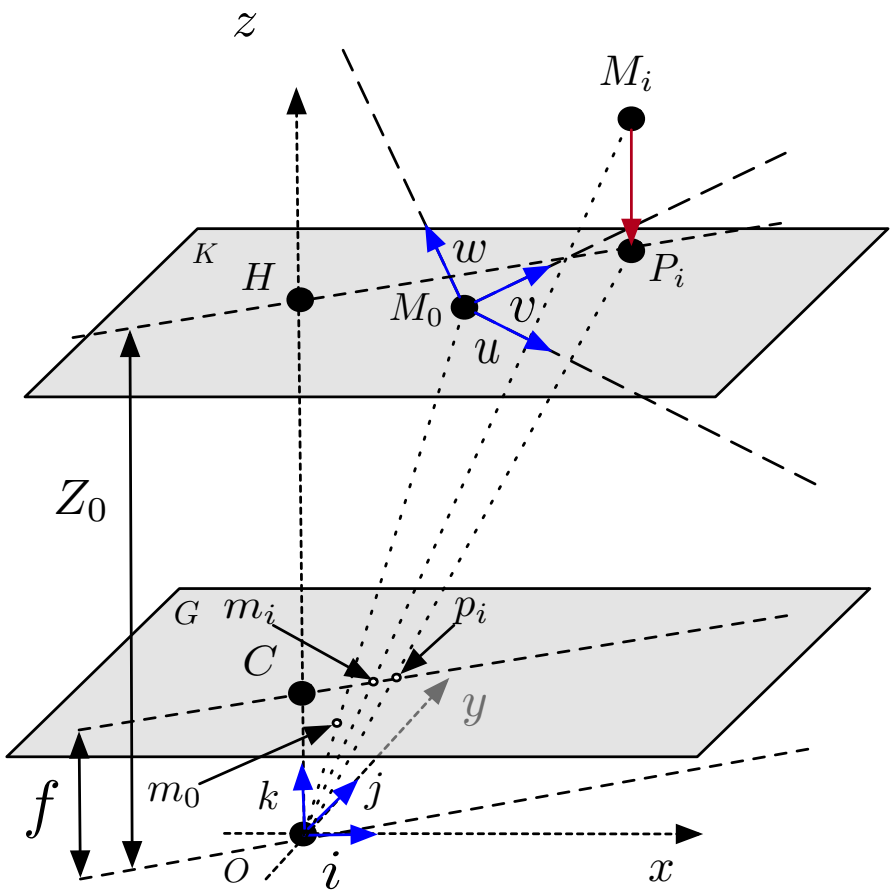
POS and POSIT [DD95]: Compute $\mathbf{R}, \mathbf{t}$

- Known coordinates of $\vec{M_0M_i}$ and $\vec{m_0m_i}$
- Unknown: $\vec{i}, \vec{j}$ and Z_0
- Links between known and unknown:

$$\vec{M_0M_i} \cdot \frac{f}{Z_0}\vec{i} = x_i(1+\epsilon_i) - x_0$$

$$\vec{M_0M_i} \cdot \frac{f}{Z_0}\vec{j} = y_i(1+\epsilon_i) - y_0$$

with $\epsilon_i = \frac{1}{Z_0}\vec{M_0M_i} \cdot \vec{k}$
[demonstration in [DD95]]



POS and POSIT [DD95]: Compute $\mathbf{R}, \mathbf{t}$

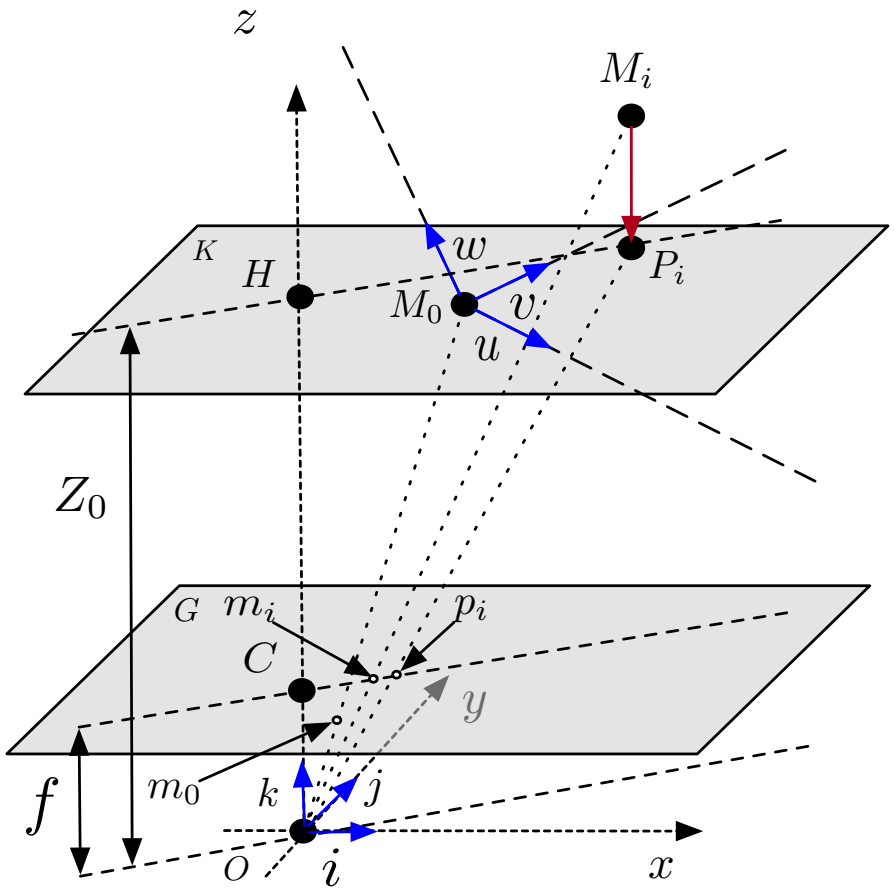
Re-writing:

$$\vec{M_0M_i} \cdot \vec{I} = x_i(1+\epsilon_i) - x_0$$
$$\vec{M_0M_i} \cdot \vec{J} = y_i(1+\epsilon_i) - y_0$$

with

$$\vec{I} = \frac{f}{Z_0}\vec{i}$$
$$\vec{J} = \frac{f}{Z_0}\vec{j}$$

So if give values to $\epsilon_i \Rightarrow$ linear system with two unknowns $\vec{I}$ and $\vec{J}$



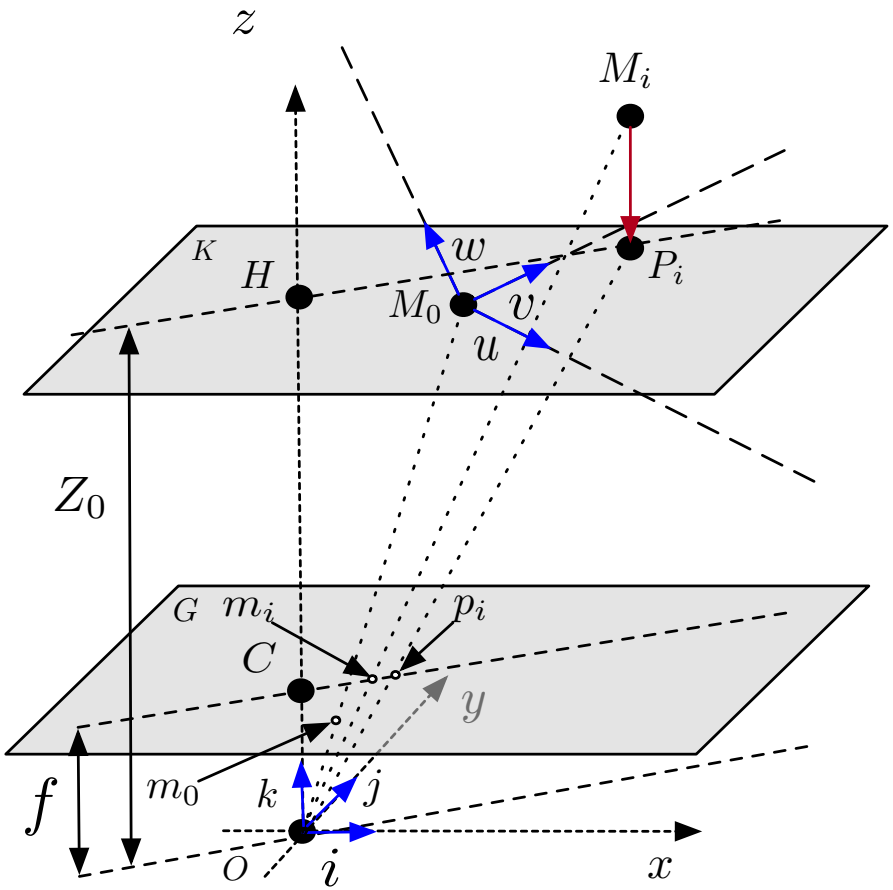
POS and POSIT [DD95]: POS algorithm

POS algorithm: *Pose from Orthography and Scaling*

- Give an approximate value to ϵ_i
- Solve linear system with unknowns $\vec{I}$ and $\vec{J}$
- Get $\vec{i}, \vec{j}, \vec{k}$ with

$$\vec{i} = \vec{I}/||\vec{I}||$$
$$\vec{j} = \vec{J}/||\vec{J}||$$
$$\vec{k} = \vec{i} \times \vec{j}$$

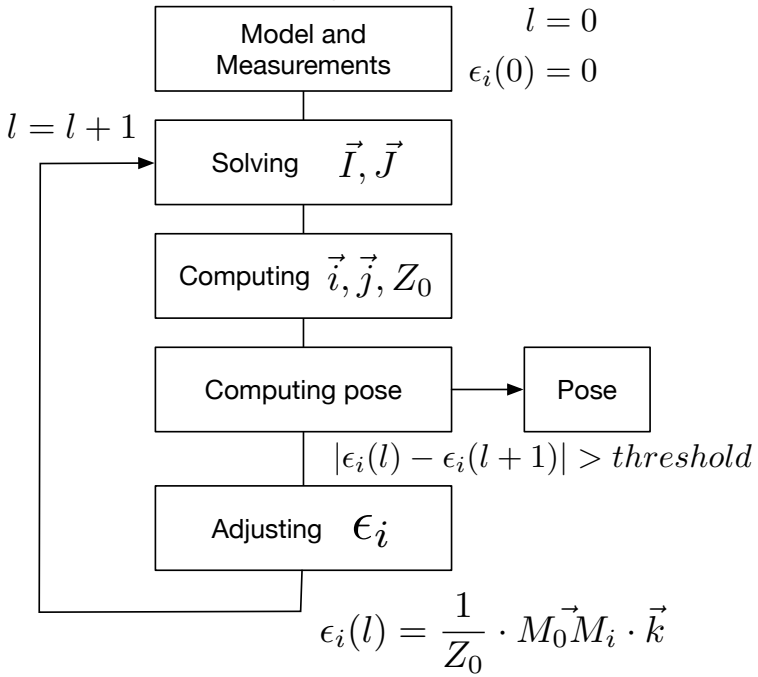
- Get $Z_0 = \frac{f}{\vec{I}}$



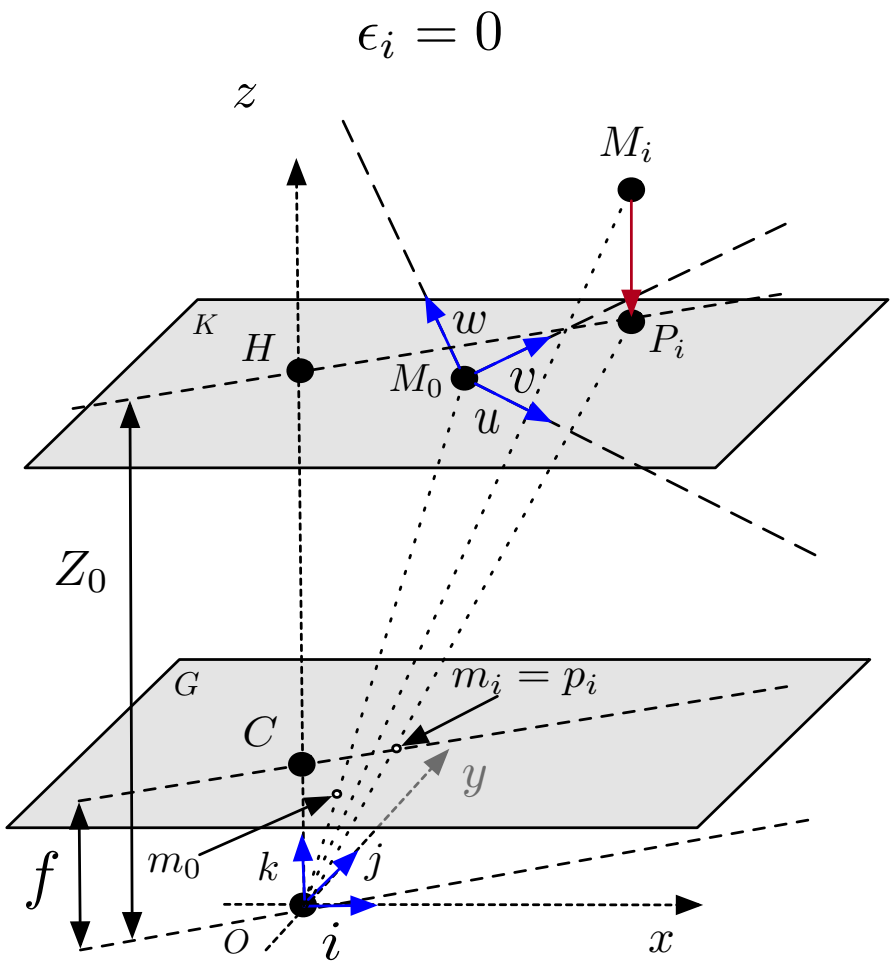
POS and POSIT [DD95]: POSIT algorithm

POSIT algorithm: POS with iterations

- Solve $\vec{i}$ and $\vec{j}$ then estimate ϵ_i



[From [Mar13]]



POS and POSIT [DD95]: POSIT algorithm

- Extended to coplanar feature points in [ODD96]
- Note: an implementation of POSIT for leaning purposes can be found at
  - http://people.rennes.inria.fr/Eric.Marchand/pose-estimation/tutorial-pose-dementhon-opencv.html

Outline

EPnP algorithm [LFNP09]

- non-iterative solution to the PnP problem applicable for all $n \geq 4$
- handles both planar and non-planar configurations
- n 3D points coordinates expressed as a weighted sum of four *virtual* control points
- pose problem : estimation of the coordinates of control points in $\mathcal{F}_c$
- can be done in $O(n)$
  - expressing these coordinates as weighted sum of the eigenvectors of a 12×12 matrix
  - solving a small constant number of quadratic equations to pick the right weights

EPnP algorithm [LFNP09]: Parameterization in the General Case

- $p_i, i = 1, \ldots, n$: the n points whose 3D coordinates are known in $\mathcal{F}_w$
- $c_j, j = 1, \ldots, 4$: the 4 control points coordinates in $\mathcal{F}_w$
- $$p_i^w = \sum_{j=1}^{4} \alpha_{ij} c_j^w, \text{with} \sum_{j=1}^{4} \alpha_{ij} = 1$$

  where the α_{ij} are homogeneous barycentric coordinates.
- Can also be expressed in $\mathcal{F}_c$: $p_i^c = \sum_{j=1}^{4} \alpha_{ij} c_j^c$
- In theory the control points can be chosen arbitrarily, but for stability reason:
  - taking the centroid of the n reference points as one
  - select the others in such a way that they form a basis aligned with the principal directions

EPnP algorithm [LFNP09]: The Solution as Weighted Sum of Eigenvectors

- $x_{i\,i=1,\ldots,n}$ the 2D projections of the $p_{i\,i=1,\ldots,n}$ reference points:

$$\forall i, \lambda_i \begin{bmatrix} x_i \\ 1 \end{bmatrix} = \mathbf{K} p_i^c = \mathbf{K} \sum_{j=1}^{4} \alpha_{ij} c_j^c$$

- with $c_j^c = [x_j^c, y_j^c, z_j^c]^T$ and $x_i = [u_i, v_i]^T$:

$$\forall i, \lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix}$$

  - The unknown parameters of this linear system are the 12 control point coordinates $\{(x_j^c, y_j^c, z_j^c)\}_{j=1,\ldots,4}$ and the n projective parameters $\{\lambda_i\}_{j=1,\ldots,n}$.
  - last row implies $\lambda_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c$

EPnP algorithm [LFNP09]: The Solution as Weighted Sum of Eigenvectors

- Substituting λ_i expression in the first two rows gives two linear equations for each reference point:

$$\sum_{j=1}^{4} \alpha_{ij} f_x x_j^c + \alpha_{ij}(u_0 - u_i) z_j^c = 0$$

$$\sum_{j=1}^{4} \alpha_{ij} f_y y_j^c + \alpha_{ij}(v_0 - v_i) z_j^c = 0$$

- λ_i does not appear anymore in those equations
- concatenating them for all n reference points give a linear system: $Mx = 0$ where
  - $x = [{c_1^c}^T, {c_2^c}^T, {c_3^c}^T, {c_4^c}^T]^T$ is a 12-vector made of the unknowns
  - M is a $n \times 12$ matrix generated by arranging the coefficients of the two last equations for each reference point

EPnP algorithm [LFNP09]: The Solution as Weighted Sum of Eigenvectors

Solving $Mx = 0$

- the solution therefore belongs to the null space of M, and can be expressed as

$$x = \sum_{i=1}^{N} \beta_i \mathbf{v}_i$$

  where the set $\mathbf{v}_i$ are the columns of the right-singular vectors of M corresponding to the N null singular values of M
- they can be computed as the null eigenvectors of matrix $M^T M$
  - $M^T M$ is 12×12
  - Computing $M^T M$ has $O(n)$ complexity

EPnP algorithm [LFNP09]: Choosing the Right Linear Combination

- the effective dimension N of the null space of $M^T M$ can vary from 1 to 4 depending on the configuration of the reference points, the focal length of the camera, and the amount of noise
- compute solutions for all four values of N and keep the one that yields the smallest reprojection error

$$res = \sum_i d^2 \left(K[R|t] \begin{pmatrix} p_i^w \\ 1 \end{pmatrix}, x_i \right)$$

with $d(\tilde{a}, b)$ the 2D distance between point a expressed in homogeneous coordinates, and point b.
Note: Details on the 4 cases are given in [LFNP09]

EPnP algorithm [LFNP09]

- non-iterative solution to the PnP problem applicable for all $n \geq 4$
- handles both planar and non-planar configurations
- n 3D points coordinates expressed as a weighted sum of four *virtual* control points
- pose problem : estimation of the coordinates of control points in $\mathcal{F}_c$
- To improve accuracy: the output of the closed-form solution can be used to initialize a Gauss-Newton scheme that will choose the values β_i that minimize the change in distance between control points

Outline

Others PnP algorithms

- EPnP[LFNP09]: first accurate $O(n)$ solution to the PnP
- Other interesting $O(n)$ solutions when the number of point correspondences increases:
  - OPnP [ZKS+13]: Optimal PnP – parameterize the rotation by using non-unit quaternion and formulate the PnP problem into an unconstrained optimization problem.
  - GPnP [KFS13]: non-iterative n-point solution with linear complexity in the number of points – Extension of EPnP to Non-Perspective-n-Point problem (NPnP problem)
  - UPnP [KLS14]: Universal PnP – applicable to both central and non-central camera systems
  - MLPnP [ULH16]: real-time Maximum Likelihood solution to the Perspective-n-Point problem – statistically optimal solution to PnP
  - ...

Others P*n*P algorithms

- It also exists iterative techniques as for ex. LHM [LHM00]: orthogonal iteration method to directly minimize the object space error
- A good review as speed and accuracy of 13 P*n*P methods can be found in [ULH16]
  $\Rightarrow$ MLP*n*P has similar execution times compared to the fastest methods (EP*n*P still faster) and is better in terms of accuracy.
- Algorithms implementations can be found in OpenCV and OpenGV [KLS14]

P*n*P calculation demo

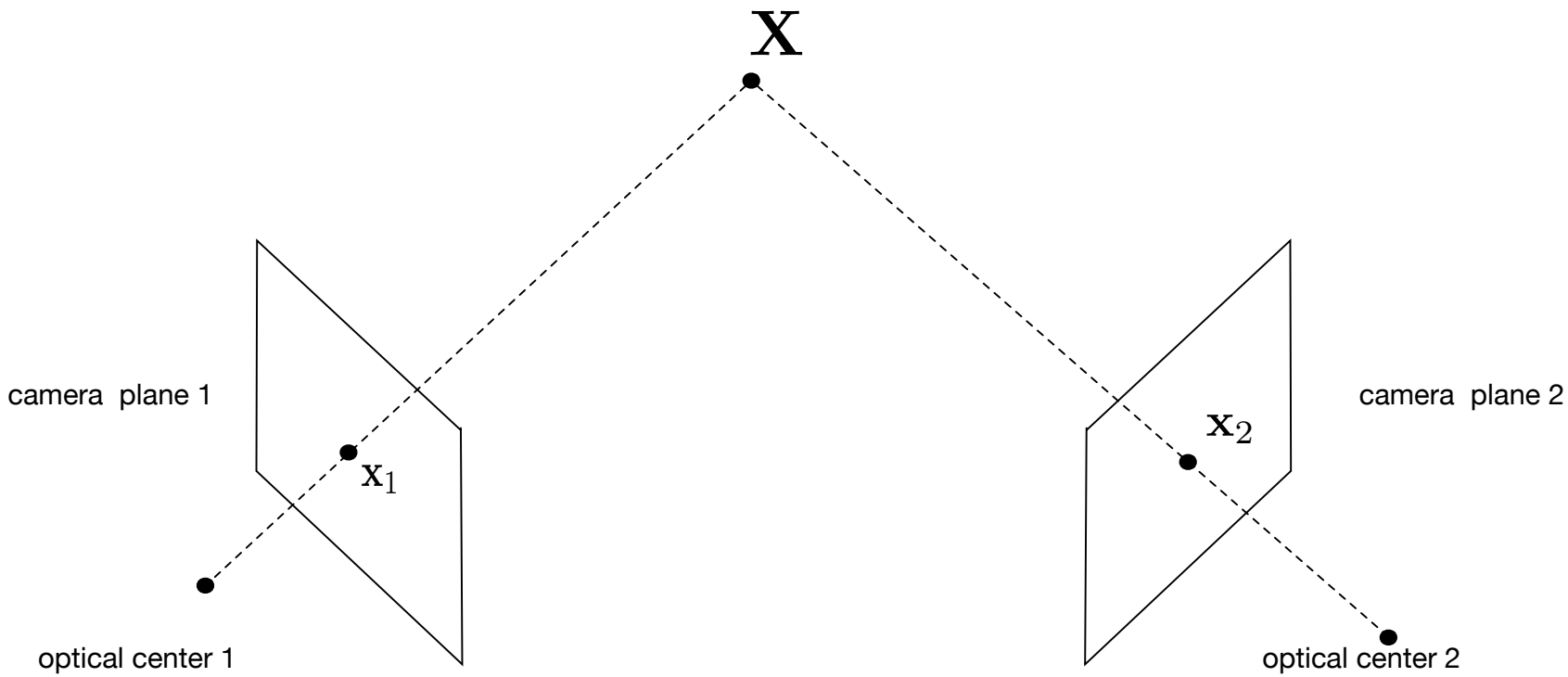
- Code from OpenCV real time pose estimation of a textured object tutorial
  - https://docs.opencv.org/3.3.0/dc/d2c/tutorial_real_time_pose.html
- Estimate the camera pose in order to track a textured object with six degrees of freedom given a 2D image and its 3D textured model
  - Read 3D textured object model and object mesh
  - Take input from Camera or Video
  - Extract ORB features and descriptors from the scene
  - Match scene descriptors with model descriptors
  - Pose estimation using PnP + Ransac
  - Linear Kalman Filter for bad poses rejection

Outline

Outline

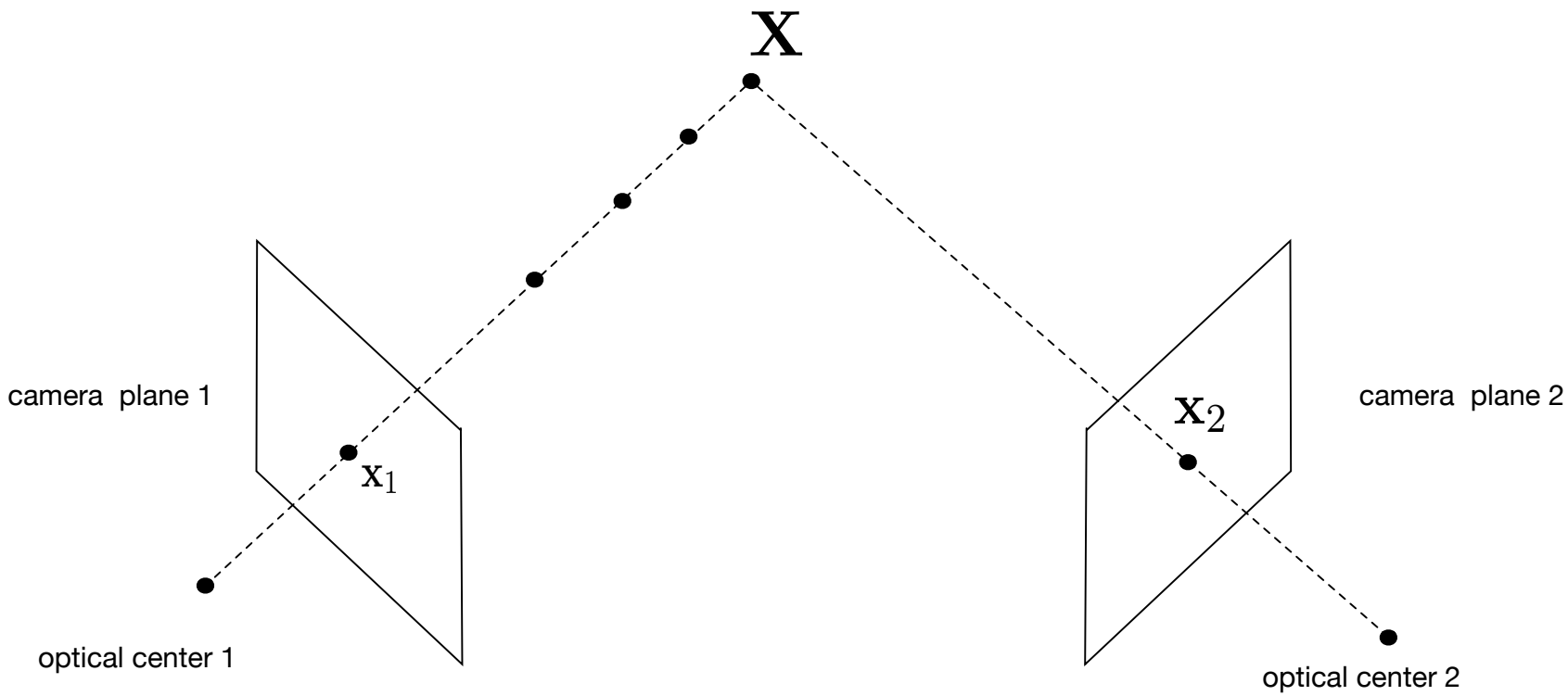
Outline

The epipolar constraint



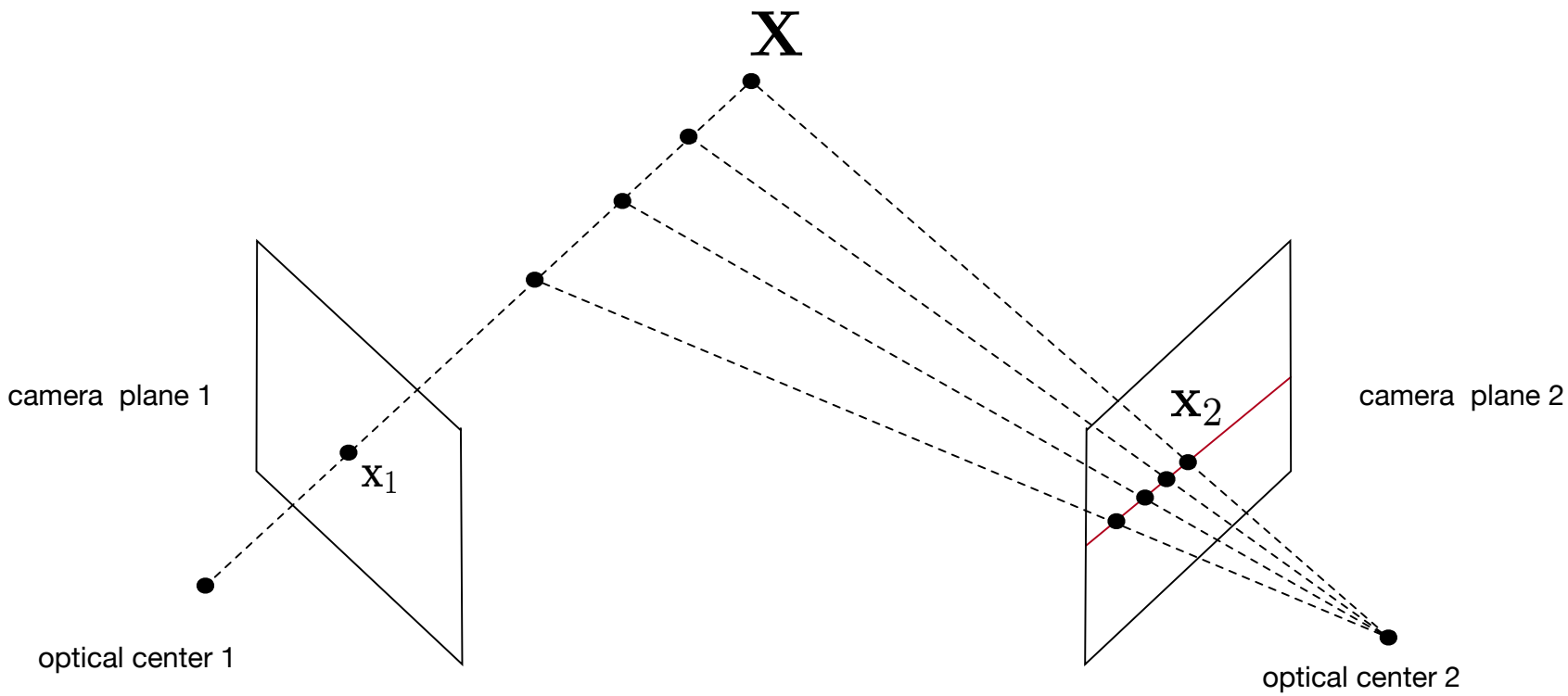
[From [Pri12]]

The epipolar constraint



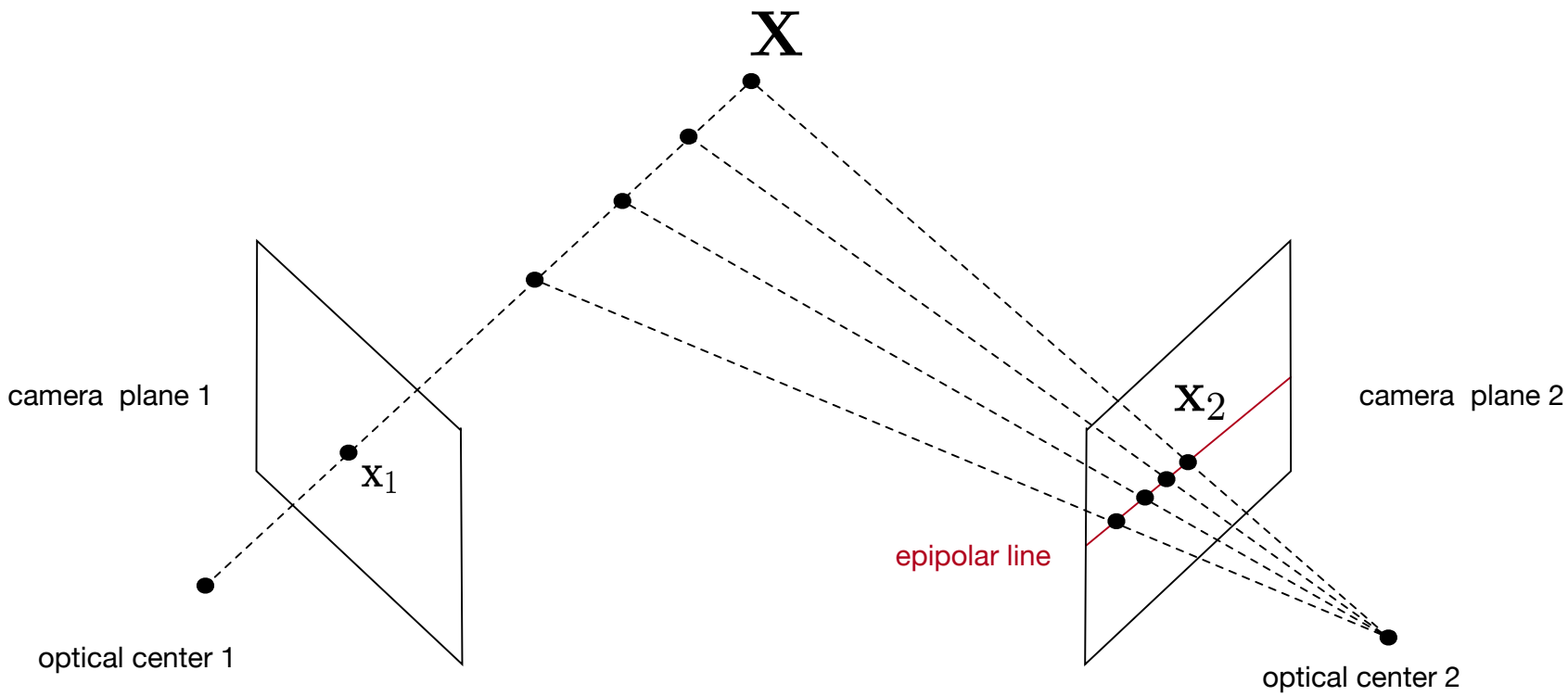
[From [Pri12]]

The epipolar constraint



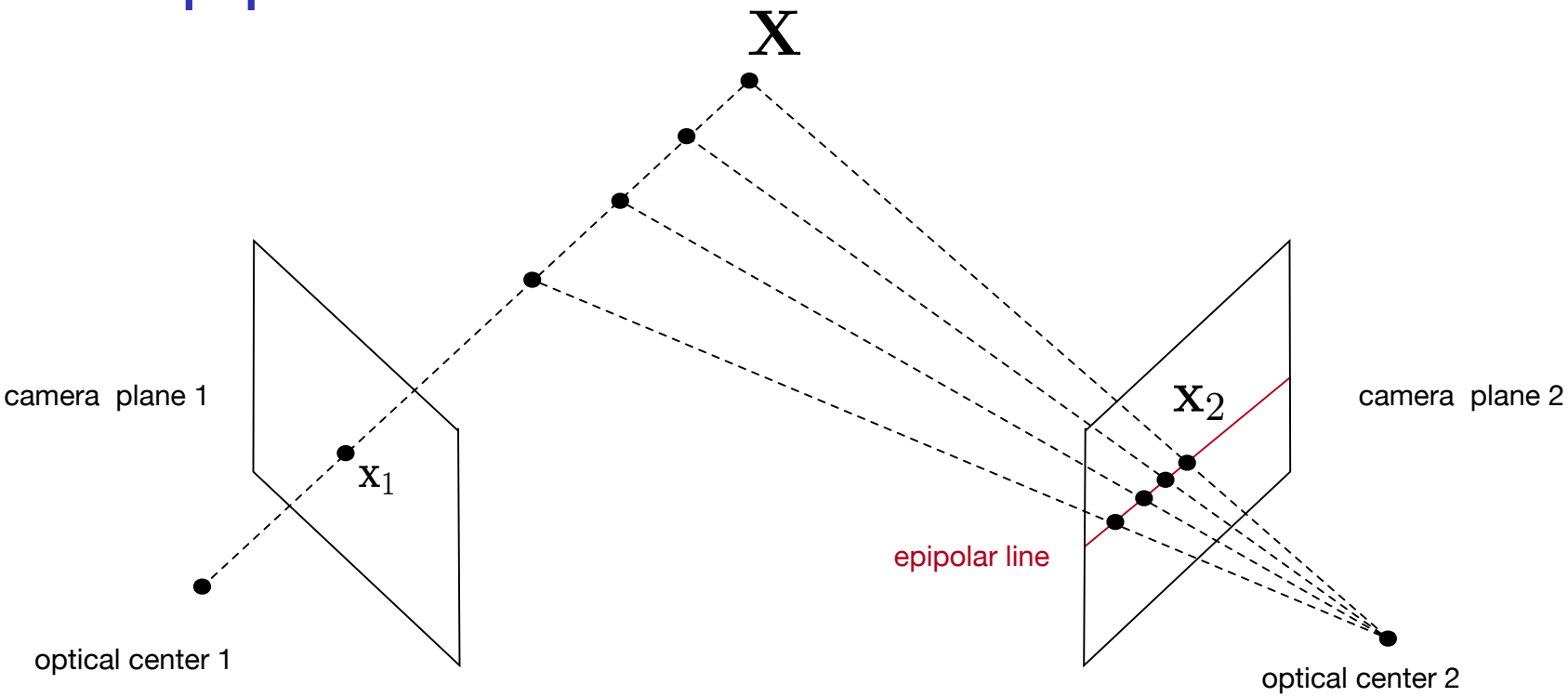
[From [Pri12]]

The epipolar constraint



[From [Pri12]]

The epipolar constraint



- *epipolar constraint*: for any point in the first image, the corresponding point in the second image is constrained to lie on a line
- The epipolar line depends on the intrinsic and extrinsic parameters of the cameras

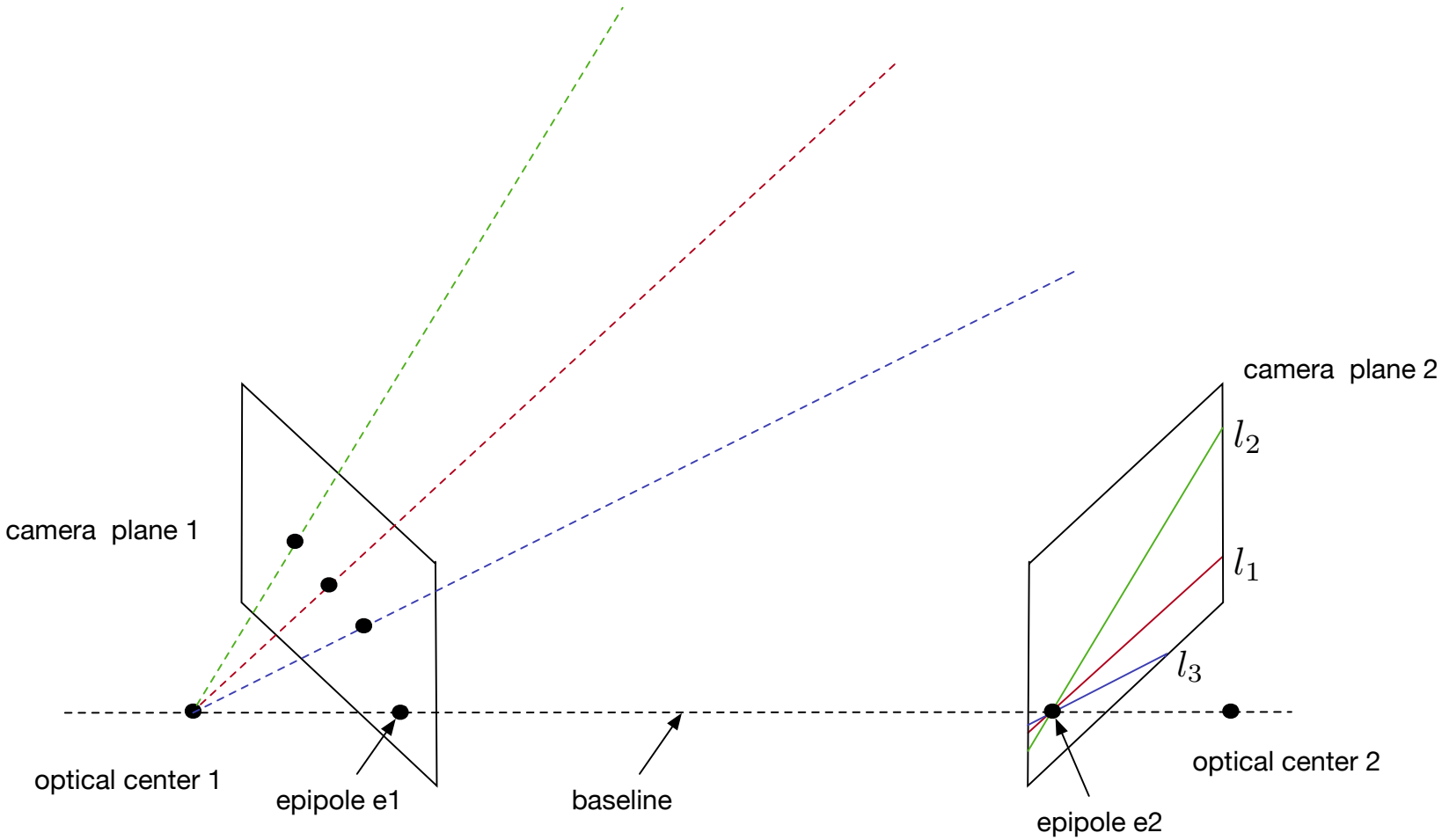
[From [Pri12]]

The epipolar constraint

Practical applications [Pri12]:

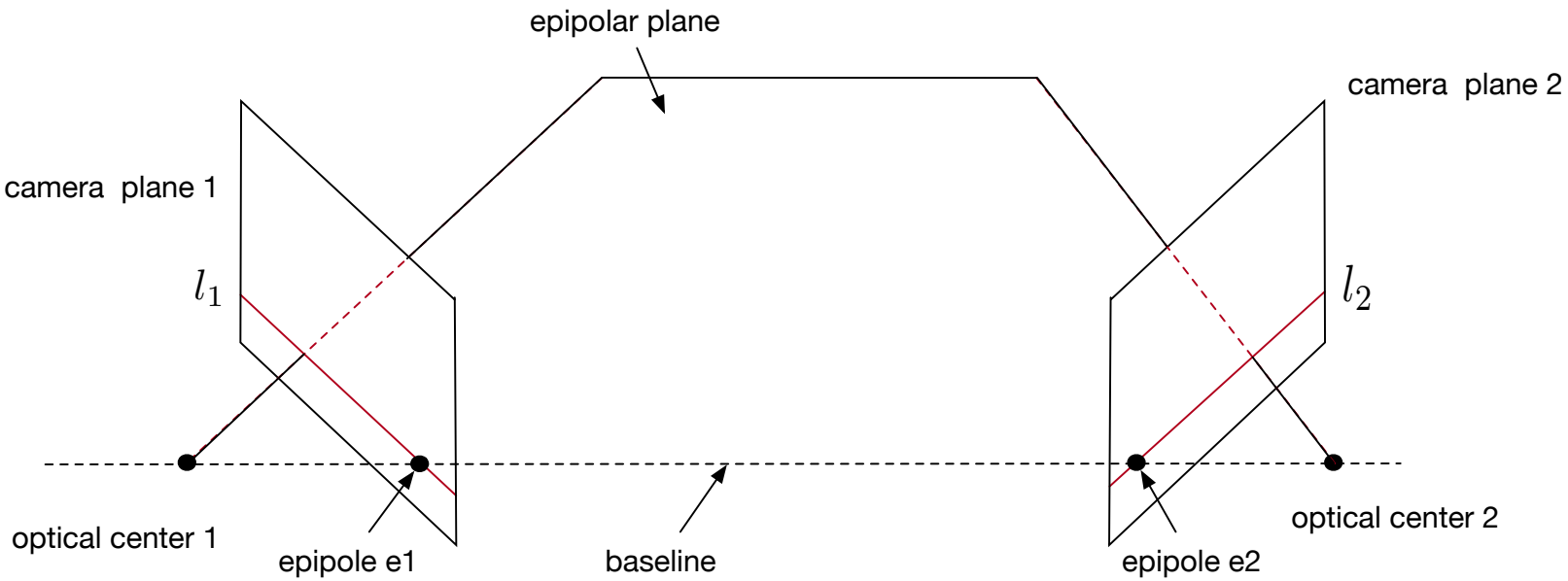
- Finding point correspondences (given intrinsic and extrinsic parameters): for a point in the first image, perform a 1D search along the epipolar line in the second image for the corresponding position
- Constraint on corresponding points is a function of the intrinsic and extrinsic parameters
  $\Rightarrow$ Use the observed pattern of point correspondences to determine the extrinsic parameters
  $\Rightarrow$ Get the geometric relationship between the two cameras

The epipole



[From [Pri12]]

The epipolar plane



[From [Sze10]]

Outline

Normalized coordinates [HZ04]

Let's consider:

- $\tilde{X}_w = \begin{bmatrix} X_w \\ 1 \end{bmatrix}$ a point in the word in homogeneous coordinates
- Camera matrix $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ and $\lambda\tilde{\mathbf{x}}_i = \mathbf{P}\tilde{X}_w$ a point in the image
- $\mathbf{K}$ is known

We can get normalized coordinates:

- $\lambda\hat{\mathbf{x}}_i = \lambda\mathbf{K}^{-1}\tilde{X}_i \Rightarrow \lambda\hat{\mathbf{x}}_i = [\mathbf{R}|\mathbf{t}]\tilde{X}_w$ ($\hat{\mathbf{x}}_i$ stills in homogeneous coordinates)
- equivalent to a camera where $\mathbf{K} = \mathbf{I}$

Normalized camera matrix: $\mathbf{P}' = \mathbf{K}^{-1}\mathbf{P} = [\mathbf{R}|\mathbf{t}]$

Essential matrix [Pri12]

The geometric relationship between the two cameras is captured by the essential matrix.

- Assume normalized cameras, first camera at origin.

$$\lambda_1 \hat{\mathbf{x}}_{i1} = [\mathbf{I}|0]\tilde{X}_w$$
$$\lambda_2 \hat{\mathbf{x}}_{i2} = [\mathbf{R}|\mathbf{t}]\tilde{X}_w$$

- 1^{rst} camera: $\lambda_1 \hat{\mathbf{x}}_{i1} = X_w$
- 2^{nd} camera: $\lambda_2 \hat{\mathbf{x}}_{i2} = \mathbf{R}X_w + \mathbf{t}$
- Substituting:

$$\lambda_2 \hat{\mathbf{x}}_{i2} = \lambda_1 \mathbf{R}\hat{\mathbf{x}}_{i1} + \mathbf{t}$$

Essential matrix [Pri12]

- Constraint between the possible positions of corresponding points in the two images

$$\lambda_2 \hat{\mathbf{x}}_{i2} = \lambda_1 \mathbf{R} \hat{\mathbf{x}}_{i1} + \mathbf{t}$$

- take cross product with $\mathbf{t}$:

$$\lambda_2 \mathbf{t} \times \hat{\mathbf{x}}_{i2} = \lambda_1 \mathbf{t} \times \mathbf{R} \hat{\mathbf{x}}_{i1}$$

- take inner product with $\hat{\mathbf{x}}_{i2}$:

$$\hat{\mathbf{x}}_{i2}^T \mathbf{t} \times \mathbf{R} \hat{\mathbf{x}}_{i1} = 0$$

Essential matrix [Pri12]

$$\hat{\mathbf{x}}_{i2}^T \mathbf{t} \times \mathbf{R}\hat{\mathbf{x}}_{i1} = 0$$

- The cross product term can be expressed as a matrix:

$$\mathbf{t}_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

- Defining the *essential matrix*:

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

- and the essential matrix relation:

$$\hat{\mathbf{x}}_{i2}^T \mathbf{E}\hat{\mathbf{x}}_{i1} = 0$$

Properties of the essential matrix [Pri12]

$$\hat{\mathbf{x}}_{i2}^T \mathbf{E} \hat{\mathbf{x}}_{i1} = 0$$

- Rank 2: $det[\mathbf{E}] = 0$
- 5 D.o.F.
- Non-linear constraint between elements:

$$2\mathbf{E}\mathbf{E}^T\mathbf{E} - trace[\mathbf{E}\mathbf{E}^T]\mathbf{E} = 0$$

Computing the essential matrix

- 5-point algorithm [Nis04]
- 8-point algorithm [Lh81]
  - $\hat{\mathbf{x}}_{i2}^T \mathbf{E} \hat{\mathbf{x}}_{i1} = 0$
  - can be solved with SVD

Recovering epipolar lines [Pri12]

- Equation of a line:

$$ax + by + c = 0$$

- or:

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

- or:

$$l\tilde{x} = 0$$

Recovering epipolar lines [Pri12]

- Equation of a line : $l\tilde{x} = 0$
- Now consider:

$$\hat{\mathbf{x}}_{i2}^T \mathbf{E} \hat{\mathbf{x}}_{i1} = 0$$

- This as the form $l_1 \hat{\mathbf{x}}_{i1} = 0$ where $l_1 = \hat{\mathbf{x}}_{i2}^T \mathbf{E}$
- So the epipolar lines can be expressed as:

$$l_1 = \hat{\mathbf{x}}_{i2}^T \mathbf{E}$$
$$l_2 = \hat{\mathbf{x}}_{i1}^T \mathbf{E}$$

Recovering epipoles [Pri12]

- Every epipolar line in image 1 passes through the epipole e_1

$$\Rightarrow \hat{\mathbf{x}}_{i2}^T \mathbf{E} \tilde{e}_1 = 0$$

for all $\hat{\mathbf{x}}_{i2}^T$

- This can only be true if $\tilde{e}_1$ is in the nullspace of $\mathbf{E}$:

$$\tilde{e}_1 = null[E]$$

- Similarly:

$$\tilde{e}_2 = null[E^T]$$

We find the null spaces by computing the SVD of $\mathbf{E}$:

$$\mathbf{E} = UDV^T$$

and taking $\tilde{e}_1$ the last column of V and $\tilde{e}_2$ the last row of U

Retrieving $\mathbf{R}, \mathbf{t}$ from $\mathbf{E}$ [Pri12]

Relative orientation problem: recover $\mathbf{R}, \mathbf{t}$ from $\mathbf{E}$

- Essential matrix

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

- To recover $\mathbf{R}, \mathbf{t}$ use the matrix

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- With $\mathbf{E} = UDV^T$ we get:

$$\mathbf{t}_\times = UDWU^T$$
$$\mathbf{R} = UW^{-1}V^T$$

(details in [HZ04])

- Need 2 corresponding points to solve ambiguities and have $\mathbf{R}$ and $\mathbf{t}$ where points are in front of both cameras

Outline

Fundamental Matrix [Pri12]

- Lets consider two normal (not normalized) cameras:

$$\lambda_1 \tilde{\mathbf{x}}_1 = \mathbf{K}_1[\mathbf{I}|0]\tilde{X}_w$$
$$\lambda_2 \tilde{\mathbf{x}}_2 = \mathbf{K}_2[\mathbf{R}|\mathbf{t}]\tilde{X}_w$$

- Using a similar procedure we can get the relation:

$$\tilde{\mathbf{x}}_2^T \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} \tilde{\mathbf{x}}_1 = 0$$

- or:

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

- with:

$$\mathbf{F} = \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} = \mathbf{K}_2^{-T} \mathbf{t}_\times \mathbf{R} \mathbf{K}_1^{-1}$$

- Relation between essential and fundamental matrix:

$$\mathbf{E} = \mathbf{K}_2^T \mathbf{F} \mathbf{K}_1$$

Estimate the fundamental matrix [Pri12]

- When the fundamental matrix is correct with $\tilde{\mathbf{x}}_{i2}^T \mathbf{F} \tilde{\mathbf{x}}_{i1} = 0$, the epipolar line induced by a point in the first image should pass through the matching point in the second image and vice-versa
- Constraint parameterized by the nine entries of $\mathbf{F}$
- Criterion: minimize the squared distance between every point and the epipolar line predicted by its match in the other image (I corresponding points):

$$\hat{F} = \arg\min_{\mathbf{F}} \left[\sum_{i=1}^{I} \left((dist[\mathbf{x}_{i1}, l_{i1}])^2 + (dist[\mathbf{x}_{i2}, l_{i2}])^2 \right) \right]$$

- $dist[\mathbf{x}, l] = \frac{ax+by+c}{\sqrt{a^2+b^2}}$ with $l = [a, b, c]$ and $\mathbf{x} = [x, y]^T$

No closed form solution

Estimate the fundamental matrix: the eight-point algorithm [Pri12]

- Approach
  - solve for fundamental matrix using homogeneous coordinates
  - closed form solution (but don't minimize a geometric error but an algebraic error)
  - solution usually very close to the values that optimize the previous cost function
- In homogeneous coordinates:

$$\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_1 = 0 \Rightarrow \begin{bmatrix} x_{i2} & y_{i2} & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_{i1} \\ y_{i1} \\ 1 \end{bmatrix} = 0$$

Estimate the fundamental matrix: the eight-point algorithm [Pri12]

- Can be expressed as: $A\mathbf{f} = 0$ with $\mathbf{f} = [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^T$ and A contains the combination of at least 8 pairs of points coordinates
- Find minimum of $|A\mathbf{f}|^2$ subject to $|\mathbf{f}| = 1$
- Solution can be found by SVD of A: $A = UDV^T$, setting $\mathbf{f}$ to the last column of V
- Reform $\mathbf{F}$ form $\mathbf{f}$

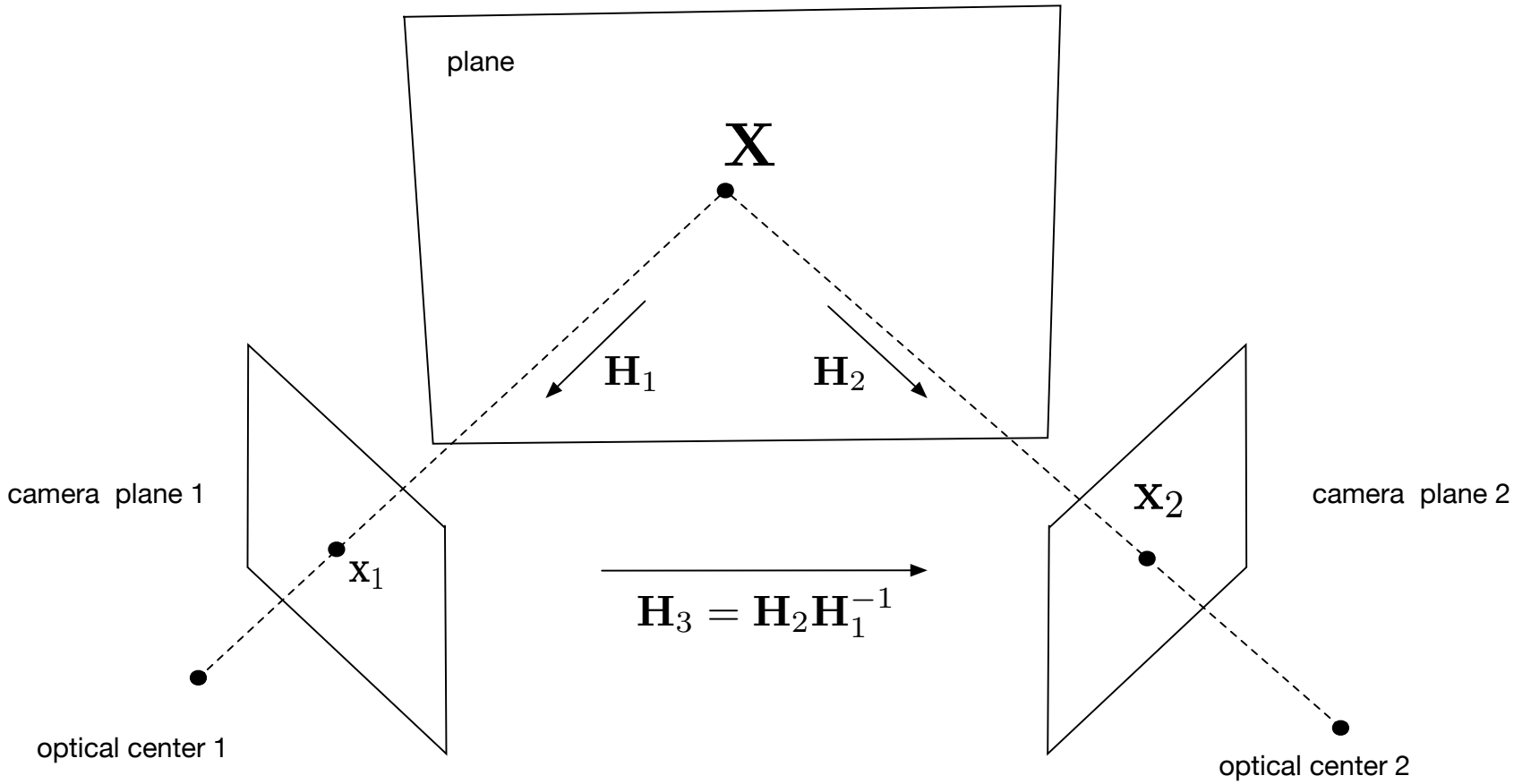
Estimate the fundamental matrix: the eight-point algorithm [Pri12]

- This procedure does not ensure that solution is rank 2.
  Solution: set last singular value to zero.
- Can be unreliable because of numerical problems to do with the data scaling $\rightarrow$ better to re-scale the data first
- Needs 8 points in general positions (cannot all be planar).
- Fails if there is not sufficient translation between the views

Outline

Outline

Homography: definition and properties



[From [Pri12]]

Homography : definition and properties

- Homography mapping $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ linear in homogeneous coordinates

$$\lambda \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

- images seen by different cameras with the *pinhole in the same place* are related by homographies
- special case **pure rotation**: If the camera rotates but *does not translate*, the tomography mapping image 1 to image 2 is written:

$$\mathbf{H} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}$$

with $\mathbf{K}$ intrinsic matrix and $\mathbf{R}$ rotation between the two camera positions

Homography : definition and properties

Homography maps between [Pri12]:

- points on a plane in the world and their positions in an image,
- points in two different images of the same plane
- two images of a 3D object where the camera has rotated but not translated

In the planar case, we can chain the homographies between consecutive frames.

Outline

Homography estimation [HZ04]

Objective:

Given $n \geq 4$ 2D to 2D point correspondences $\mathbf{x'}_i \leftrightarrow \mathbf{x'}_i$, determine the 2D homography matrix $\mathbf{H}$ such as $\mathbf{x'}_i = \mathbf{H}\mathbf{x}_i$

Algorithm

1. **Normalization of $\mathbf{x}_i$:** $\tilde{\mathbf{x}}_i = T\mathbf{x}_i$
2. **Normalization of $\mathbf{x'}_i$:** $\tilde{\mathbf{x}'}_i = T\mathbf{x'}_i$
3. **DLT**:
   1. For each correspondence $\tilde{\mathbf{x}'}_i \leftrightarrow \tilde{\mathbf{x}}_i$ compute the matrix A_i.
   2. Form the $2n \times 9$ matrix A
   3. Write $\tilde{\mathbf{h}}$ for the vector containing the entries of the matrix $\tilde{\mathbf{H}}$. A solution of $A\tilde{\mathbf{h}} = 0$, subject to $||\tilde{h}|| = 1$, is obtained from the unit singular vector of A corresponding to the smallest singular value.
   4. The matrix $\tilde{\mathbf{H}}$ is determined from $\tilde{\mathbf{h}}$
4. **Denormalization**: Set $\mathbf{H} = T'^{-1}\tilde{\mathbf{H}}T$

Outline

From homography to pose computation

- All points are in the same plane, then fix $z_w = 0$
- $\Rightarrow$ each 3D point coordinate is given by $X_w = \begin{bmatrix} x_w \\ y_w \\ 0 \\ 1 \end{bmatrix}$
- Their projections in the image plane are given by

$$X_c = [\mathbf{I}|0][\mathbf{R}|\mathbf{t}] \begin{bmatrix} x_w \\ y_w \\ 0 \\ 1 \end{bmatrix} = [\mathbf{I}|0][r_1 r_2 \mathbf{t}] \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix}$$

  with r_i the i^{th} column of $\mathbf{R}$
- $\mathbf{H}$ can be computed using DLT

From homography to pose computation

- Knowing $\mathbf{H}$, $\mathbf{R}, \mathbf{t}$ can be computed noting:

$$[r_1 r_2 \mathbf{t}] = [\mathbf{I}|0]^{-1}\mathbf{H}$$

- $\mathbf{R}$ is orthogonal then : $r_3 = r_1 \times r_2$

Outline

Outline

Feature point matching principle

- Finding corresponding points (2D point in the image and a 3D reference or between two 2D images)
- Common framework:
  - keypoints extraction: subset of pixels ("cornerness")
  - description: conversion into a descriptor
  - matching between descriptors
- Common process:
  - off-line :
    - ⋆ keypoint descriptors computed off-line on a reference model
    - ⋆ key-point storage
  - on-line:
    - ⋆ extract keypoint from each image
    - ⋆ match in the descriptor space with those in the database
    - ⋆ from correspondences, compute camera pose

Feature points detectors properties

- Repeatability
- Invariance
- Robustness
- Distinctiveness/informativeness
- Locality
- Quantity
- Accuracy
- Efficiency

Outline

Corner detector

- Moravec [Mor80], Harris corner-detector [HS88], Shi-Tomasi [ST02]
- Harris corner-detector: Studies the average variation in intensity for a small movement

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

  - E: the difference between the original and the moved window.
  - u, v: window's displacement in the x (resp. y) direction
  - $w(x,y)$: window function
  - I: image intensity at a position (x,y)
  - $I(x,y)$: original intensity
  - $I(x+u, y+v)$: shifted intensity

Corner detector

- Expansion using Talor series:

$$E(u,v) \approx \sum_{x,y} w(x,y)[I(x,y) + uI_x + vI_y - I(x,y)]^2$$
$$\approx \sum_{x,y} w(x,y)[u^2 I_x^2 + 2uvI_x I_y + v^2 I_y^2]$$

  with I_x gradient along x, I_y gradient along y

- in matrix form

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$
$$\approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

- M: symmetric, define, positive $\Rightarrow$ eigenvalues decomposition

Corner detector

- A score, R, is calculated for each window:

$$R = det(M) - k(trace(M))^2$$

  with $det(M) = \lambda_1 \lambda_2$ and $trace(M) = \lambda_1 + \lambda_2$
- R values are > 0 around a corner, < 0 around an edge and small in a constant region

Note: Shi-Tomasi detector has improved Harris detector

Outline

SIFT [Low04]

- SIFT: Distinctive image features from scale invariant keypoint (1999) [Low04]
- has been considered a breakthrough for 2D points matching
- Robust to
  - Scale
  - Rotation
  - Illumination
  - Viewpoint

SIFT [Low04] algorithm principle

1. Constructing a scale space
2. Laplacian of Gaussian (LoG) approximation using Difference of Gaussian (DoG)
3. Finding keypoints
4. Get rid of bad key points (edges and low contrast regions)
5. Assigning an orientation to the keypoints (cancels out the effect of orientation)
6. Generate SIFT features

A detailed explanation can be found in [ROD14].

Outline

Feature detectors

- FAST [ST02]
- SIFT [Low04]
- SURF [BETV08]
- ORB [RRKB11]
- KAZE [ABD12]
- CARD [ABD12]
- BRIEF [CLÖ+12]
- BRISK [LCS11]
- FREAK [AOV12]
- LDB [XK12]
- ...

Demo : Learning OpenCV3 example code

▸ https://github.com/oreillymedia/Learning-OpenCV-3_examples/blob/master/example_16-02.cpp

Outline

Remove outliers

- Causes of outliers: image noise, occlusions, blur, changes in view point or illumination (non accounted by the feature detector)
- Use of RANSAC algorithm [FB81]
  1. Initial: let A be a set of N feature correspondences
  2. repeat
     1. Randomly select a sample of s points from A
     2. Fit a model to these points
     3. Compute the distance of all other points to this model
     4. Construct the inlier set (i.e. count the number of points whose distance from the model $< d$)
     5. Store these inliers
     6. until maximum number of iterations reached
  3. The set with the maximum number of inliers is chosen as a solution to the problem
  4. Estimate the model using all the inliers

Remove outliers

- The number of iterations N which ensures a probability p that at least one sample with only inliers is drawn is given by

$$N = \frac{log(1-p)}{log(1-(1-\nu)^s)}$$

with

  - ν the probability that a correspondence is an outlier
  - s the number of point from which the model can be instantiated

- Example: p=99%, s=5, ν=50% $\Rightarrow$ N=145

Outline

3D reconstruction principle pipeline [Pri12]

1. Compute image features
2. Compute feature descriptors
3. Find initial matches
4. Compute fundamental matrix
5. Refine matches
6. Estimate essential matrix
7. Decompose essential matrix (four possibles solutions)
8. Estimate 3D points.

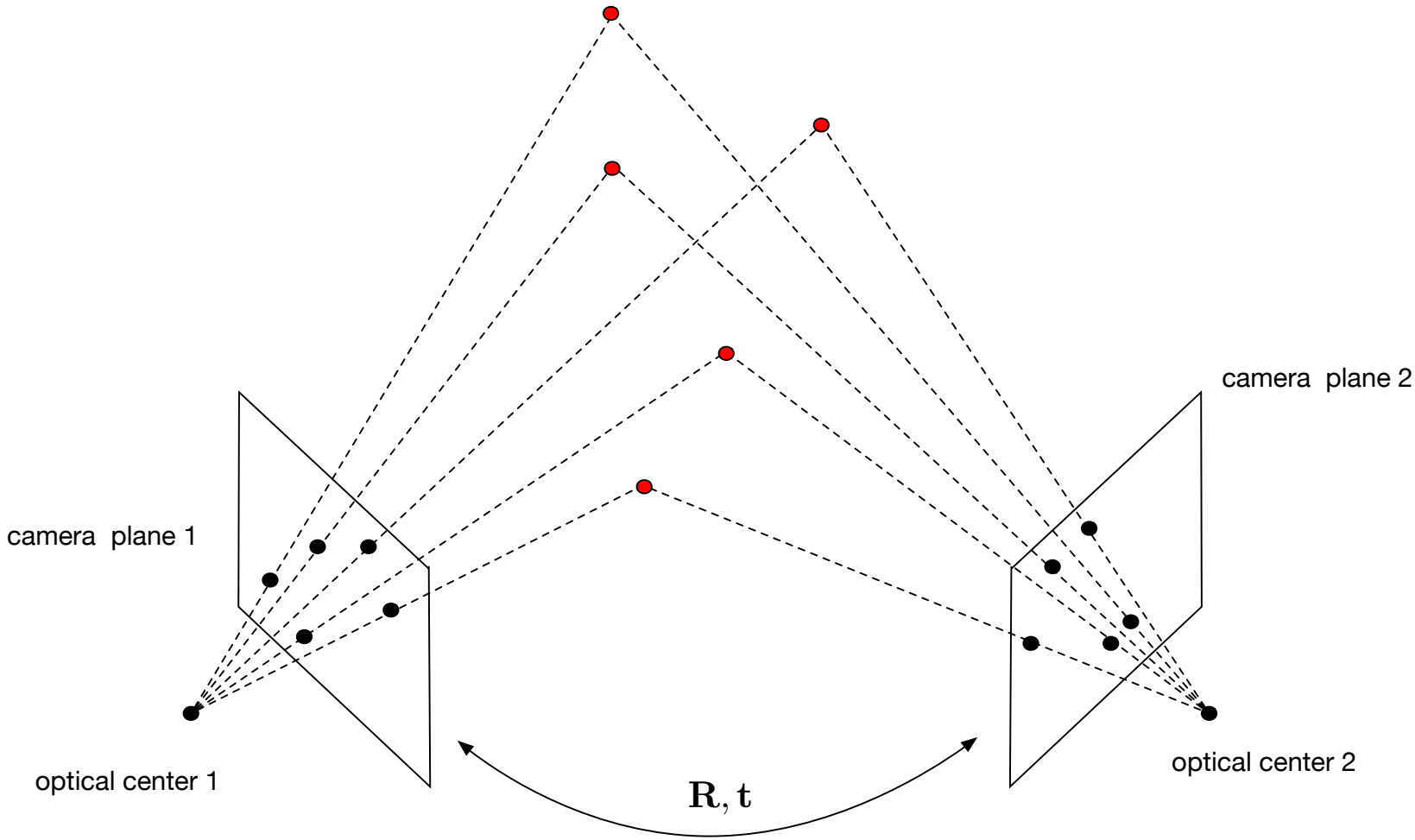
Outline

Outline

Principle

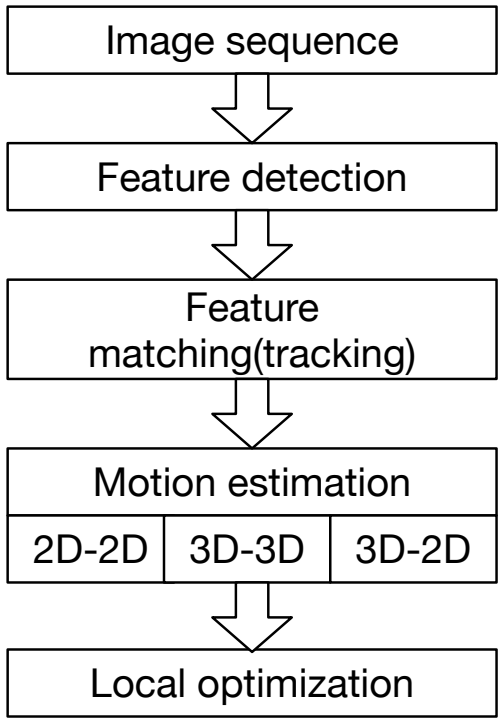

Computer Vision
Algorithms and Applications
Multiple View Geometry
in computer vision
Richard Hartley and Andrew Zisserman
SECOND EDITION

Principle


camera plane 1
camera plane 2
optical center 1
optical center 2
$\mathbf{R}, \mathbf{t}$

Principle

Visual Odometry (VO) compute the camera path incrementally



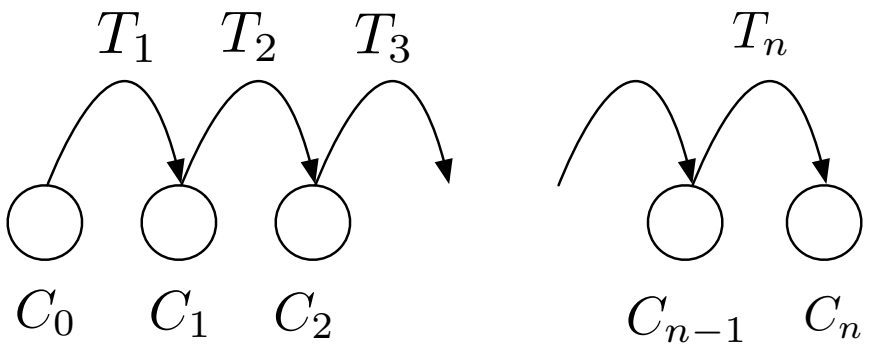
[From [SF11]]

Problem formulation [SF11]

- Two camera position at adjacent timestamps $k-1$ and k are related by

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

- $T_{0:n} = T_1, \dots T_n$ contents to all subsequent motions
- The set of camera poses $C_{0:n} = C_0, \cdots, C_n$ contains the transformation of the camera w.r.t. $\mathcal{F}_{w0}$ at $k = 0$
- $C_n = C_{n-1} T_n = C_0 T_1 \cdots T_n$



Problem formulation [SF11]

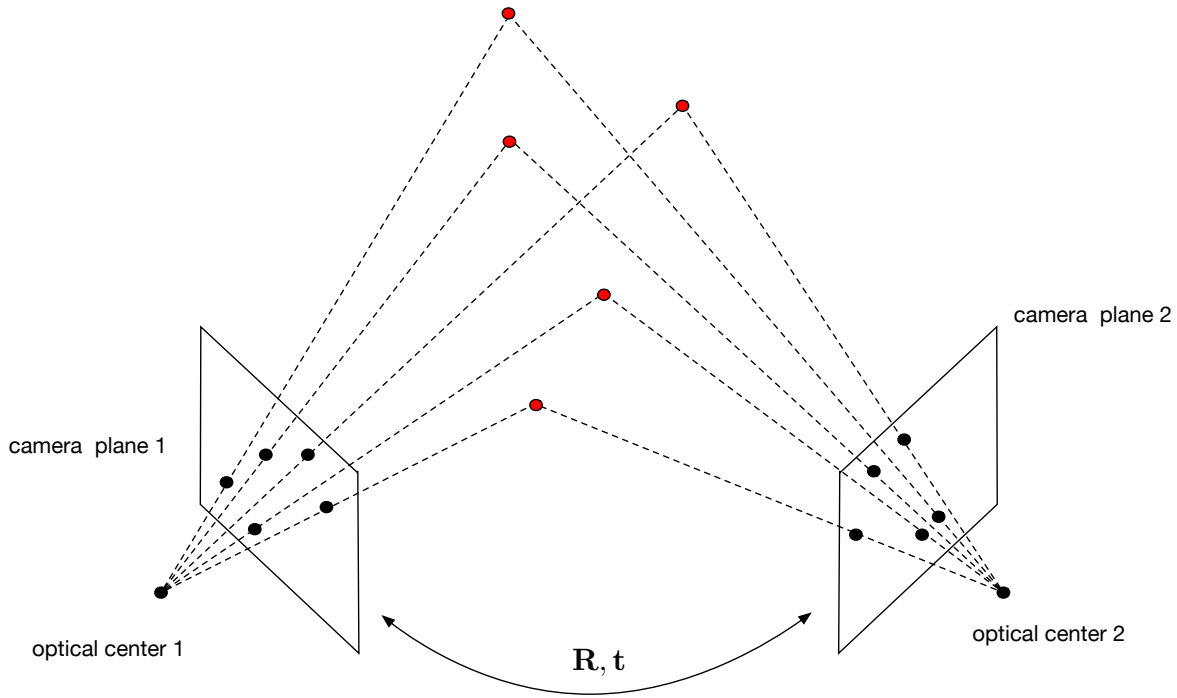
VO tasks:

- compute the relative transformations T_k from the images I_k and I_{k-1}
- concatenate the transformations to recover the full trajectory $C_{0:n}$ of the camera
- optionally perform an iterative refinement over last m poses to estimate local trajectory with more accuracy (bundle adjustment)

Outline

Outline

2D-2D [SF11]



- Both images features are specified in 2D
- The minimal-case solution involve 5-point correspondences
- The solution is found by determining the transformation that minimizes the reprojection error of the triangulated points in each image

2D-2D relative scale [SF11]

- Absolute scale of the translation cannot be computed from two images
- Triangulate 3D points position from 2D points pairs
- From 3D points, the relative distances between any combination of two 3-D points can be computed.
- Scale can then be determined from the distance ratio r between a point pair in X_{k-1} and a pair in X_k

$$r = \frac{||X_{k-1,i} - X_{k-1,j}||}{||X_{k,i} - X_{k,j}||}$$

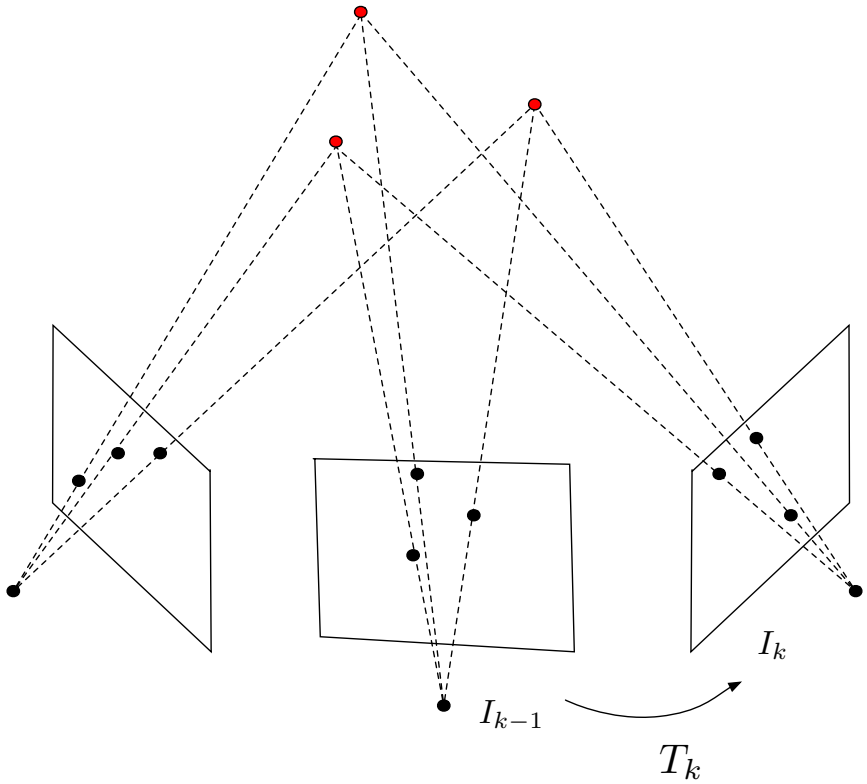
- Mean scale ratio is used to scale $\mathbf{t}$

Algorithm: VO for 2D-2D correspondence [SF11]

1. Capture new frame I_k
2. Extract and match features between I_{k-1} and I_k
3. Compute essential matrix for image pair I_{k-1} and I_k
4. Decompose essential matrix into $\mathbf{R}_k$ and $\mathbf{t}_k$ and form T_k
5. Compute relative scale and rescale $\mathbf{t}_k$ accordingly
6. Concatenate transformation by computing $C_k = C_{k-1}T_k$
7. Repeat from 1

Outline

3D-3D [SF11]



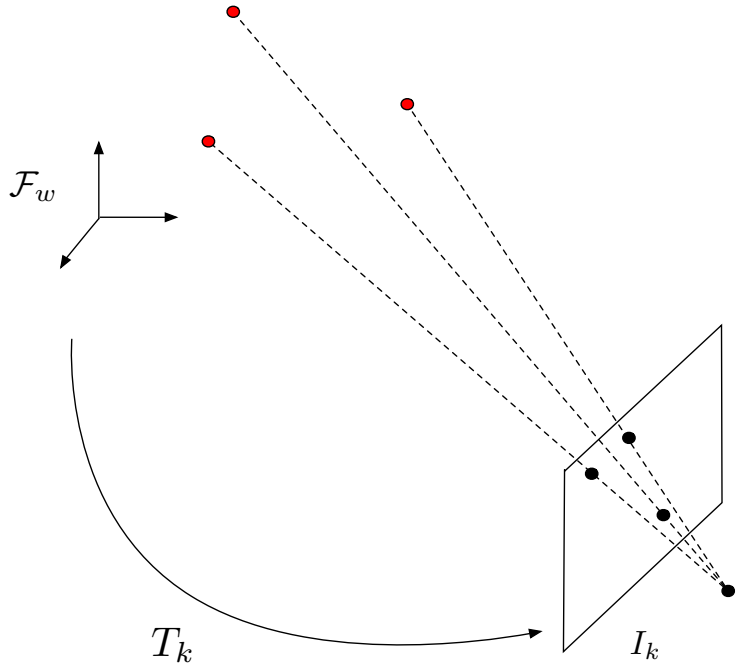
- Both image features are specified in 3D
- The minimal-case solution involve 3 non-linear correspondences
- The solution is found by determining the aligning transformation that minimizes the 3D-3D distance

Algorithm: VO for 3D-3D correspondence [SF11]

1. Capture two stereo image pairs $I_{l,k-1}, I_{r,k-1}$ and $I_{l,k}, I_{r,k}$
2. Extract and match features between $I_{l,k-1}$ and $I_{l,k}$
3. Triangulate matched features for each stereo pair
4. Compute T_k from 3-D features X_{k-1} and X_k
5. Concatenate transformation by computing $C_k = C_{k-1}T_k$
6. Repeat from 1

Outline

2D-3D [SF11]

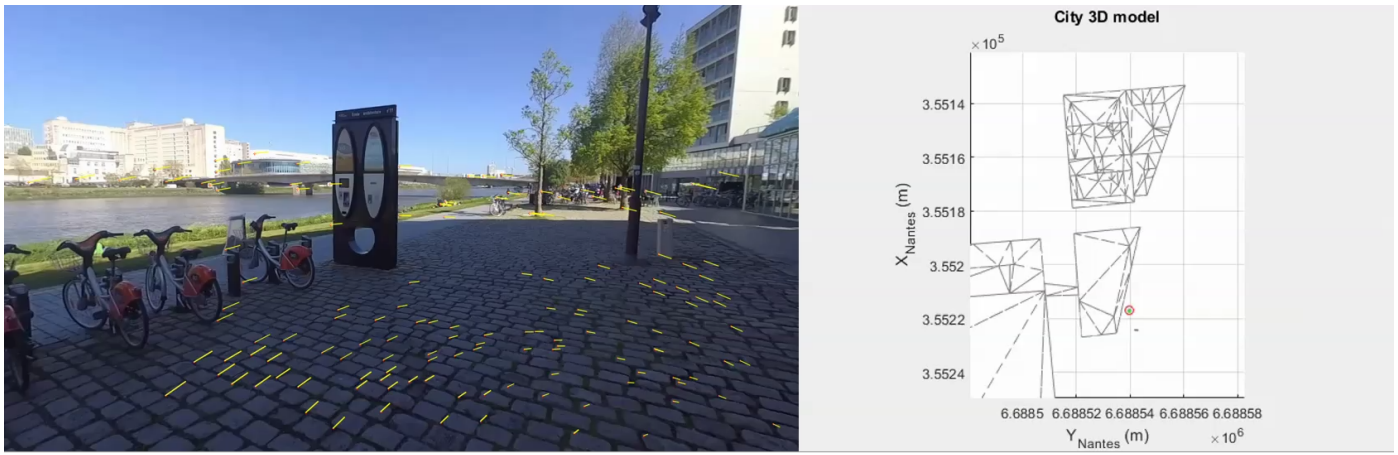


- Features at instant $k-1$ are in 3D and in 2D at k
- PnP problem
- The minimal-case solution involve 3 non-linear correspondences
- The solution is found by determining the transformation that minimizes the reprojection error

Algorithm: VO for 2D-3D correspondence [SF11]

1. Do only once:
   1. Capture two frames I_{k-2}, I_{k-1}
   2. Extract and match features between them
   3. Triangulate features from I_{k-2}, I_{k-1}
2. Do at each iteration:
   1. Capture new frame I_k
   2. Extract features and match with previous frame I_{k-1}
   3. Compute camera pose (PnP) from 3-D-to-2-D matches
   4. Triangulate all new feature matches between I_k and I_{k-1}
   5. Iterate

VO trajectory estimation example



work from [ASR18]

Outline

Outline

VO vs. SLAM

- VO
  - incremental localization
  - local consistent estimate of the trajectory
  - potential windowed bundles adjustments
- SLAM: *Simultaneous Localization And Mapping*
  - global consistent estimate of the localization (and mapping)
  - use loop-closure to reduce the drift in the map and in the localization of the camera (global bundle adjustment)
- VO can be a part of SLAM (before closing the loop)

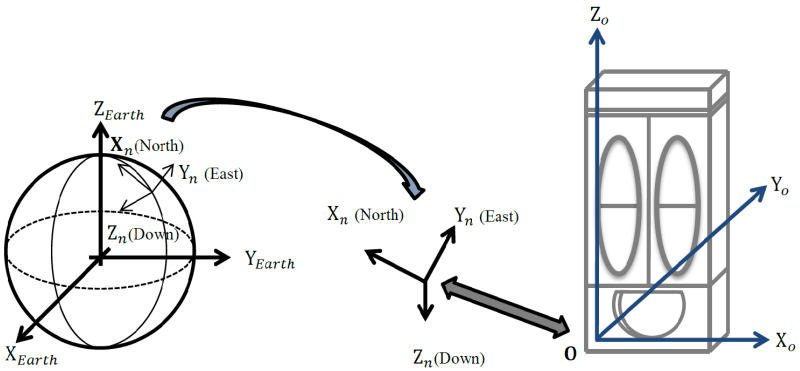
Outline

vSLAM and viSLAM algorithms

- Visual SLAM
  - PTAM [KM07]
  - ORB-SLAM [MAMT15]
  - DTAM [NLD11]
  - LSD-SLAM [ESC14]
  - DSO [EKC16]
  - ...
- Visual inertial SLAM
  - MSCKF [MR07]
  - ROVIO [BOHS15]
  - S-MSCKF [SMP+17]
  - VINS-Mono [QLS17]
  - ...

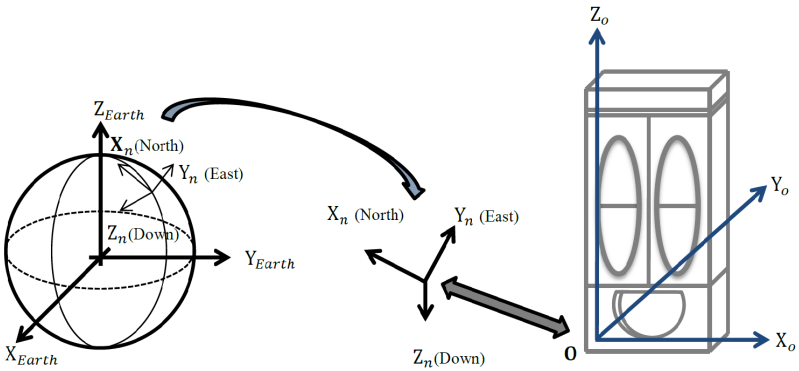
Outline

Conclusion



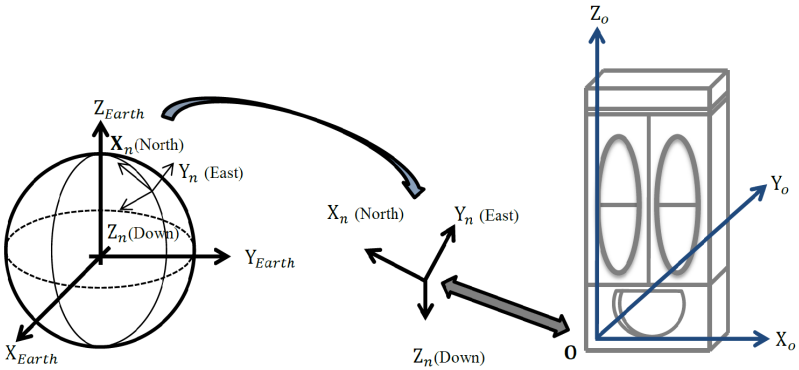
Images from [ASR17]

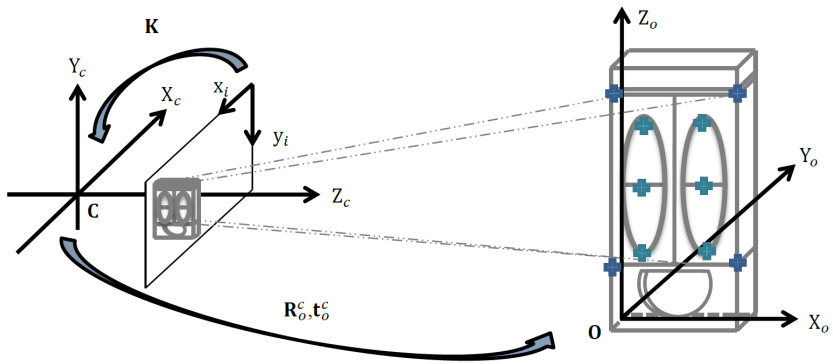
Conclusion



Images from [ASR17]

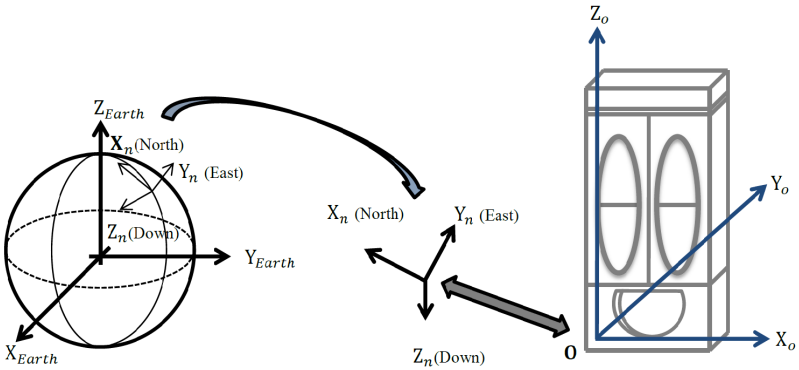
Conclusion

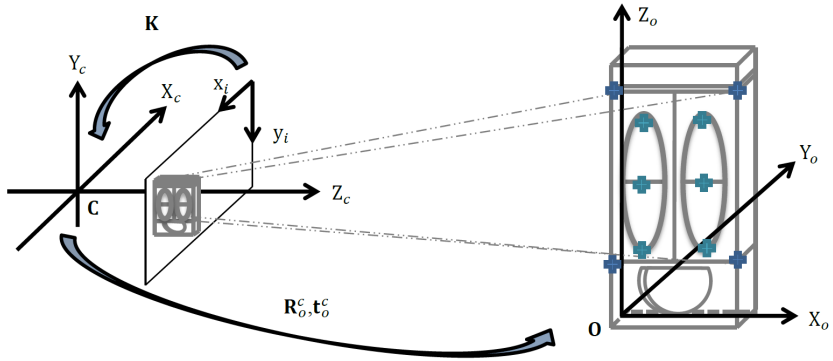


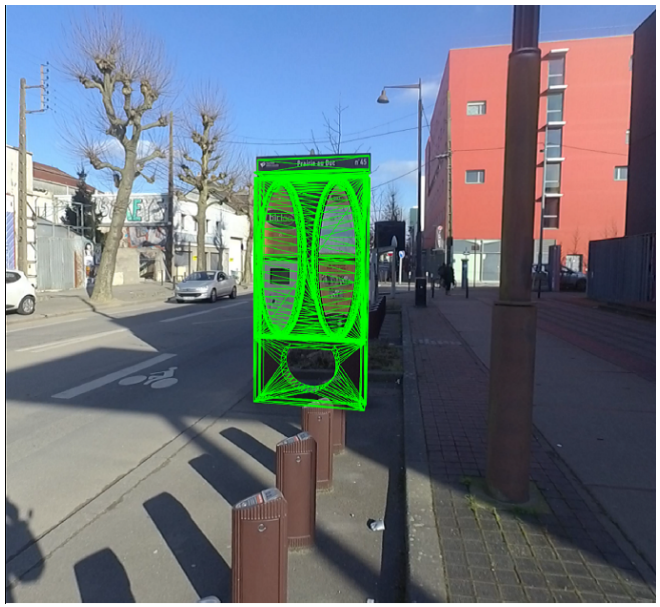


Images from [ASR17]

Conclusion

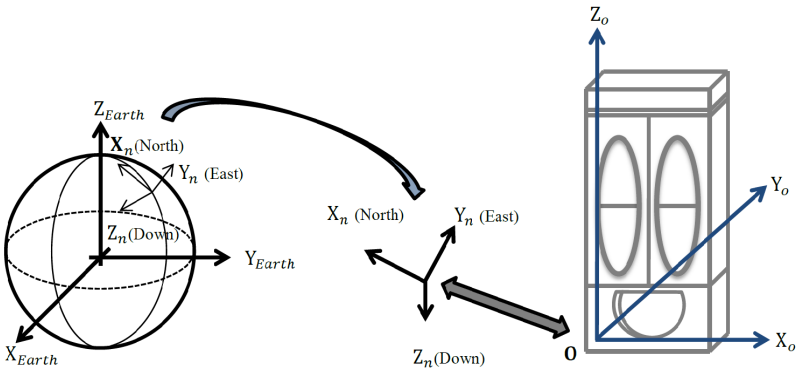


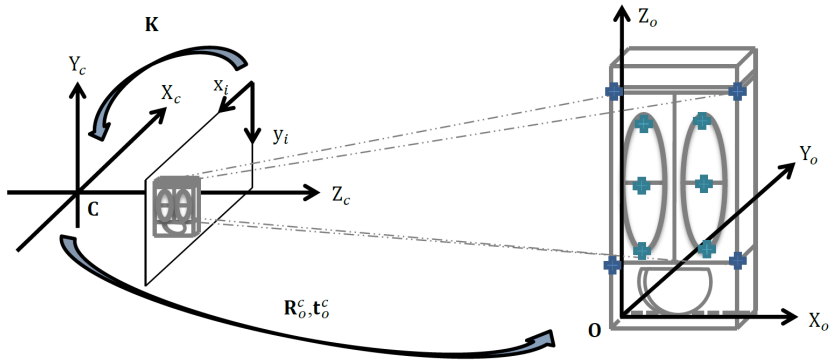




Images from [ASR17]

Conclusion





Images from [ASR17]

Conclusion

- But... pose calculation is not very accurate in the presence of fast rotation movements
- ... fusion with IMMU can help!

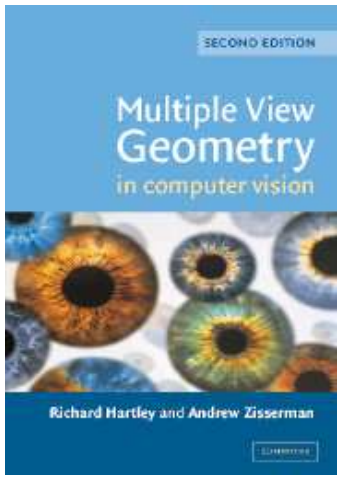
Outline

Links

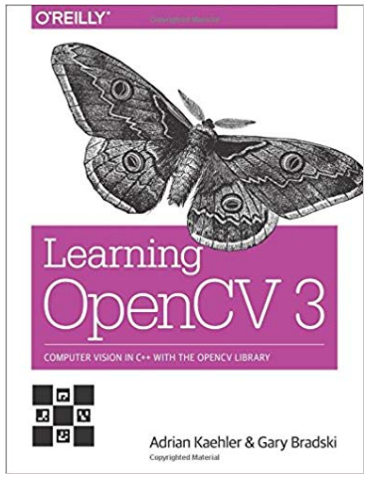
- Tools for Computer Vision
  - OpenCV https://opencv.org *(Open source)*
  - OpenGV https://laurentkneip.github.io/opengv/index.html *(Open source)*
  - ViSP https://visp.inria.fr *(Open source)*
  - MATLAB Toolboxes https://fr.mathworks.com/solutions/image-video-processing.html
- Websites
  - Annotated Computer Vision Bibliography
    http://www.visionbib.com/bibliography/contents.html
  - Computer Vision conferences list
    http://conferences.visionbib.com/Iris-Conferences.html

(very) Useful books and tutorials

- **Multiple View Geometry in Computer Vision** [HZ04]
  From Richard Hartley, Andrew Zisserman
  ▸ Book site



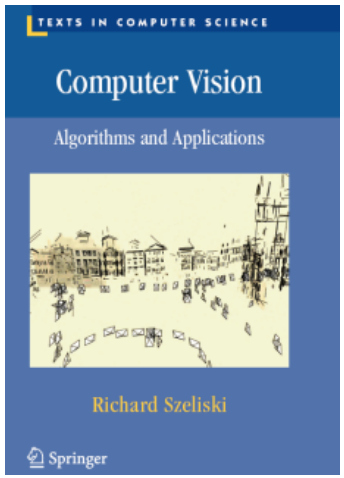
- **Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library** [KB17]
  From Adrian Kaehler and Gary Bradski



(very) Useful books and tutorials

- **Computer Vision: Models Learning and Inference** [Pri12]
  From Simon J.D. Prince,
  - http://www.computervisionmodels.com

- **Computer Vision: Algorithms and Applications** [Sze10]
  From Richard Szeliski
  - http://szeliski.org/Book/



(very) Useful books and tutorials

- E. Marchand, H. Uchiyama, F. Spindler "Pose estimation for augmented reality : a hands-on survey", IEEE Trans. Vis. Comput. Graph., 2016. [MUS16]
  - http://people.rennes.inria.fr/Eric.Marchand/pose-estimation/index.html
- D. Scaramuzza and F. Fraundorfer, "Visual Odometry: Part I - The First 30 Years and Fundamentals," IEEE Robot. Autom. Mag., vol. 18, no. 4, pp. 80-92, Dec. 2011. [SF11]
- F. Fraundorfer and D. Scaramuzza, "Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications," IEEE Robot. Autom. Mag., vol. 19, no. 2, pp. 78-90, Jun. 2012. [FS12]
  - http://rpg.ifi.uzh.ch/visual_odometry_tutorial.html

So now, if you know what you look at, you can tell me where you are.

References I

Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison.

KAZE features.

Lecture Notes in Computer Science, 7577 LNCS(PART 6):214–227, 2012.

247

Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst.

FREAK: Fast Retina Keypoint.

2012 Ieee Conference on Computer Vision and Pattern Recognition (Cvpr), pages 510–517, 2012.

247

Nicolas Antigny, Myriam Servieres, and Valerie Renaudin.

Pedestrian track estimation with handheld monocular camera and inertial-magnetic sensor for urban augmented reality.

In *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8, Sapporo, Japan, sep 2017. IEEE.

9, 11, 278, 279, 280, 281, 282

Nicolas Antigny, Myriam Servières, and Valérie Renaudin.

Continuous Pose Estimation for Urban Pedestrian Applications on Hand-held Mobile Device.

In *IPIN*, page to be published, 2018.

10, 271

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool.

Speeded-Up Robust Features (SURF).

Computer Vision and Image Understanding, 110(3):346–359, 2008.

247

M. Bloesch, S. Omari, M. Hutter, and R. Siegwart.

Robust visual inertial odometry using a direct ekf-based approach.

In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 298–304, Sept 2015.

276

References II

Michael Calonder, Vincent Lepetit, Mustafa Özuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua.

BRIEF: Computing a local binary descriptor very fast.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(7):1281–1298, 2012.

247

Daniel F. Dementhon and Larry S. Davis.

Model-based object pose in 25 lines of code.

International Journal of Computer Vision, 15(1-2):123–141, jun 1995.

165, 166, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 193

Jakob Engel, Vladlen Koltun, and Daniel Cremers.

Direct sparse odometry.

CoRR, abs/1607.02565, 2016.

276

Jakob Engel, Thomas Schöps, and Daniel Cremers.

Lsd-slam: Large-scale direct monocular slam.

In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham, 2014. Springer International Publishing.

276

Martin a Fischler and Robert C Bolles.

Random Sample Consensus: A Paradigm for Model Fitting with.

Communications of the ACM, 24:381–395, 1981.

249

Friedrich Fraundorfer and Davide Scaramuzza.

Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications.

IEEE Robotics & Automation Magazine, 19(2):78–90, jun 2012.

288

References III

Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng.

Complete solution classification for the P3P problem.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(8):930–943, 2003.

169, 170

C Harris and M Stephens.

A Combined Corner and Edge Detection.

In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

240

Richard Hartley and Andrew Zisserman.

Multiple view geometry in computer vision.

2004.

61, 67, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 149, 150, 151, 152, 209, 218, 231, 286

Adrian Kaehler and Gary Bradski.

Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library.

O'Reilly Media, 2017.

286

Laurent Kneip, Paul Furgale, and Roland Siegwart.

Using multi-camera systems in robotics: Efficient solutions to the NPnP problem.

Proceedings - IEEE International Conference on Robotics and Automation, (2004):3770–3776, 2013.

194

Laurent Kneip, Hongdong Li, and Yongduek Seo.

UPnP: An optimal O(n) solution to the absolute pose problem with universal applicability.

Lecture Notes in Computer Science, 8689 LNCS(PART 1):127–142, 2014.

194, 195

References IV

G. Klein and D. Murray.

Parallel tracking and mapping for small ar workspaces.

In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, Nov 2007.

276

Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart.

BRISK: Binary Robust invariant scalable keypoints.

In *2011 International Conference on Computer Vision*, pages 2548–2555. IEEE, nov 2011.

247

V. Lepetit, F.Moreno-Noguer, and P.Fua.

Epnp: An accurate o(n) solution to the pnp problem.

International Journal Computer Vision, 81(2), 2009.

165, 166, 171, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194

H. C. Longuet-higgins.

A computer algorithm for reconstructing a scene from two projections.

Nature, 293(5828):133–135, 1981.

214

Chien Ping Lu, Gregory D. Hager, and Eric Mjolsness.

Fast and globally convergent pose estimation from video images.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(6):610–622, 2000.

195

David G Lowe.

Perceptual Organization and Visual Recognition.

1985.

154

References V

David G. Lowe.

Fitting Parameterized Three-Dimensional Models to Images.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(5):441–450, 1991.
154

David G Lowe.

Distinctive image features from scale-invariant keypoints.
International journal of computer vision, 60.2:91–110, 2004.
244, 245, 247

R. Mur-Artal, J. M. M. Montiel, and J. D. Tards.

Orb-slam: A versatile and accurate monocular slam system.
IEEE Transactions on Robotics, 31(5):1147–1163, Oct 2015.
276

Philippe Martinet.

Computer Vision - Visual Geometry, EMARO Erasmus Mundus Master, 2013.
120, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 143, 144, 145, 146, 147, 183

Hans Peter Moravec.

Obstacle avoidance and navigation in the real world by a seeing robot rover.
tech. report CMU-RI-TR-80-03, page 175, 1980.
240

A. I. Mourikis and S. I. Roumeliotis.

A multi-state constraint kalman filter for vision-aided inertial navigation.
In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, April 2007.
276

References VI

Eric Marchand, Hideaki Uchiyama, and Fabie Spindler.

Pose estimation for augmented reality : a hands-on survey.

IEEE transactions on visualization and computer graphics, 2016.

155, 288

D. Nister.

An efficient solution to the five-point relative pose problem.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(6):756–770, jun 2004.

214

Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison.

Dtam: Dense tracking and mapping in real-time.

2011 International Conference on Computer Vision, pages 2320–2327, 2011.

276

Denis Oberkampf, Daniel F. DeMenthon, and Larry S. Davis.

Iterative Pose Estimation Using Coplanar Feature Points.

Computer Vision and Image Understanding, 63(3):495–511, 1996.

184

S.J.D. Prince.

Computer Vision: Models Learning and Inference.

Cambridge University Press, 2012.

34, 35, 36, 200, 201, 202, 203, 204, 205, 206, 210, 211, 212, 213, 215, 216, 217, 218, 220, 221, 222, 223, 224, 227, 229, 252, 287

Tong Qin, Peiliang Li, and Shaojie Shen.

Vins-mono: A robust and versatile monocular visual-inertial state estimator.

CoRR, abs/1708.03852, 2017.

276

References VII

Lawrence GIlman Roberts.

Machine perception of three-dimensional solids.
PhD thesis, 1963.
154

Ives Rey-Otero and Mauricio Delbracio.

Anatomy of the SIFT Method.
Image Processing On Line, 4:370–396, 2014.
245

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski.

ORB: An efficient alternative to SIFT or SURF.
In *2011 International Conference on Computer Vision*, pages 2564–2571. IEEE, nov 2011.
247

Davide Scaramuzza and Friedrich Fraundorfer.

Visual Odometry: Part I - The First 30 Years and Fundamentals.
IEEE Robotics & Automation Magazine, 18(4):80–92, dec 2011.
257, 258, 259, 262, 263, 264, 266, 267, 269, 270, 288

Johannes Lutz Schönberger and Jan-Michael Frahm.

Structure-from-motion revisited.
In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
13

P.F. Sturm and S.J. Maybank.

On plane-based camera calibration: A general algorithm, singularities, applications.
In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, pages 432–437. IEEE Comput. Soc, 1999.
156, 157

References VIII

Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J. Taylor, and Vijay Kumar.
Robust stereo visual inertial odometry for fast autonomous flight.
CoRR, abs/1712.00036, 2017.
276

Jianbo Shi and Carlo Tomasi.
Good features to track.
In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, volume 54, page 258. IEEE, 2002.
240, 247

Richard Szeliski.
Computer Vision: Algorithms and Applications.
2010.
60, 61, 67, 207, 287

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm.
Pixelwise view selection for unstructured multi-view stereo.
In *European Conference on Computer Vision (ECCV)*, 2016.
13

Roger Y. Tsai.
A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses.
IEEE Journal on Robotics and Automation, 3(4):323–344, 1987.
154

References IX

S. Urban, J. Leitloff, and S. Hinz.

MLPNP - A real-time maximum likelihood solution to the perspective-n-point problem.

ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, III-3:131–138, 2016.

194, 195

Xin Yang and Kwang-Ting Cheng.

LDB: An ultra-fast feature for scalable Augmented Reality on mobile devices.

In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, volume 2, pages 49–57. IEEE, nov 2012.

247

Joseph S.C. Yuan.

A General Photogrammetric Method for Determining Object Position and Orientation.

IEEE Transactions on Robotics and Automation, 5(2):129–142, 1989.

154

Z Zhang.

A flexible new technique for camera calibration.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330–1334, 2000.

154, 156, 157

Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Astrom, and Masatoshi Okutomi.

Revisiting the PnP problem: A fast, general and optimal solution.

Proceedings of the IEEE International Conference on Computer Vision, pages 2344–2351, 2013.

194