

TRaitement des Images pour la Vision Artificielle

Vincent Lepetit

slides in part based on material from Mathieu Aubry, Andrew Zisserman, David Lowe

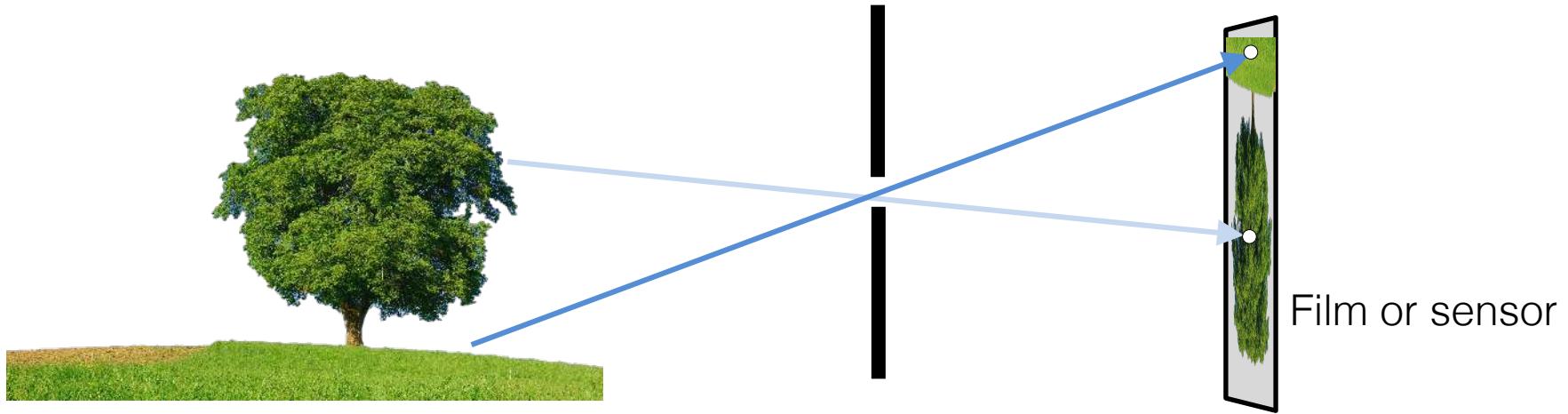
LIGM-Imagine Lab

3D Geometry for Computer Vision

Camera Model

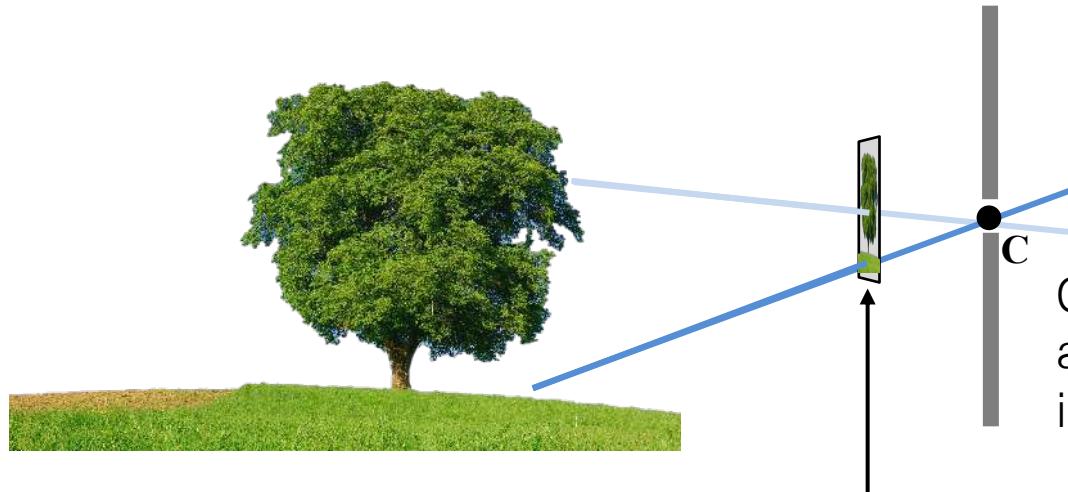
From World to Images

Image Formation: Pinhole Camera



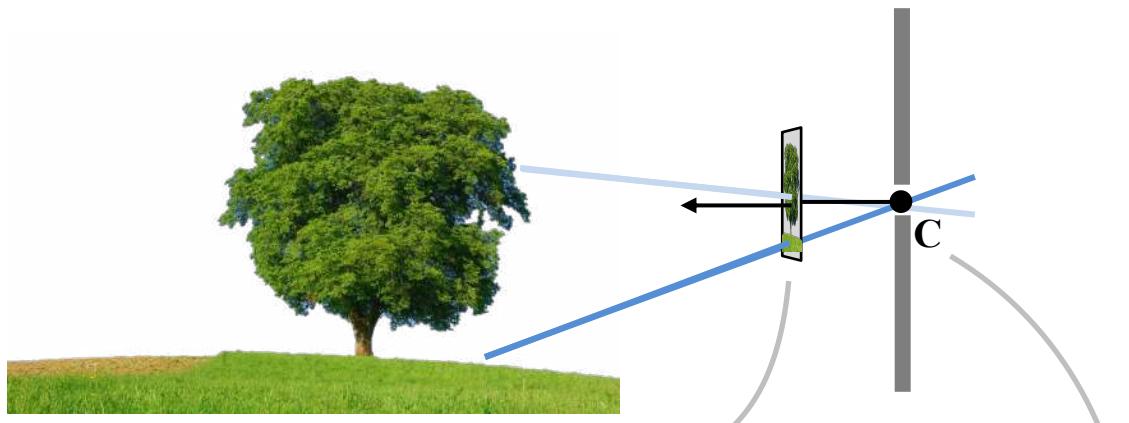
Film or sensor

Pinhole Camera Model

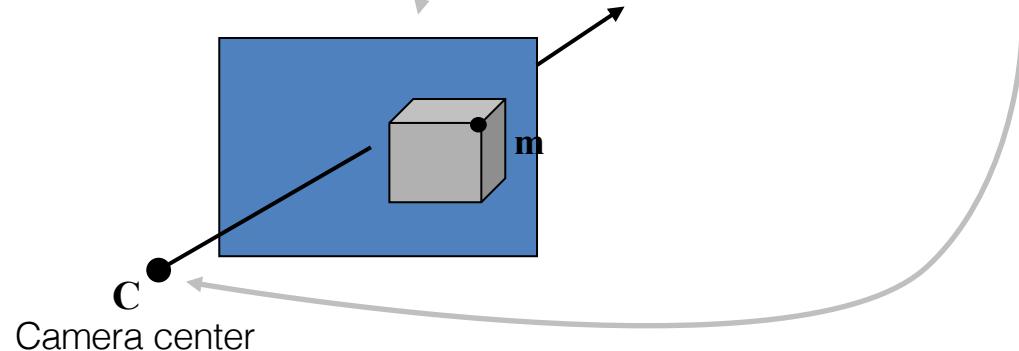


Camera center: Located at the aperture is, i.e. where the rays intersect.

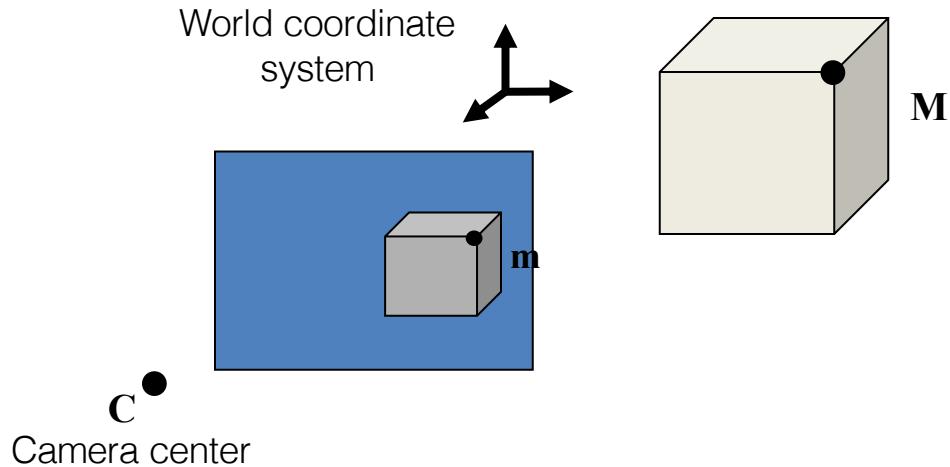
Image plane: does not correspond to a “physical” plane but the image is not reversed (it is reversed on the sensor or film).



On the next slides:



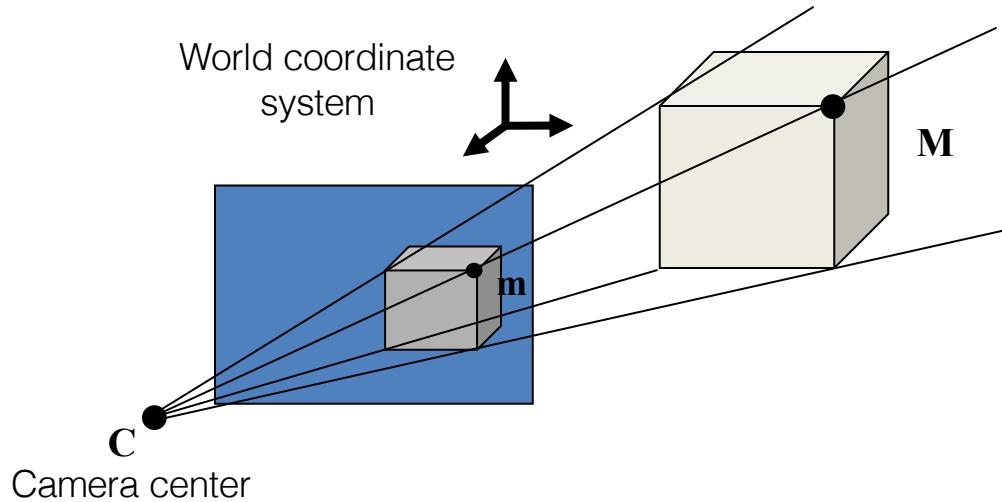
From the World to the Image



What is the relation between

- the 3D coordinates of a point **M** (expressed in a coordinate system related to the world) and
- the corresponding pixel **m** in the image captured by the camera ?

Perspective Projection

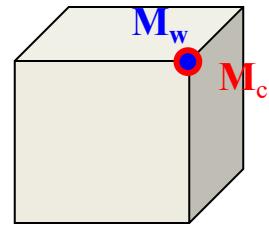
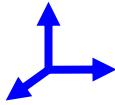


The image formation is modeled as a perspective projection, which is realistic for standard cameras:

All the rays passing through a 3D point \mathbf{M} and the corresponding pixel \mathbf{m} in the image intersect at a single point \mathbf{C} , the camera center.

Expressing \mathbf{M} in the Camera Coordinate System

World coordinate system



Step 1: Convert the coordinates of \mathbf{M}_w to the camera coordinate system as \mathbf{M}_c .

This transformation corresponds to a Euclidean displacement (a rotation plus a translation):

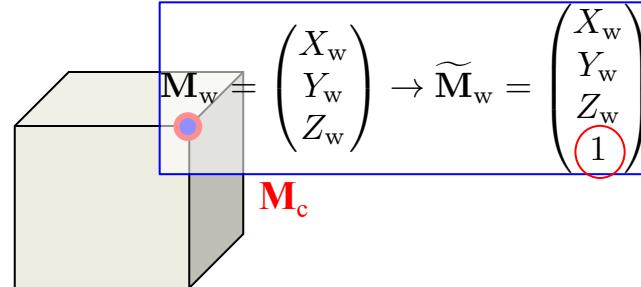
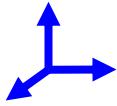
$$\mathbf{M}_c = \mathbf{R}\mathbf{M}_w + \mathbf{T}$$

where:

\mathbf{R} is a 3×3 rotation matrix, and \mathbf{T} is a 3-translation vector.

Homogeneous Coordinates

World coordinate system



Let's replace \mathbf{M}_w by the 4- homogeneous vector $\tilde{\mathbf{M}}_w$: This simply amounts adding a 1 as the fourth coordinate (also see next slides).

This will allow us to simplify the notations.

Homogeneous Coordinates

A mathematical trick to express non-linear transformations such as translation or projection as linear transformations.

A 3D point is expressed in homogeneous coordinates as a 4D vector:

$$\mathbf{M} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad \rightarrow \quad \tilde{\mathbf{M}} = \begin{pmatrix} kX \\ kY \\ kz \\ k \end{pmatrix} \text{ for any } k \in \mathbb{R}^*$$

→ A homogeneous vector is defined up to a scale factor:

$$\begin{pmatrix} kX \\ kY \\ kz \\ k \end{pmatrix} \equiv \begin{pmatrix} k'X \\ k'Y \\ k'Z \\ k' \end{pmatrix} \text{ for any } k, k' \in \mathbb{R}^*$$

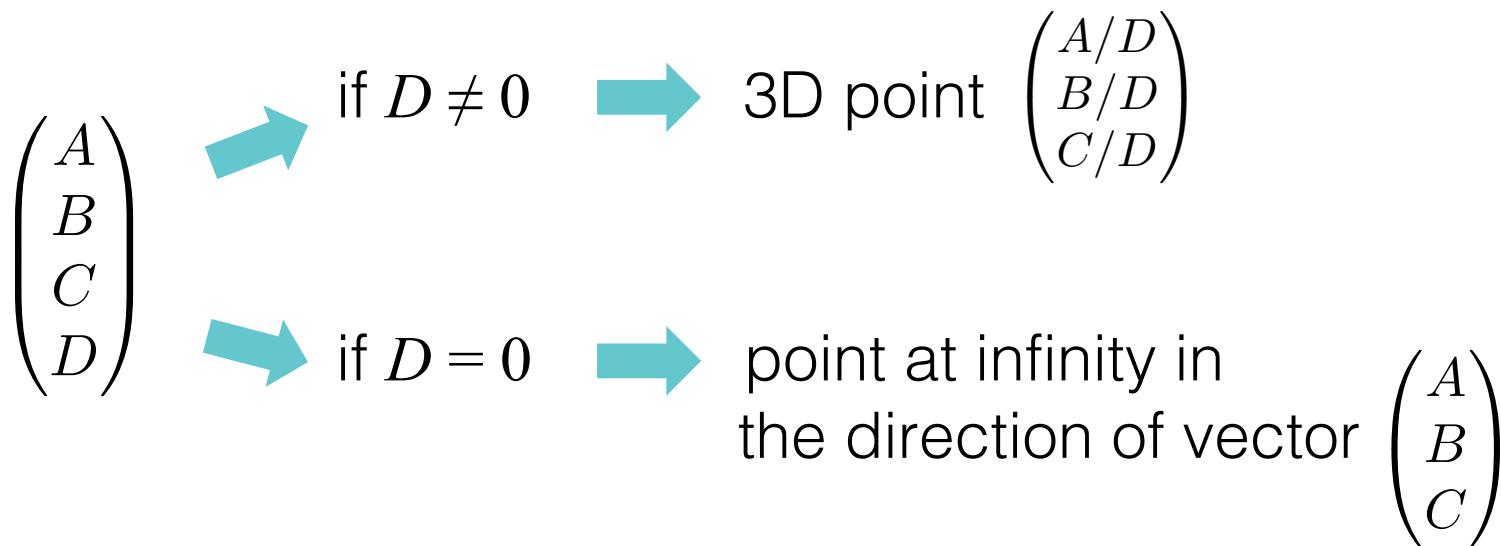
Homogeneous Coordinates

A homogeneous vector is defined up to a scale factor:

$$\begin{pmatrix} kX \\ kY \\ kZ \\ k \end{pmatrix} \equiv \begin{pmatrix} k'X \\ k'Y \\ k'Z \\ k' \end{pmatrix} \text{ for any } k, k' \in \mathbb{R}^*$$

If $\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \equiv \begin{pmatrix} a' \\ b' \\ c' \\ d' \end{pmatrix}$ and $d \neq 0, d' \neq 0$ then $\begin{pmatrix} \frac{a}{d} \\ \frac{b}{d} \\ \frac{c}{d} \end{pmatrix} = \begin{pmatrix} \frac{a'}{d'} \\ \frac{b'}{d'} \\ \frac{c'}{d'} \end{pmatrix}$

From Homogeneous Coordinates to Euclidean Coordinates



Using Homogeneous Coordinates (1)

Using homogeneous coordinates, we can write a Euclidean rigid motion as a linear transformation in the homogeneous space:

$$\mathbf{RM}_w + \mathbf{T} \rightarrow \underbrace{\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix}}_{4 \times 4 \text{ matrix}} \widetilde{\mathbf{M}}_w$$

proof

$$\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \widetilde{\mathbf{M}} \equiv \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} kX \\ kY \\ kZ \\ k \end{pmatrix} \equiv \begin{pmatrix} k\mathbf{R} \begin{pmatrix} X \\ Y \\ Z \\ k \end{pmatrix} + k\mathbf{T} \end{pmatrix}$$
$$\rightarrow \mathbf{RM} + \mathbf{T}$$

Applying the Motion to a Point at Infinity

The translation has no effect on points at infinity:

$$\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ 0 \end{pmatrix} \equiv \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix} \equiv \begin{pmatrix} \mathbf{R}\mathbf{v} \\ 0 \end{pmatrix} \rightarrow \mathbf{R}\mathbf{v} = \mathbf{R} \begin{pmatrix} A \\ B \\ C \end{pmatrix}$$

Using Homogeneous Coordinates (1)

Using homogeneous coordinates, we can write a Euclidean rigid motion as a linear transformation in the homogeneous space:

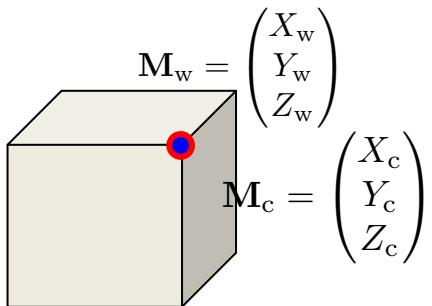
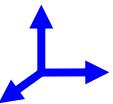
$$\mathbf{RM} + \mathbf{T} \rightarrow \underbrace{\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix}}_{4 \times 4 \text{ matrix}} \widetilde{\mathbf{M}}$$

In computer vision, the fourth coordinate (k) is usually set to 1, and a 3×4 matrix is used instead:

$$\mathbf{M}_c = \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \mathbf{RM}_w + \mathbf{T} = \mathbf{R} \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + \mathbf{T} = \underbrace{\begin{pmatrix} \mathbf{R} & \mathbf{T} \end{pmatrix}}_{3 \times 4 \text{ matrix}} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

World Coordinates to Camera Coordinates

World coordinate system



$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = (\mathbf{R} \quad \mathbf{T}) \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$
$$= \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

The External Calibration Matrix

The matrix

$$\begin{pmatrix} \mathbf{R} & \mathbf{T} \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{pmatrix}$$

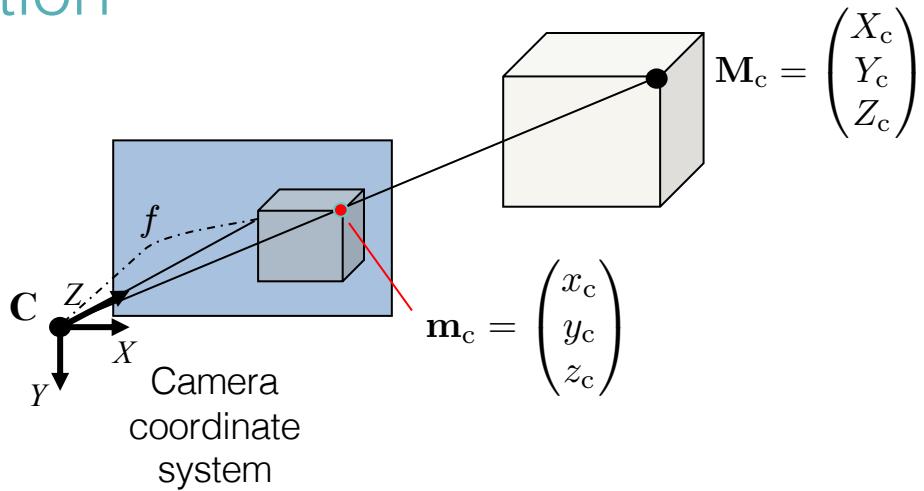
is called the "External calibration matrix", or the "extrinsic calibration matrix", or the "matrix of external parameters".

It can be parameterized by 6 values: 3 for the rotation, 3 for the translation.

The parameterization of the translation is trivial, while parametrizing rotations is not.

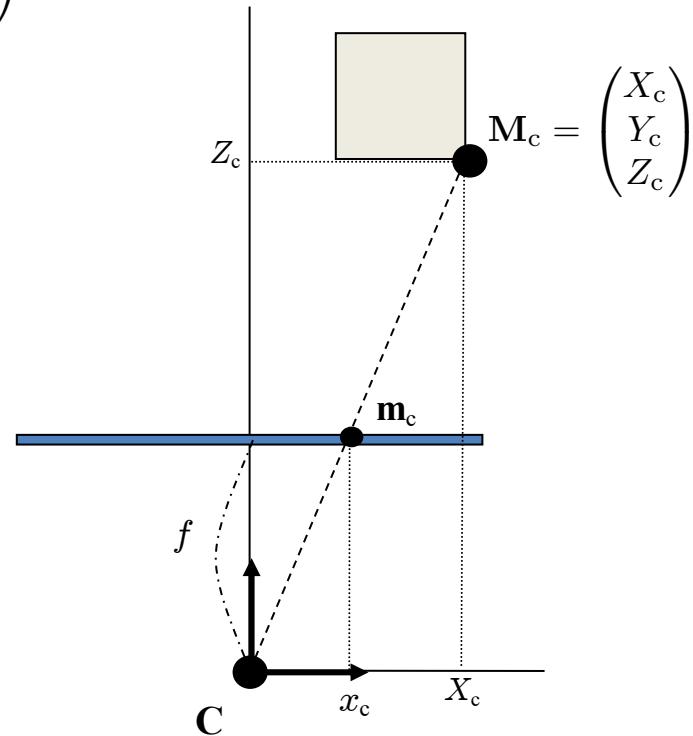
We will detail the possible parameterizations of rotations in the 3D space.

Projection



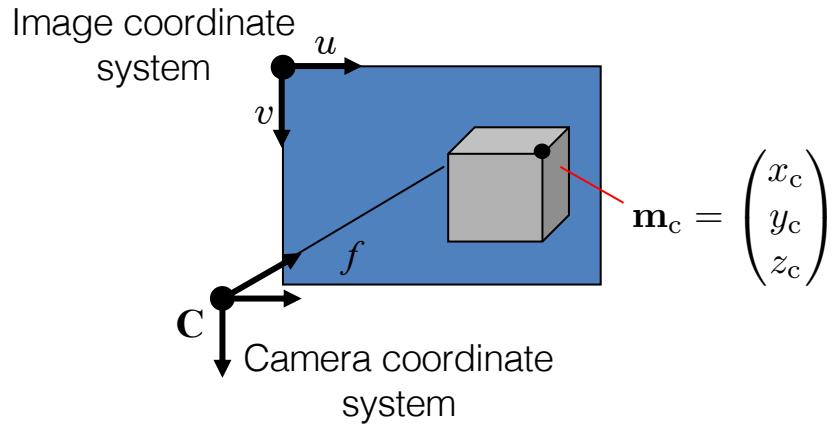
Coordinates of \mathbf{m}_c (a 3D point!) in the camera coordinate system: Simply use Thales' theorem

$$\frac{x_c}{f} = \frac{X_c}{Z_c} \rightarrow x_c = f \frac{X_c}{Z_c}$$



From Projection to Image

$$x_c = f \frac{X_c}{Z_c}, y_c = f \frac{Y_c}{Z_c}$$

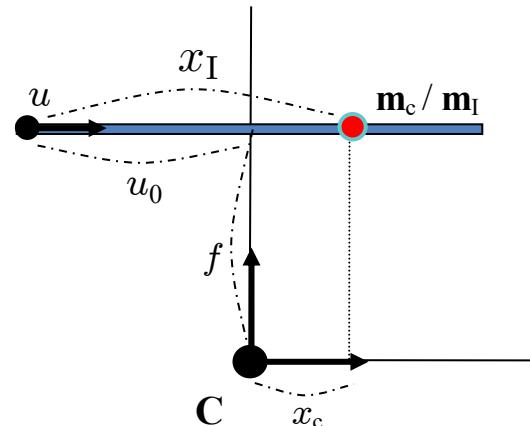
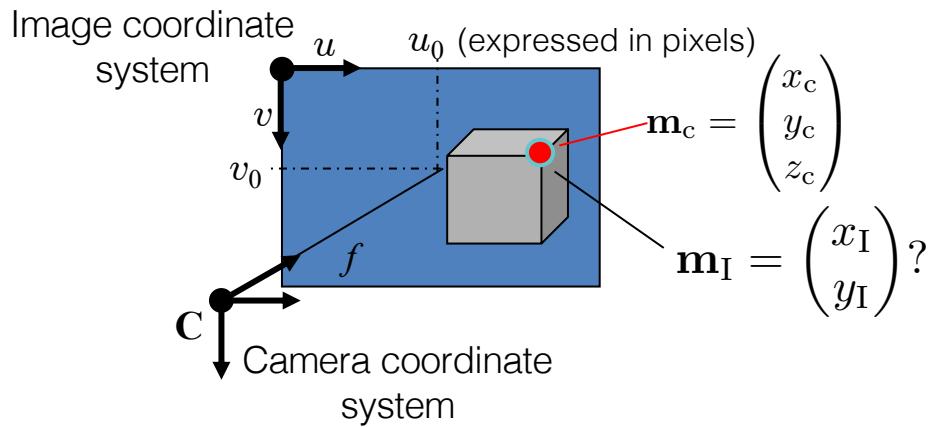


Coordinates of \mathbf{m} in pixels ?

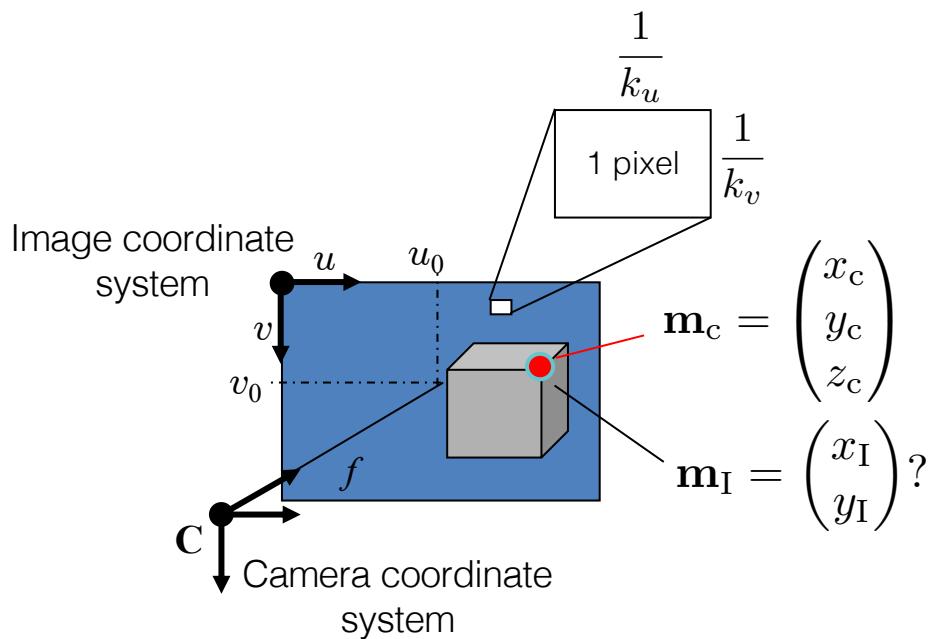
From Projection to Image

$$x_c = f \frac{X_c}{Z_c}, \quad y_c = f \frac{Y_c}{Z_c}$$

Coordinates of \mathbf{m}_I in pixels ?

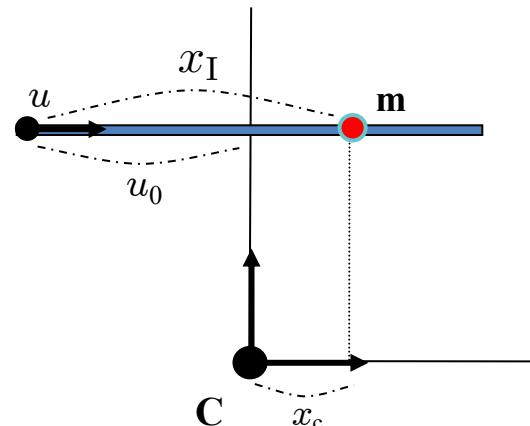


From Projection to Image

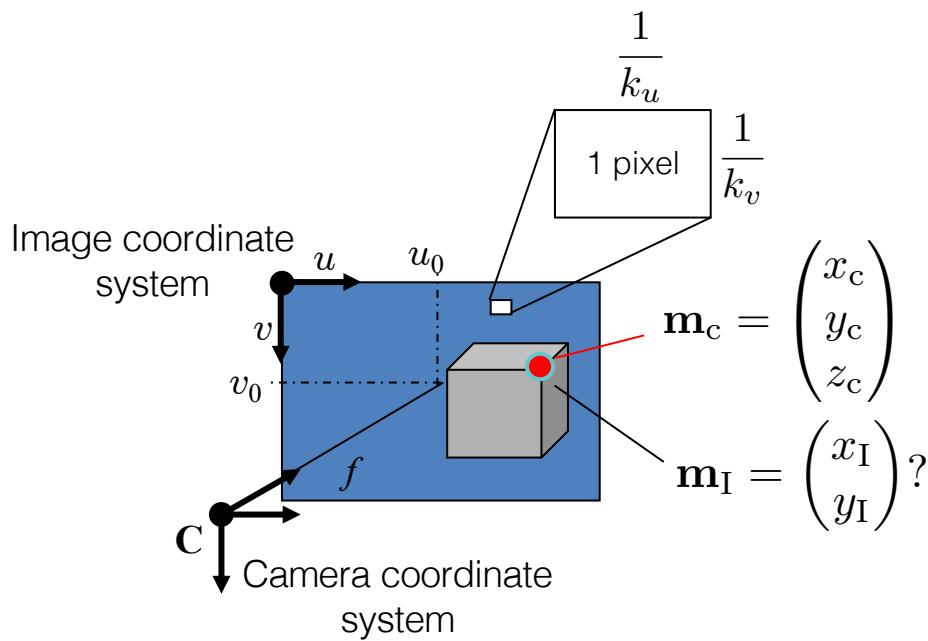


$\frac{1}{k_u}$ size of a pixel along the X axis

x_I ?

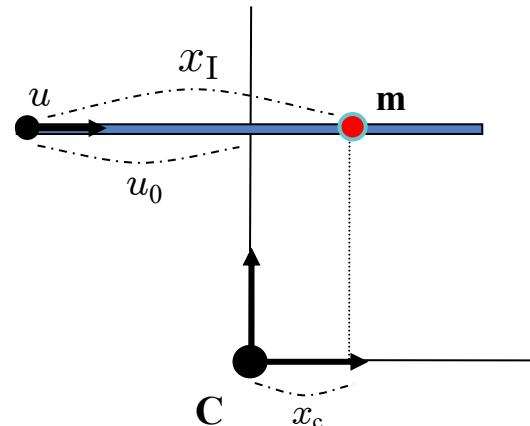


From Projection to Image

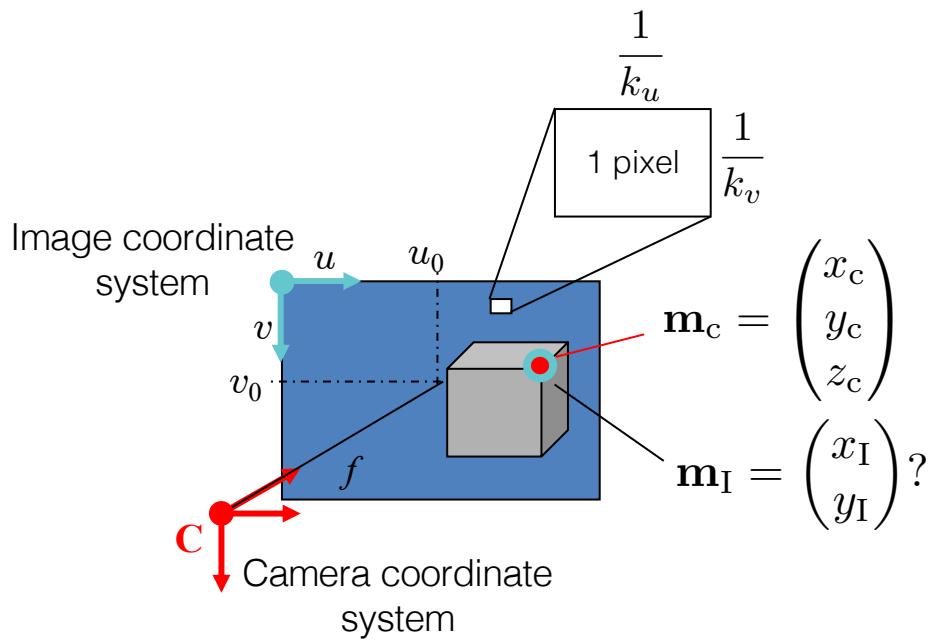


$\frac{1}{k_u}$ size of a pixel along the X axis

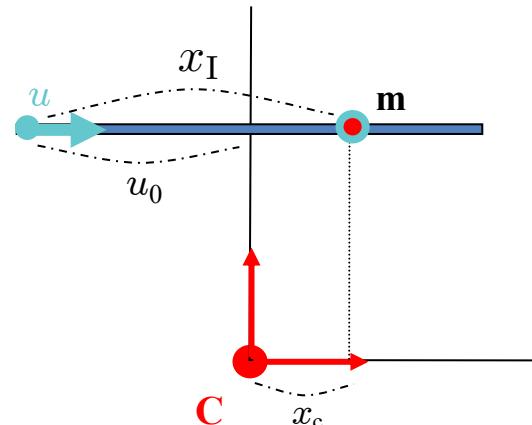
$$x_I = u_0 + k_u x_c$$



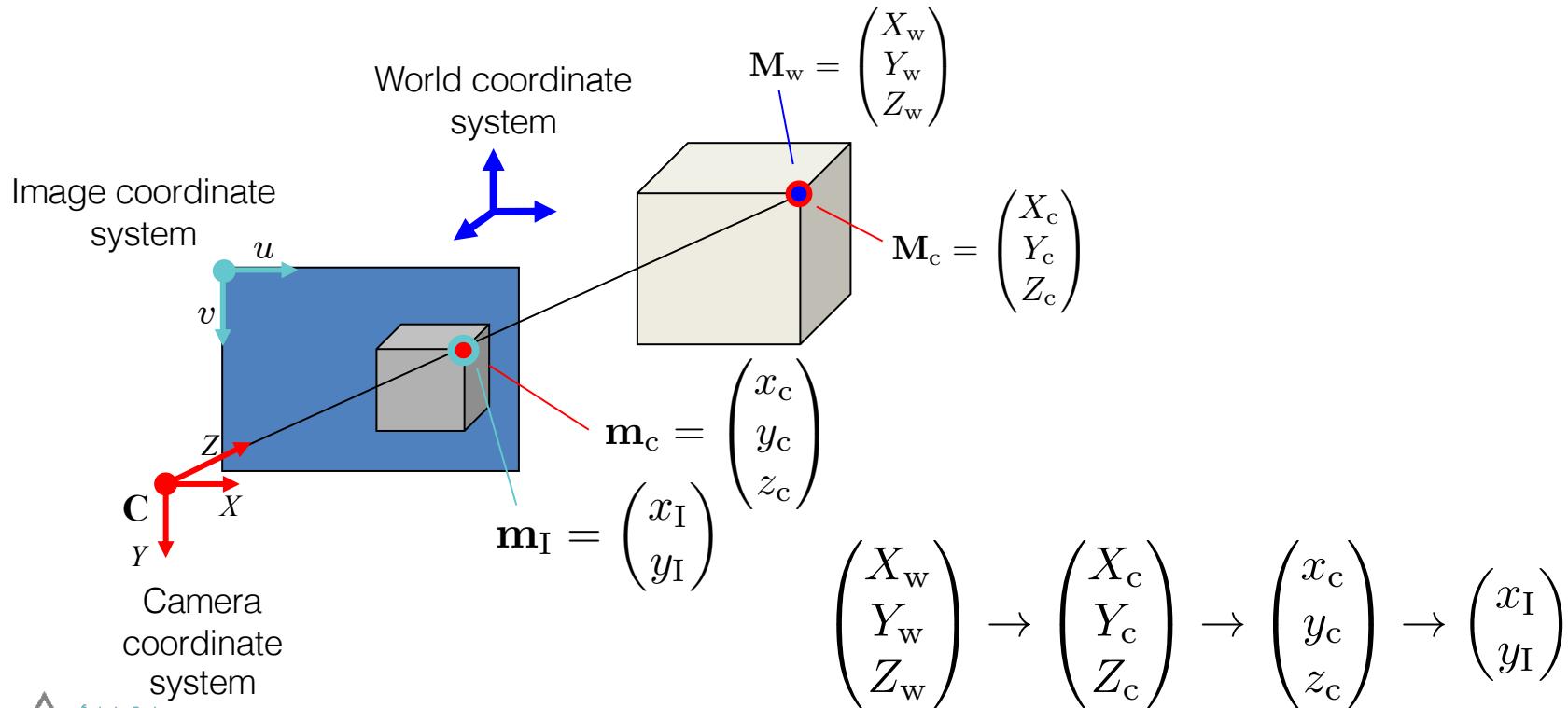
From Projection to Image



$$\left\{ \begin{array}{l} x_I = u_0 + k_u x_c \\ y_I = v_0 + k_v y_c \end{array} \right.$$



Overview



Using Homogeneous Coordinates (2)

$$x_c = f \frac{X_c}{Z_c}, \quad y_c = f \frac{Y_c}{Z_c} \quad \begin{cases} x_I = u_0 + k_u x_c \\ y_I = v_0 + k_v y_c \end{cases}$$

$$\Rightarrow \begin{cases} x_I = u_0 + k_u f \frac{X_c}{Z_c} \\ y_I = v_0 + k_v f \frac{Y_c}{Z_c} \end{cases}$$

The transformation from $(X_c, Y_c, Z_c)^T$ to $(x_I, y_I)^T$ can be written in matrix form using homogeneous coordinates:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

Using Homogeneous Coordinates (2)

$$\begin{cases} x_I = u_0 + k_u f \frac{X_c}{Z_c} \\ y_I = v_0 + k_v f \frac{Y_c}{Z_c} \end{cases}$$

The transformation from $(X_c, Y_c, Z_c)^T$ to $(x_I, y_I)^T$ can be written in matrix form using homogeneous coordinates:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} k_u f X_c + u_0 Z_c \\ k_v f Y_c + v_0 Z_c \\ Z_c \end{pmatrix}$$

$$\rightarrow \begin{cases} x_I = \frac{u}{w} = \frac{k_u f X_c + u_0 Z_c}{Z_c} = u_0 + k_u f \frac{X_c}{Z_c} \\ y_I = \frac{v}{w} = \frac{k_v f Y_c + v_0 Z_c}{Z_c} = v_0 + k_v f \frac{Y_c}{Z_c} \end{cases}$$

The Internal Calibration Matrix

The 3×3 matrix

$$\begin{pmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

is called the "Internal calibration matrix", or the "intrinsic calibration matrix", or the "calibration matrix", or the "matrix of internal parameters".

It is not possible to estimate k_u , k_v , and f separately, only their products, and the intrinsic matrix can be parameterized by 4 values: α_u , α_v , u_0 , v_0 :

$$\begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

The number of parameters are often further reduced, under some assumptions:

- (u_0, v_0) is often taken at the center of the image;
- Taking $\alpha_u = \alpha_v$ assumes that the pixels are squared.



Putting Things Together: The Projection Matrix

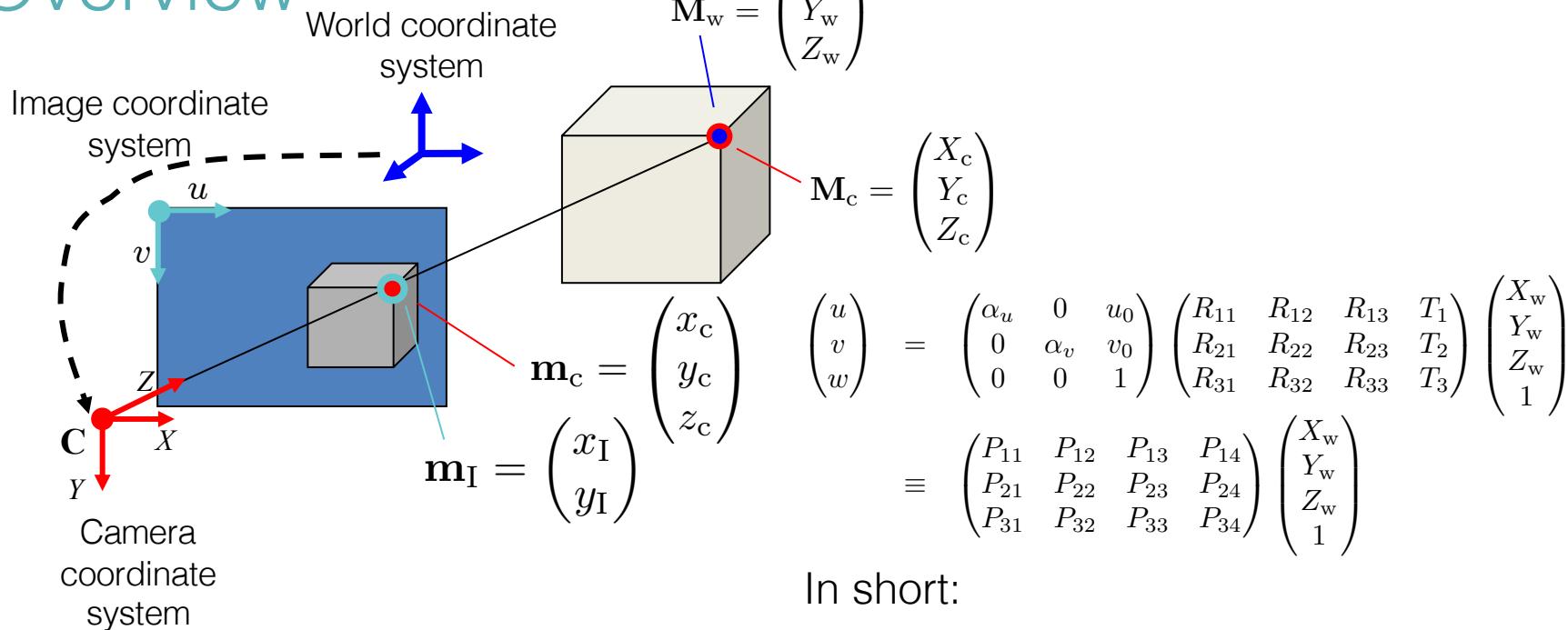
The two previous transformations can be chained to form the full transformation from a 3D point in the world coordinate system to its projection in the image:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$
$$\equiv \underbrace{\begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix}}_{3 \times 4 \text{ projection matrix}} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

The product of the internal calibration matrix and the external calibration matrix is a 3x4 matrix called the "projection matrix".

More exactly, any matrix proportional to this product is equivalent. It is defined up to a scale factor.

Overview

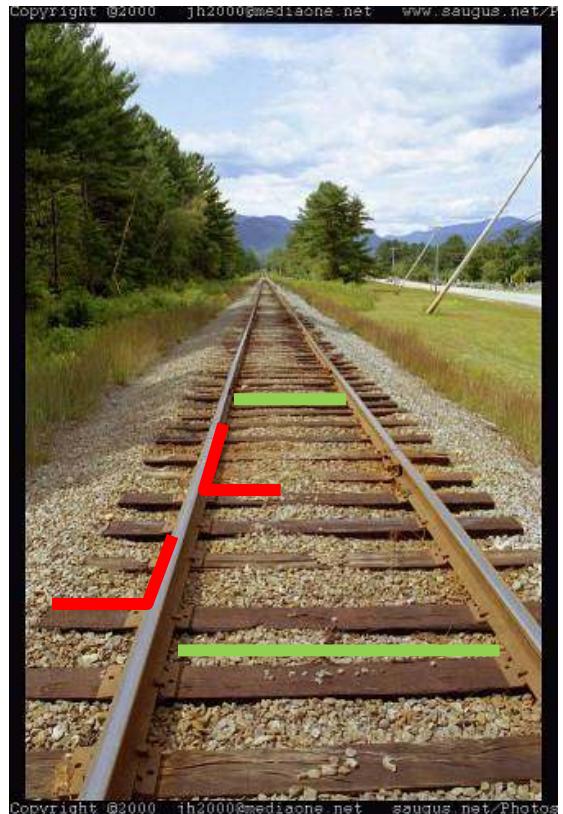


In short:

$$\begin{aligned}\tilde{\mathbf{m}} &\equiv \mathbf{K}(\mathbf{R} \ \mathbf{T})\tilde{\mathbf{M}}_w \\ &\equiv \widetilde{\mathbf{P}}\tilde{\mathbf{M}}_w\end{aligned}$$

Perspective Geometry

Lengths, angles are not preserved by perspective projection.



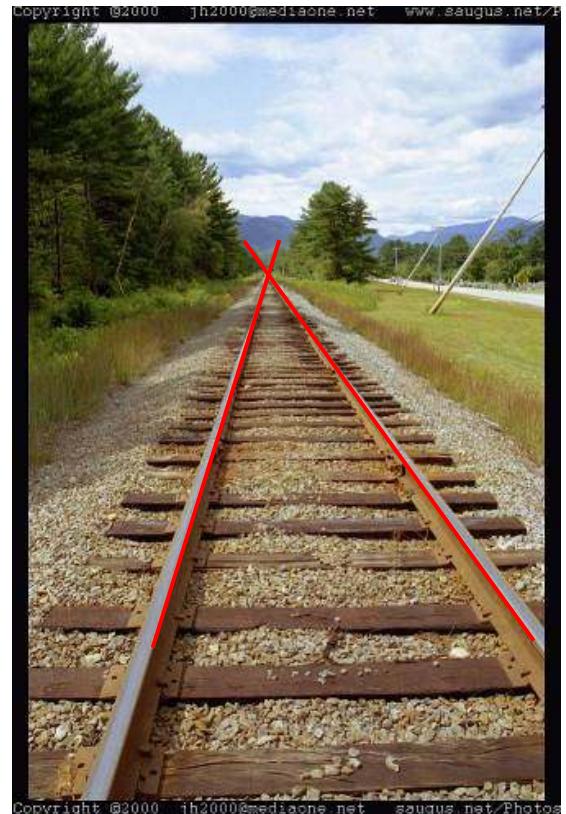
Copyright ©2000 jh2000@mediacne.net www.saugus.net/Photos

Perspective Geometry

Lengths, angles are not preserved by perspective projection.

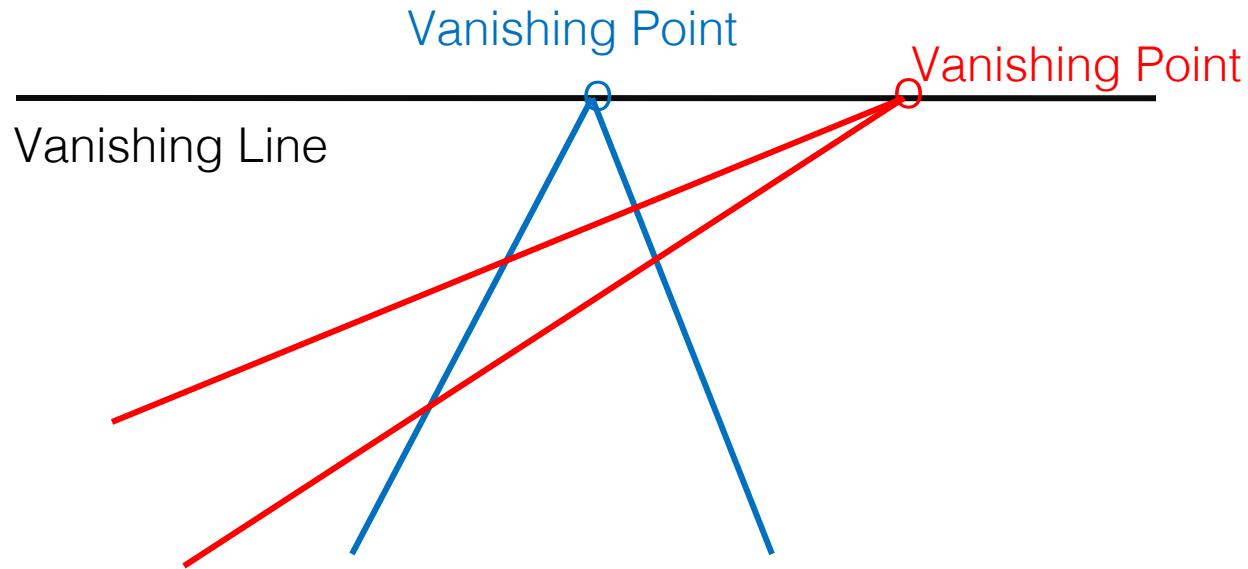
Straight lines remain straight.

Parallel lines in the world intersect in the image at a “vanishing point”

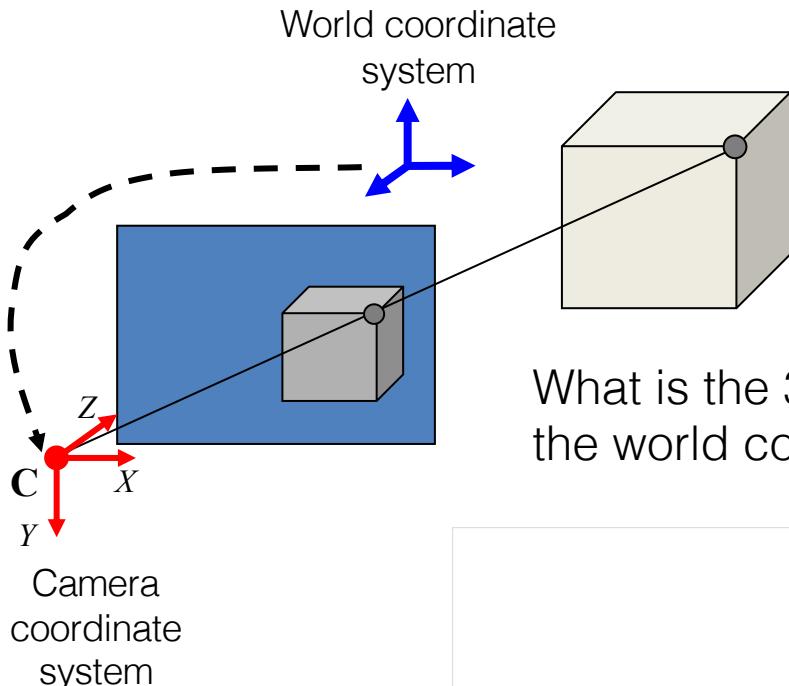


Copyright ©2000 jh2000@mediacne.net www.saugus.net/Photos

Vanishing Points and Lines



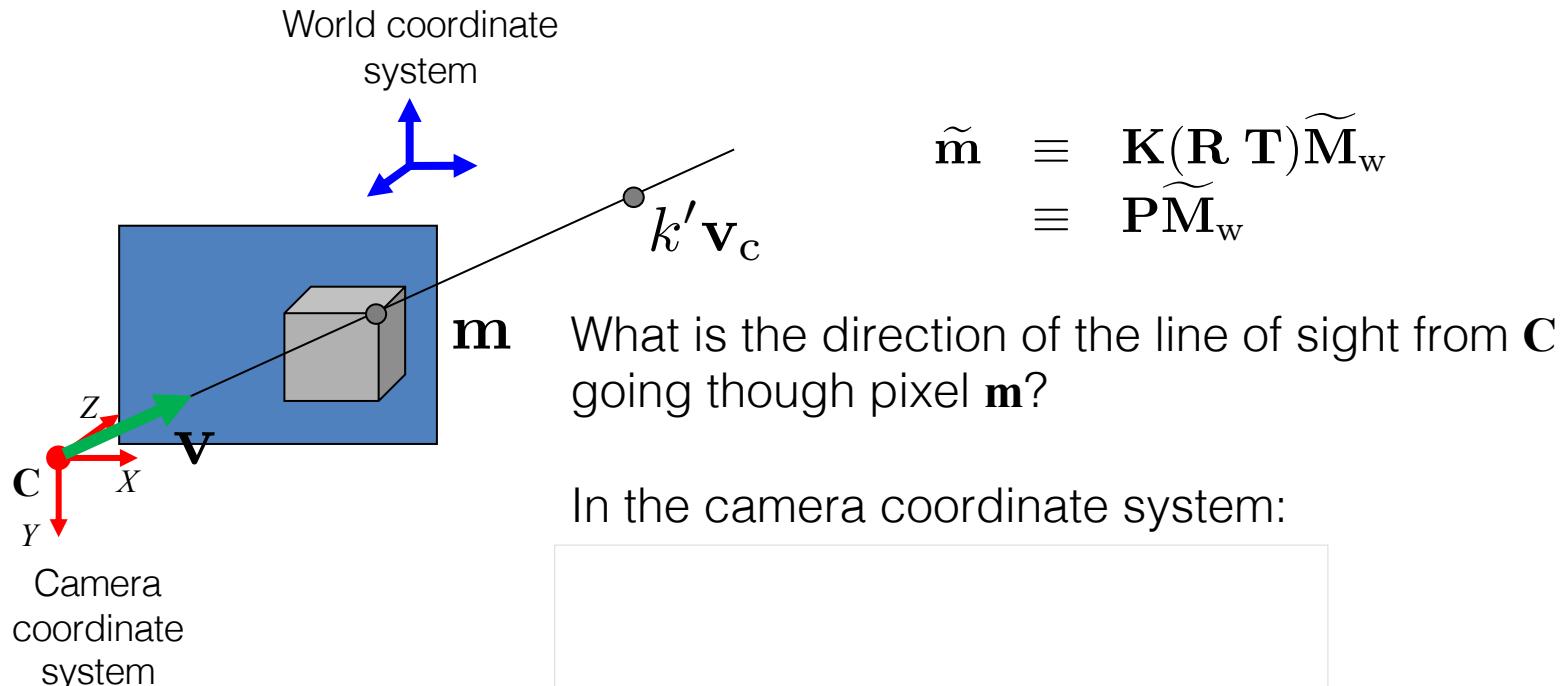
Some Relations



$$\begin{aligned}\tilde{\mathbf{m}} &\equiv \mathbf{K}(\mathbf{R} \ \mathbf{T})\widetilde{\mathbf{M}}_w \\ &\equiv \widetilde{\mathbf{P}}\widetilde{\mathbf{M}}_w\end{aligned}$$

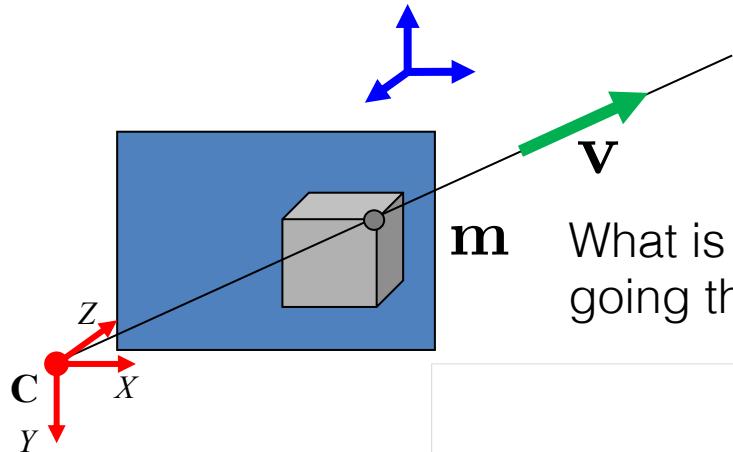
What is the 3D coordinates of **C**, the camera center, in the world coordinate system?

Some Relations



Some Relations

World coordinate
system



Camera
coordinate
system

$$\begin{aligned}\tilde{\mathbf{m}} &\equiv \mathbf{K}(\mathbf{R} \ \mathbf{T})\widetilde{\mathbf{M}}_w \\ &\equiv \mathbf{P}\widetilde{\mathbf{M}}_w\end{aligned}$$

m What is the direction of the line of sight from **C** going through pixel **m**?

Estimating a Transformation Between Points

- An affine transformation between 2D points;
- A homography between 2D points;
- A projection from 3D points to 2D points.

Estimating an affine transformation between pairs of 2D points

Problem:

we have given a set \mathcal{M} of pairs of 2D points:

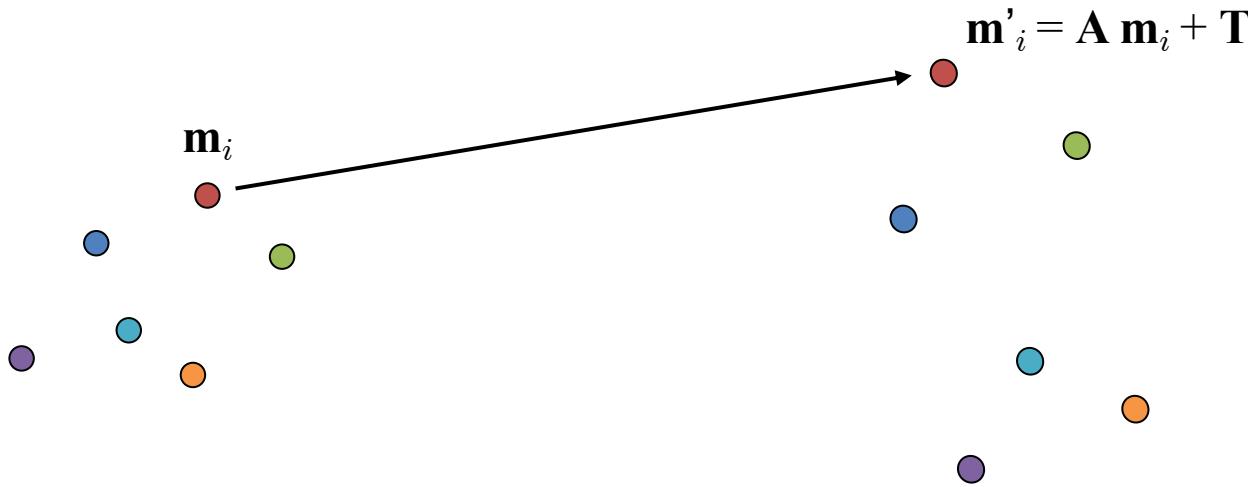
$$\mathcal{M} = \{(\mathbf{m}_i, \mathbf{m}'_i) \mid i \in [0 : n]\}$$

We assume there is an affine transformation that, for each pair, transforms the first point into the second one:

$$\exists \mathbf{A}, \mathbf{T} \mid \forall i \quad \mathbf{m}'_i = \mathbf{A}\mathbf{m}_i + \mathbf{T}$$

How can we find the coefficients of \mathbf{A} (a 2x2 matrix) and \mathbf{T} (a 2-vector) ?

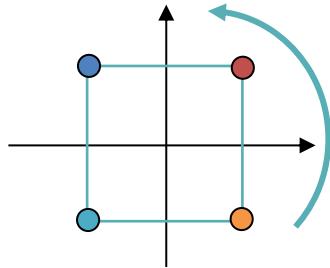
Affine transformation applied to points



Affine transformation applied to points

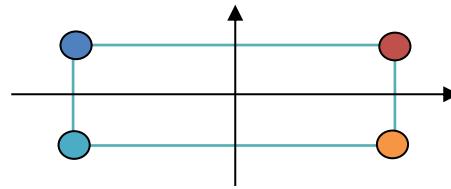
rotation

$$\mathbf{A} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}, \mathbf{T} = \mathbf{0}$$



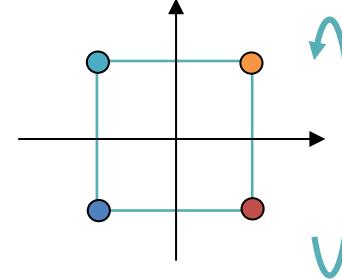
scale

$$\mathbf{A} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}, \mathbf{T} = \mathbf{0}$$



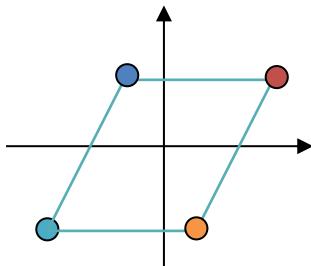
symmetry

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \mathbf{T} = \mathbf{0}$$



skew/shear

$$\mathbf{A} = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \mathbf{T} = \mathbf{0}$$



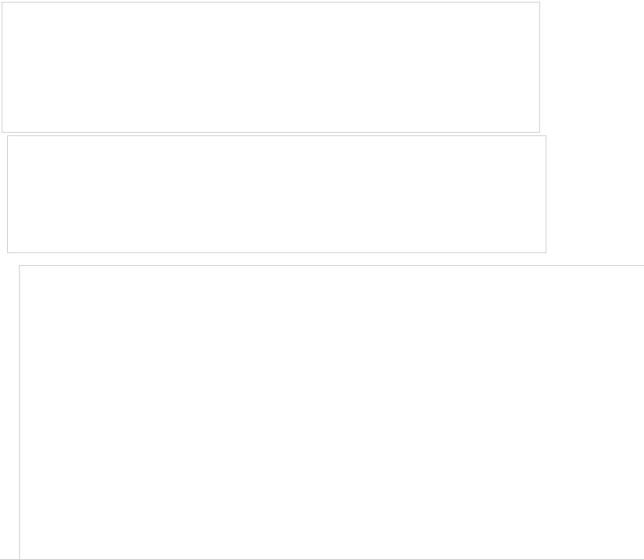
Any 2x2 matrix can be decomposed as:

$$\mathbf{A} = \mathbf{R}(\theta)\mathbf{R}(-\phi) \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix} \mathbf{R}(\phi)$$

(why? hint: SVD)

What does a pair of points give us?

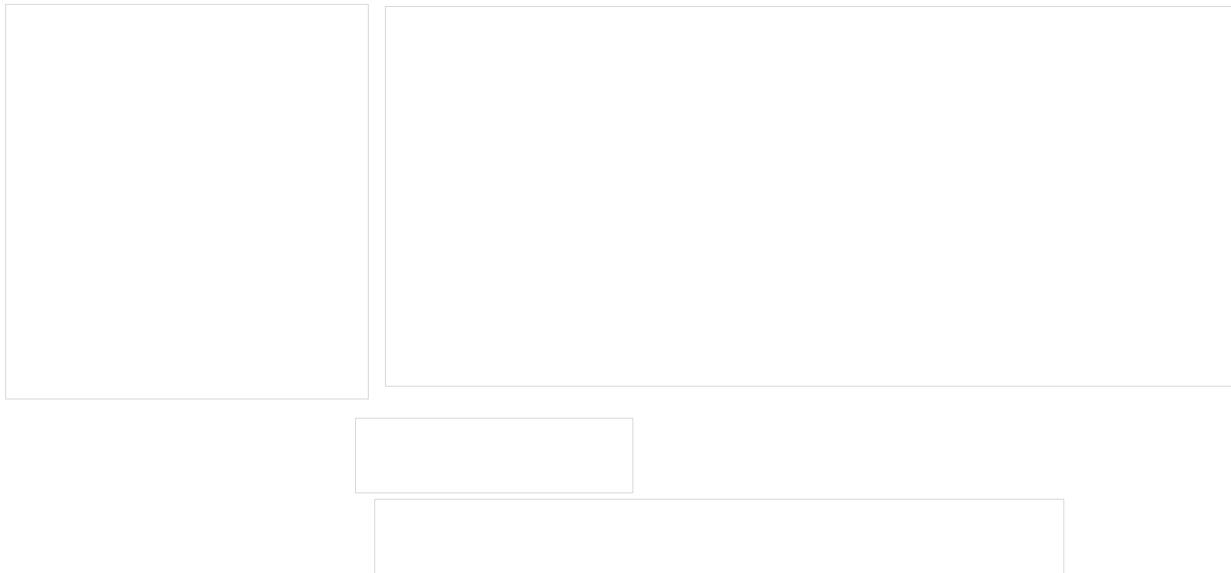
$$\mathbf{m} = \begin{pmatrix} x \\ y \end{pmatrix}, \mathbf{m}' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \mathbf{A} = \begin{pmatrix} a & b \\ d & e \end{pmatrix}, \mathbf{T} = \begin{pmatrix} c \\ f \end{pmatrix}$$



A linear system of 2 equations in unknowns (a, \dots, f)

Minimal problem: $|\mathcal{M}| = 3$

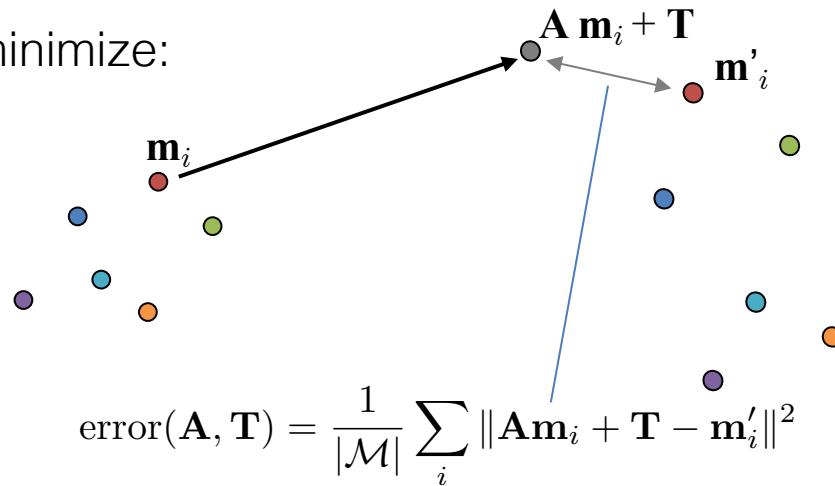
With 3 pairs of points:



What if $|\mathcal{M}| > 3$?

In practice, when $|\mathcal{M}| > 3$, there is no guarantee that there are an \mathbf{A} and a \mathbf{T} such that we have *exactly* $\mathbf{m}'_i = \mathbf{A} \mathbf{m}_i + \mathbf{T}$ for all i (for example, because the points are inaccurately detected).

In that case, we minimize:



We are looking for $\arg \min_{\mathbf{A}, \mathbf{T}} \text{error}(\mathbf{A}, \mathbf{T})$

When $|\mathcal{M}| > 3$

\mathbf{M} : a $(2N) \times 6$ matrix
→ overdetermined
problem when $N > 3$

$$\left\{ \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ \vdots & & & & & \\ x_N & y_N & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_N & y_N & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} x'_1 \\ y'_1 \\ \vdots \\ x'_N \\ y'_N \end{pmatrix} \right.$$

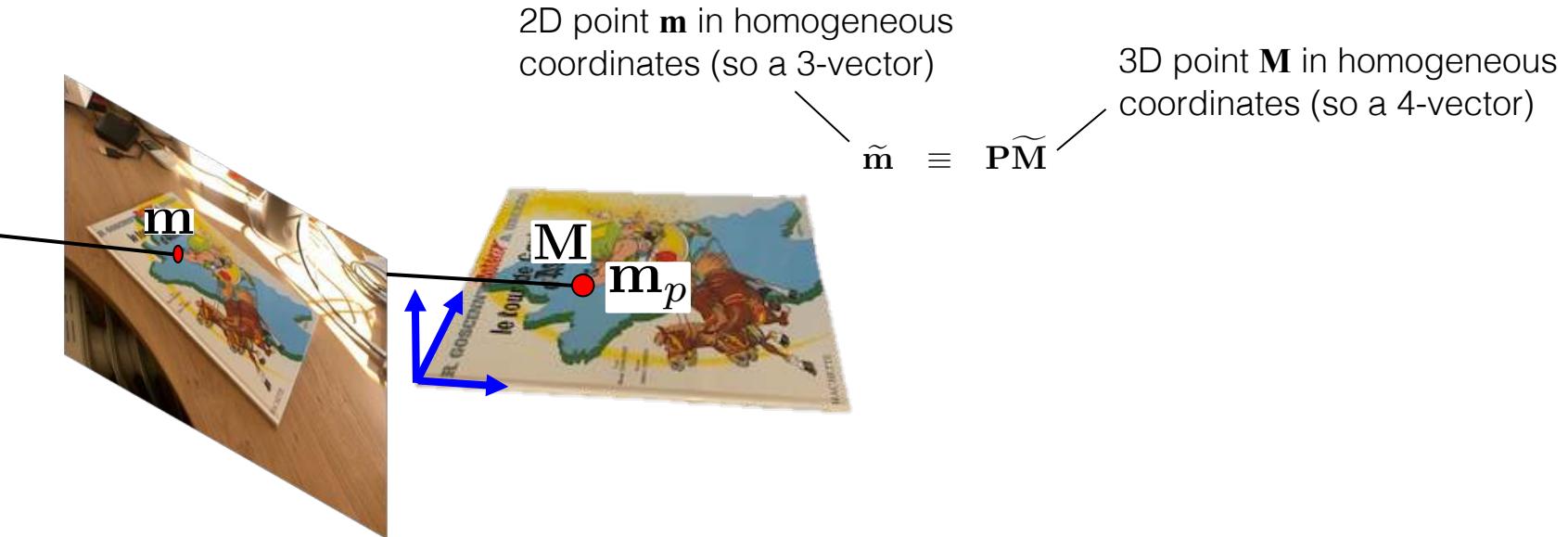
Consider $\mathbf{x} = \mathbf{M}^+ \mathbf{b}$, with \mathbf{M}^+ the pseudo-inverse of \mathbf{M} : $\mathbf{M}^+ = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T$

This gives the solution of our optimization problem: [why?]

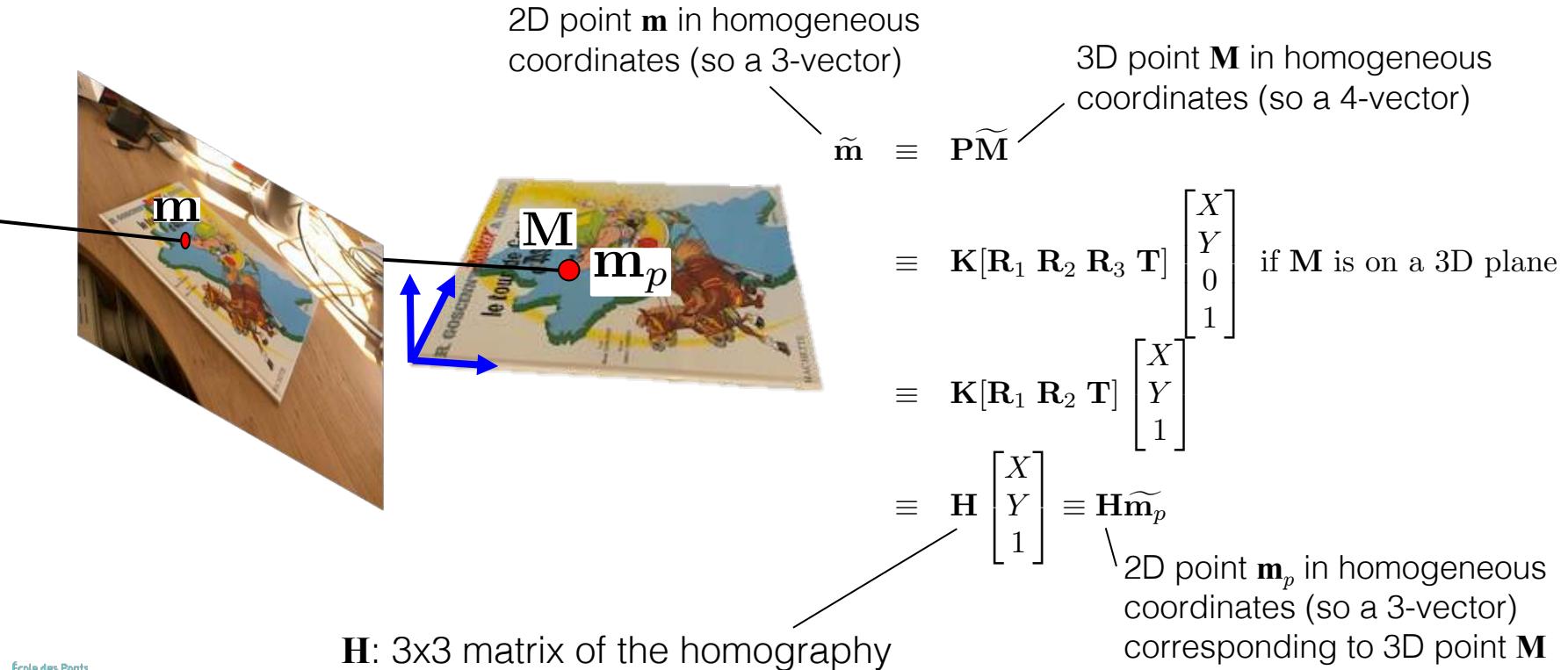
$$\arg \min_{\mathbf{A}, \mathbf{T}} \frac{1}{|\mathcal{M}|} \sum_i \|\mathbf{A}\mathbf{m}_i + \mathbf{T} - \mathbf{m}'_i\|^2$$

Estimating a Homography from Correspondences

The transformation between a 3D plane and its projection is a homography

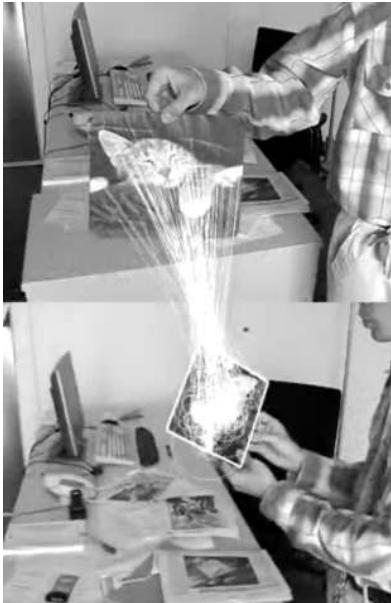


The transformation between a 3D plane and its projection is a homography

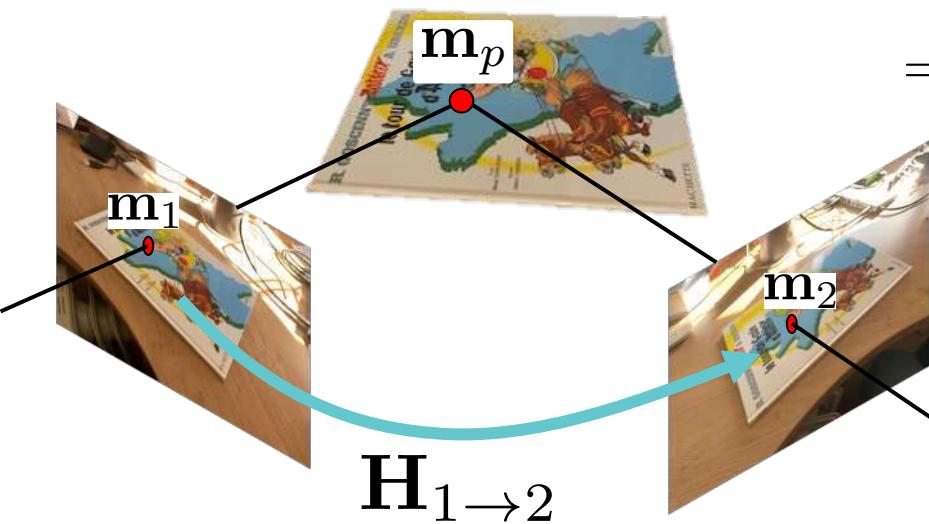


Homography: Transformation between a 3D plane and its projection

Example

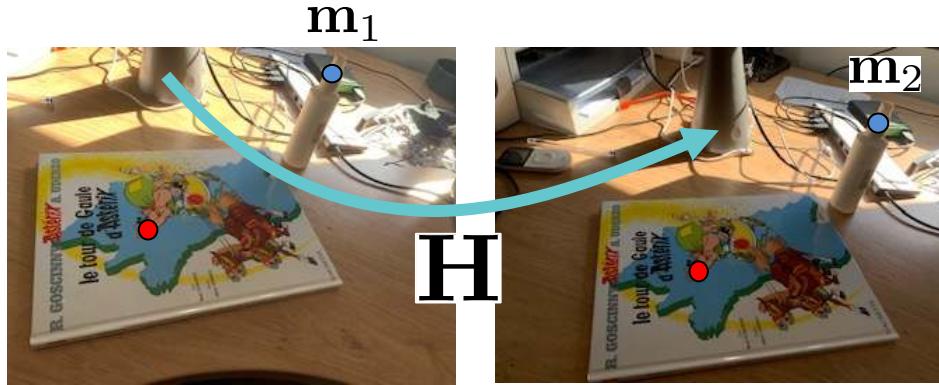


The transformation between 2 projections of a 3D plane is also a homography



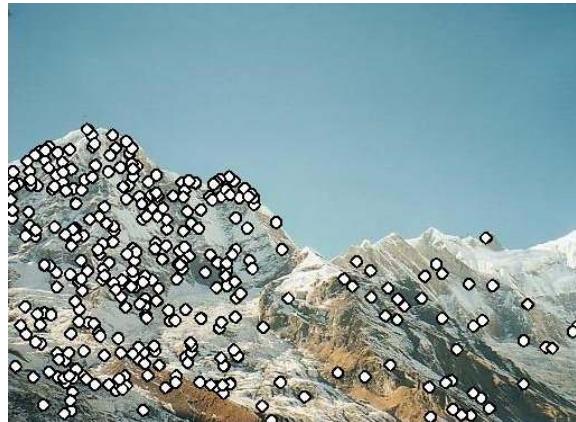
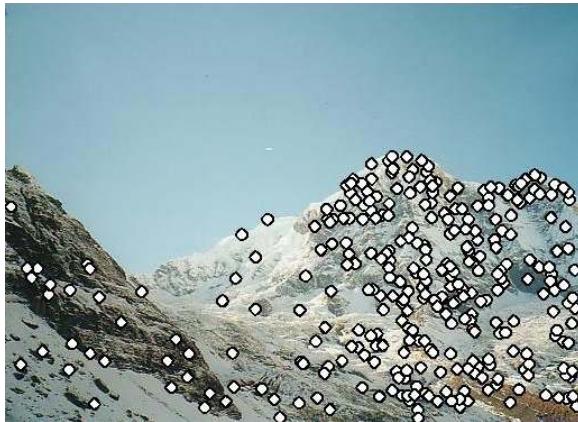
$$\begin{aligned}\tilde{m}_1 &\equiv H_1 \tilde{m}_p \text{ and } \tilde{m}_2 \equiv H_2 \tilde{m}_p \\ \Rightarrow \tilde{m}_2 &\equiv H_2 H_1^{-1} \tilde{m}_1 \equiv \underbrace{H_{1 \rightarrow 2}}_{\substack{\text{3x3 matrix}}} \tilde{m}_1\end{aligned}$$

The transformation between 2 projections of a 3D scene under rotation only is a homography as well



$$\tilde{m}_2 \equiv H\tilde{m}_1 \quad [\text{why?}]$$

Application: Constructing a Panorama



Homography in general

Transformation between 2D point \mathbf{m} and 2D point \mathbf{m}' of the form:

$$\widetilde{\mathbf{m}'} \equiv \mathbf{H}\widetilde{\mathbf{m}}.$$

With $\mathbf{m} = \begin{pmatrix} x \\ y \end{pmatrix}$, $\mathbf{m}' = \begin{pmatrix} x' \\ y' \end{pmatrix}$, and $\mathbf{H} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$, $\forall k, k' \neq 0$:

$$\rightarrow \begin{pmatrix} k'x' \\ k'y' \\ k' \end{pmatrix} \equiv \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} kx \\ ky \\ k \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} k'x' \\ k'y' \\ k' \end{pmatrix} \equiv \begin{pmatrix} akx + bky + ck \\ dkx + eky + fk \\ gkx + hky + ik \end{pmatrix} \rightarrow \begin{pmatrix} \frac{k'x'}{k'} \\ \frac{k'y'}{k'} \end{pmatrix} = \begin{pmatrix} \frac{akx + bky + ck}{gkx + hky + ik} \\ \frac{dkx + eky + fk}{gkx + hky + ik} \end{pmatrix} \rightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \frac{ax + by + c}{gx + hy + i} \\ \frac{dx + ey + f}{gx + hy + i} \end{pmatrix}$$

Homography in general

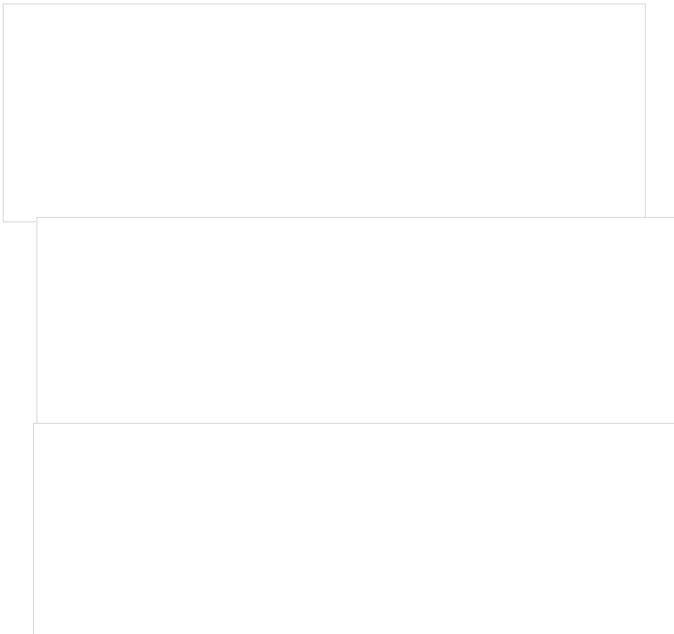
Transformation between 2D point \mathbf{m} and 2D point \mathbf{m}' of the form:
 $\widetilde{\mathbf{m}'} \equiv \mathbf{H}\widetilde{\mathbf{m}}$.

With $\mathbf{m} = \begin{pmatrix} x \\ y \end{pmatrix}$, $\mathbf{m}' = \begin{pmatrix} x' \\ y' \end{pmatrix}$, and $\mathbf{H} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$:

$$\begin{cases} x' = \frac{ax+by+c}{gx+hy+i} \\ y' = \frac{ax+by+c}{gx+hy+i} \end{cases}$$

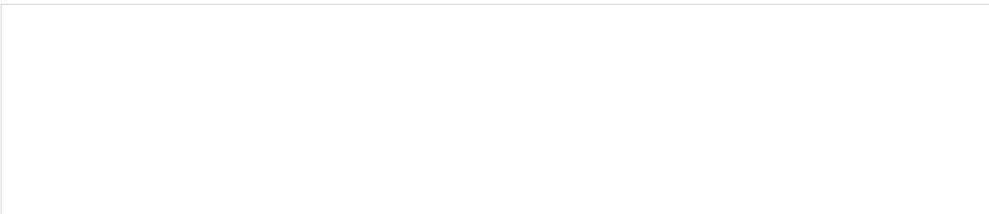
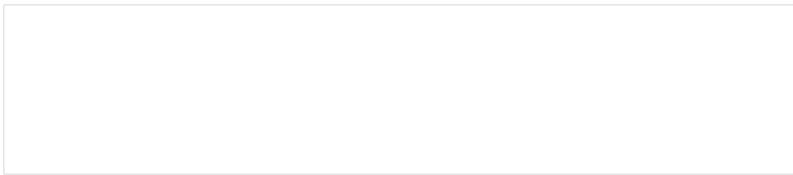
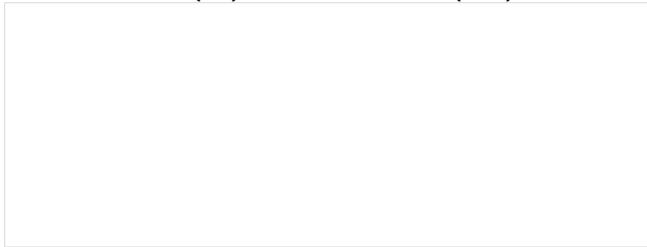
What does a pair of points give us?

$$\mathbf{m} = \begin{pmatrix} x \\ y \end{pmatrix}, \mathbf{m}' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \mathbf{H} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$



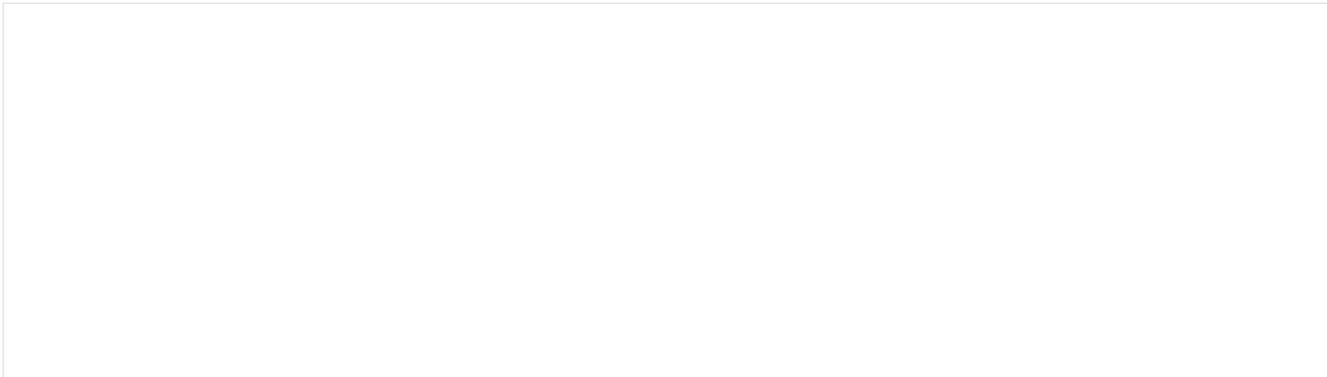
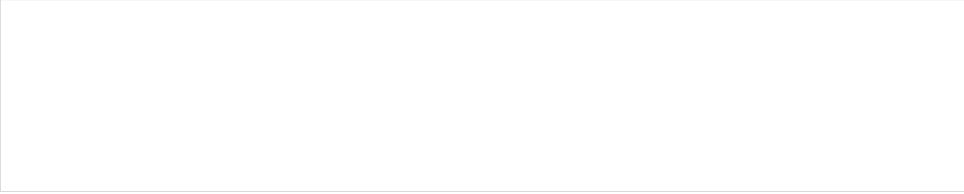
What does a pair of points give us?

$$\mathbf{m} = \begin{pmatrix} x \\ y \end{pmatrix}, \mathbf{m}' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \mathbf{H} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$



What does a pair of points give us?

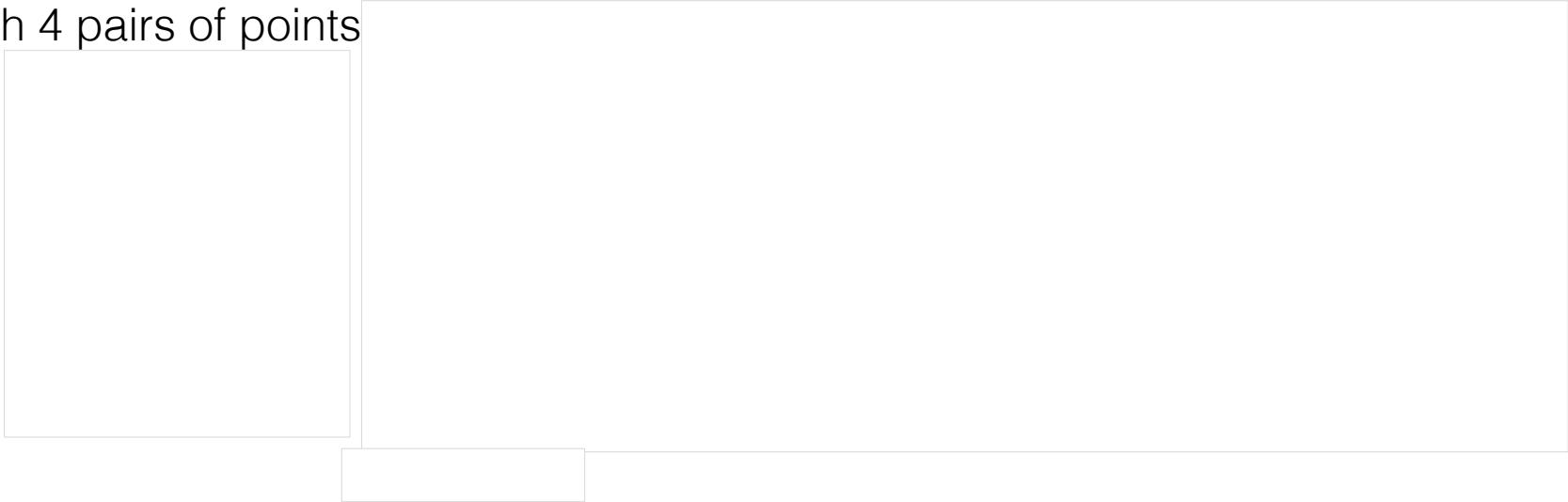
$$\mathbf{m} = \begin{pmatrix} x \\ y \end{pmatrix}, \mathbf{m}' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \mathbf{H} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$



A linear system of 2 equations in unknowns (a, \dots, i)

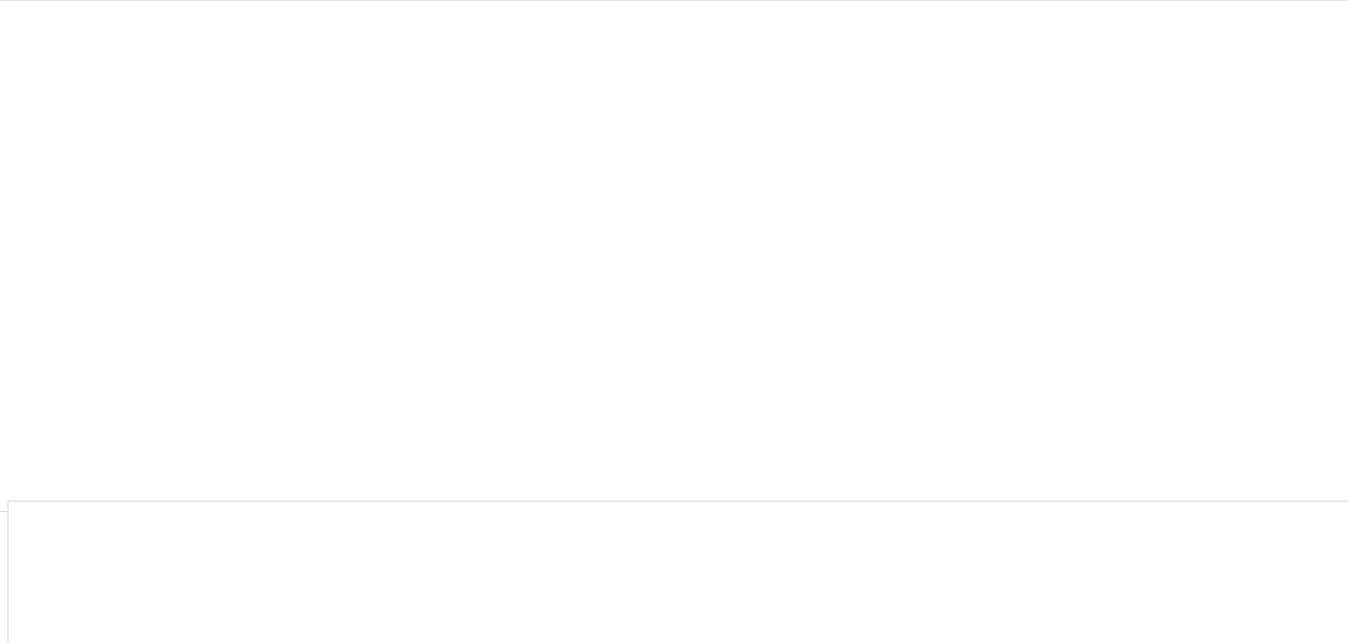
Minimal problem: $|\mathcal{M}| = 4$

With 4 pairs of points



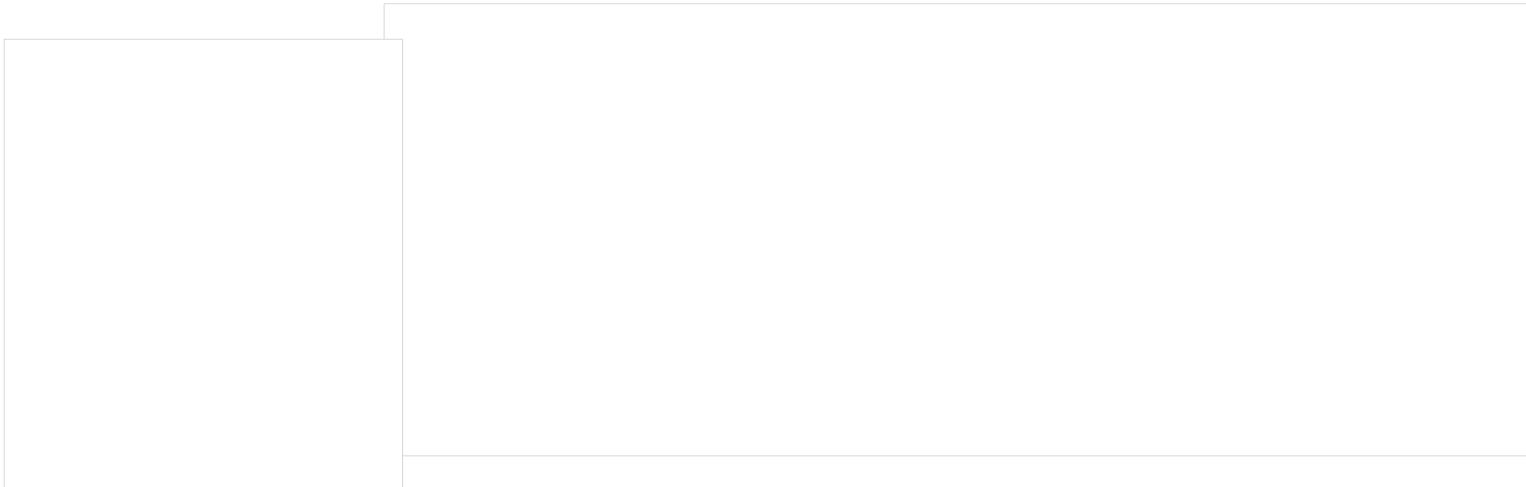
[why is **0** not a solution to our problem?]

Minimal problem: Solution #1



Minimal problem: Solution #1

\mathbf{H} is defined up to a scale factor and we can choose $i = 1$



A linear system of 8 equations in 8 unknowns

Minimal problem: Solution #2

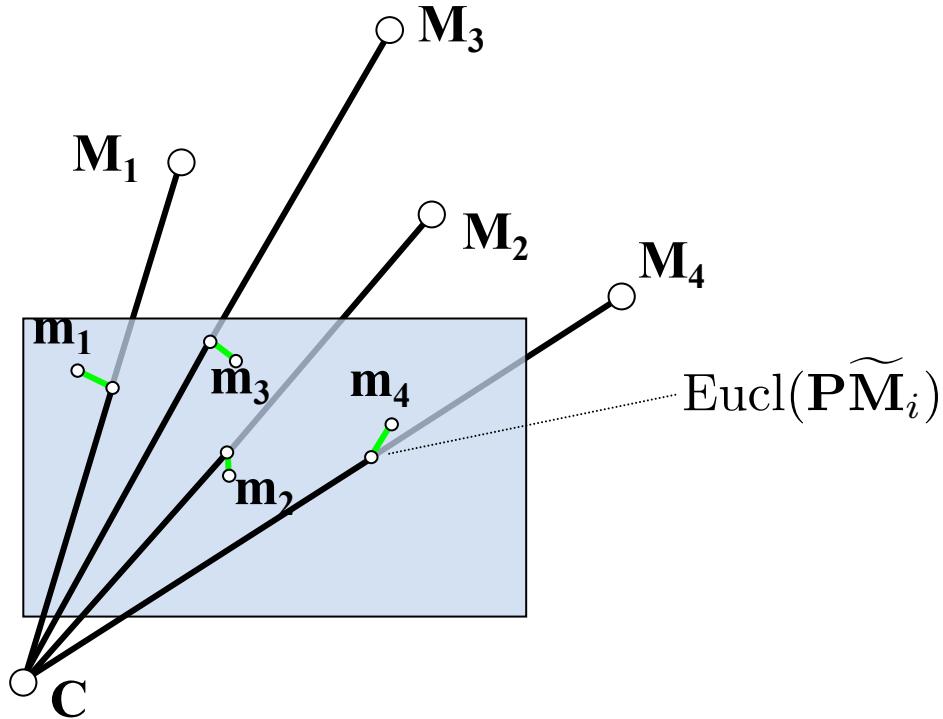
$$\mathbf{M}\mathbf{x} = \mathbf{0}$$

→ \mathbf{x} is a singular vector of \mathbf{M} , with singular value 0 ($\mathbf{M}\mathbf{x} = 0.\mathbf{x}$)

(In practice, we can obtain the singular vectors and their singular values with an SVD operation for example.)

There will be no singular value exactly equal to 0, and we consider the singular value with the smallest absolute value.)

Estimating a 3D projection from correspondences between 3D points and their 2D projections



What does a correspondence give us?

$$\mathbf{M} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \mathbf{m} = \begin{pmatrix} x \\ y \end{pmatrix}, \mathbf{P} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix}$$

$$\begin{pmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -x \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -yX & -yY & -yZ & -y \end{pmatrix} \begin{pmatrix} P_{11} \\ \vdots \\ P_{34} \end{pmatrix} = \mathbf{0}$$

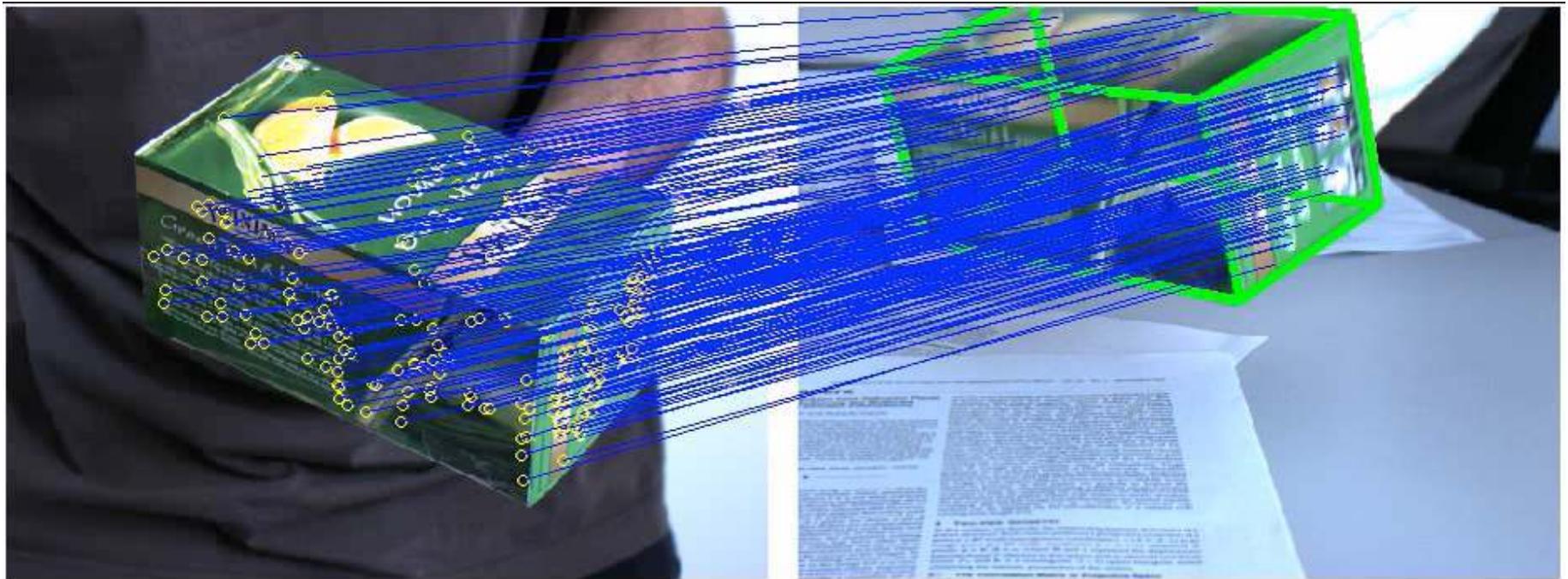
2 linear equations in 12 unknowns

Minimal problem: $|\mathcal{M}| = 6$

$$\mathbf{Mx} = \mathbf{0}$$

Solution #2 (the one that relies on singular vectors) is called the Direct Linear Transform algorithm.

Example



The 3D locations of the yellow points are known in the object coordinate system

From the matches, we can estimate \mathbf{K} in the object coordinate system

Can we retrieve K from P? [R T]?

Given a projection matrix:

$$\mathbf{P} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix}$$

can we retrieve its decomposition

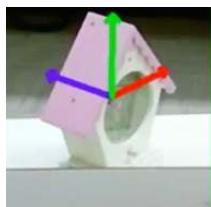
$$\mathbf{P} \equiv \underbrace{\begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{K}} \underbrace{\begin{pmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{pmatrix}}_{[\mathbf{R} \ \mathbf{T}]}$$

? Why is it interesting in practice?

Using **K**, **R**, and **T**



R and **T** give the location and orientation of the camera. This is interesting for robot self-localization for example.



If **R** and **T** are in the coordinate system of an object, we know the location and orientation of the camera with respect to the object. This is interesting for grasping for example.



K, **R**, and **T** are also required to generate images in Augmented Reality.

Can we retrieve K from P? [R T]?

Given a projection matrix:

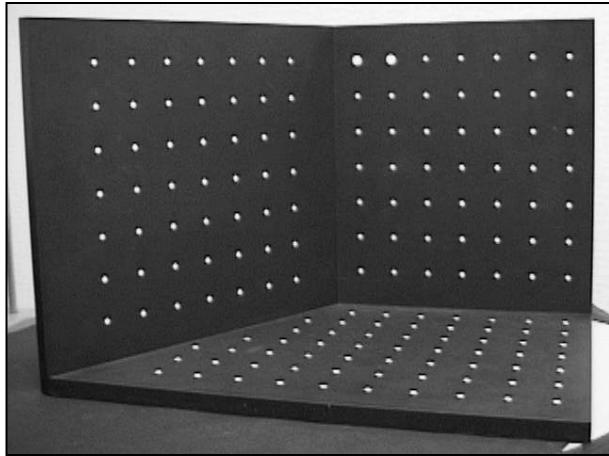
$$\mathbf{P} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix}$$

can we retrieve its decomposition

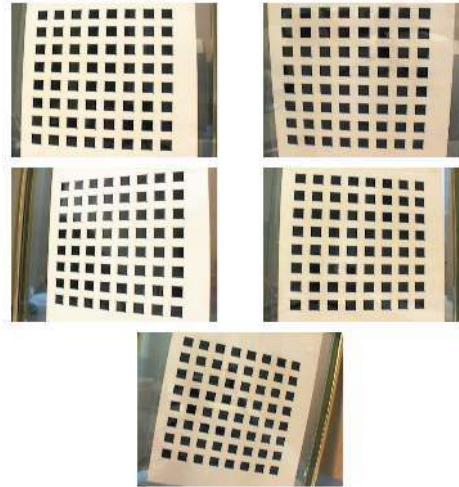
$$\mathbf{P} \equiv \underbrace{\begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{K}} \underbrace{\begin{pmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{pmatrix}}_{[\mathbf{R} \ \mathbf{T}]}$$

→ Yes! See ANNEX A.

Estimating K in practice



using correspondences of 3D points and their reprojections



using Zhengyou Zhang's method and multiple views of a planar pattern

Numerical Optimization

Numerical Optimization

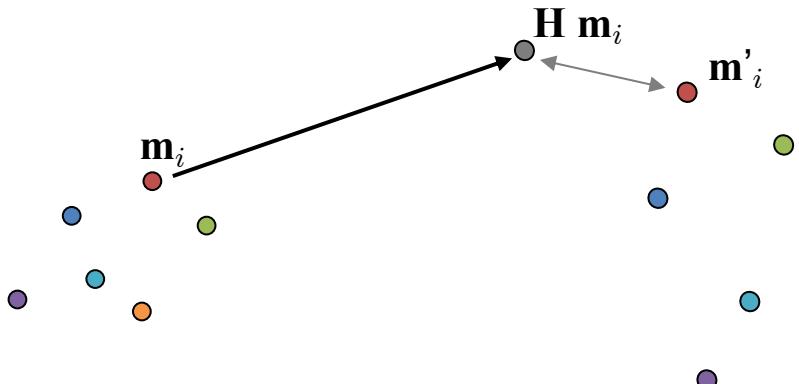
Non-minimal problems: when more correspondences than theoretically needed are available.

It is interesting to exploit all the correspondences to compensate the noise on the detection locations.

Non-Linear Least-Squares Optimization

$$\arg \min_{\mathbf{H}} \frac{1}{|\mathcal{M}|} \sum_i \|\text{Eucl}(\mathbf{H}\tilde{\mathbf{m}}_i) - \mathbf{m}'_i\|^2$$

- Minimizes a meaningful error (reprojection error, in pixels);
- Handles an arbitrary number of points;
- Can be very accurate.



Non-Linear Least-Squares Optimization: General Form

$$\text{error}(\mathbf{H}) = \frac{1}{|\mathcal{M}|} \sum_i \|\text{Eucl}(\mathbf{H}\tilde{\mathbf{m}}_i) - \mathbf{m}'_i\|^2$$

$$\text{error}(\mathbf{h}) = \frac{1}{|\mathcal{M}|} \|f(\mathbf{h}) - \mathbf{b}\|^2$$

$$\text{with } \mathbf{h} = \begin{bmatrix} H_{11} \\ \vdots \\ H_{33} \end{bmatrix}, f(\mathbf{h}) = \begin{bmatrix} u(\text{Eucl}(\mathbf{H}_h \tilde{\mathbf{m}}_1)) \\ v(\text{Eucl}(\mathbf{H}_h \tilde{\mathbf{m}}_1)) \\ \vdots \end{bmatrix}, \mathbf{b} = \begin{bmatrix} u(\mathbf{m}'_1) \\ v(\mathbf{m}'_1) \\ \vdots \end{bmatrix}$$

The Linear Case

$$\text{error}(\mathbf{H}) = \frac{1}{|\mathcal{M}|} \|f(\mathbf{h}) - \mathbf{b}\|^2$$

If we assume that f is linear, then $f(\mathbf{h}) = \mathbf{M}\mathbf{h}$. The problem becomes

$$\arg \min_{\mathbf{h}} \frac{1}{|\mathcal{M}|} \|\mathbf{M}\mathbf{h} - \mathbf{b}\|^2$$

and can be solved by using the pseudo-inverse of \mathbf{M} , or (better) using the `numpy.linalg.lstsq` in Python or `cvSolve` in OpenCV.

The Non-Linear Case

$$\text{error}(\mathbf{H}) = \frac{1}{|\mathcal{M}|} \|f(\mathbf{h}) - \mathbf{b}\|^2$$

In general, requires an iterative optimization: We start from a first estimate \mathbf{h}_0 , which we will try to improve iteratively.

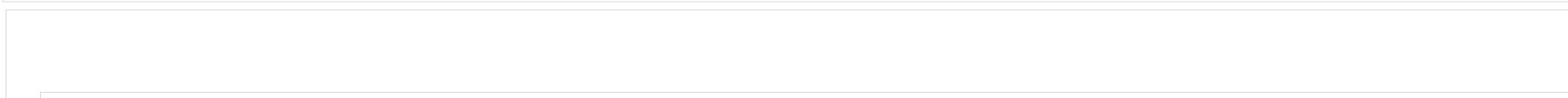
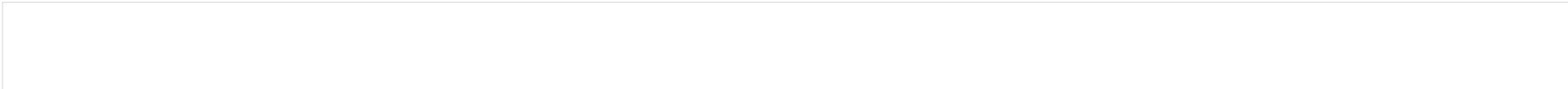
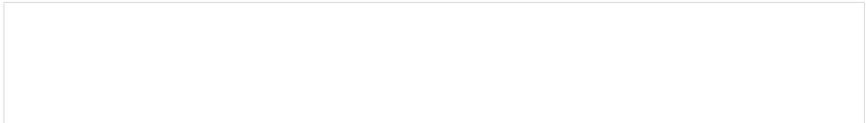
\mathbf{h}_0 can be given by a non-optimal solution, for example.

The Gauss-Newton Algorithm

$$\arg \min_{\mathbf{h}} \frac{1}{|\mathcal{M}|} \|f(\mathbf{h}) - \mathbf{b}\|^2$$

Steps: $\mathbf{h}_{i+1} \leftarrow \mathbf{h}_i + \Delta_i$

How can we find a good Δ_i ?

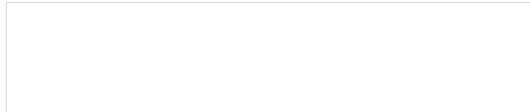


Non-Linear Least-Squares Optimization: Rotation and Translation

$$\text{error}(\mathbf{R}, \mathbf{T}) = \frac{1}{|\mathcal{M}|} \sum_i \|\text{Eucl}(\mathbf{K}[\mathbf{R} \ \mathbf{T}] \widetilde{\mathbf{M}}_i) - \mathbf{m}_i\|^2$$

$$\text{error}(\mathbf{p}) = \frac{1}{|\mathcal{M}|} \|f(\mathbf{p}) - \mathbf{b}\|^2$$

$$\text{with } f(\mathbf{p}) = \begin{bmatrix} u(\text{Eucl}(\mathbf{K}[\mathbf{R}_{\mathbf{p}} \ \mathbf{T}_{\mathbf{p}}] \widetilde{\mathbf{M}}_i)) \\ v(\text{Eucl}(\mathbf{K}[\mathbf{R}_{\mathbf{p}} \ \mathbf{T}_{\mathbf{p}}] \widetilde{\mathbf{M}}_i)) \\ \vdots \end{bmatrix}, \mathbf{b} = \begin{bmatrix} u(\mathbf{m}_1) \\ v(\mathbf{m}_1) \\ \vdots \end{bmatrix}$$



Parameterizing a Rotation Matrix

- Matrix coefficients $[R_{11}, \dots, R_{33}]^T$:

9 values, needs additional constraints to get a rotation matrix.

- Euler Angles:

3 values only, but some technical problems (gimbal lock).

- A unit quaternion:

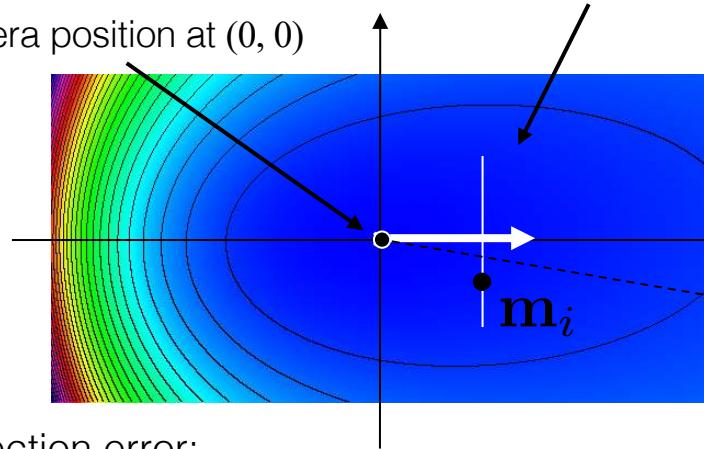
4 values + 1 constraint.

- Exponential Map:

3 values.

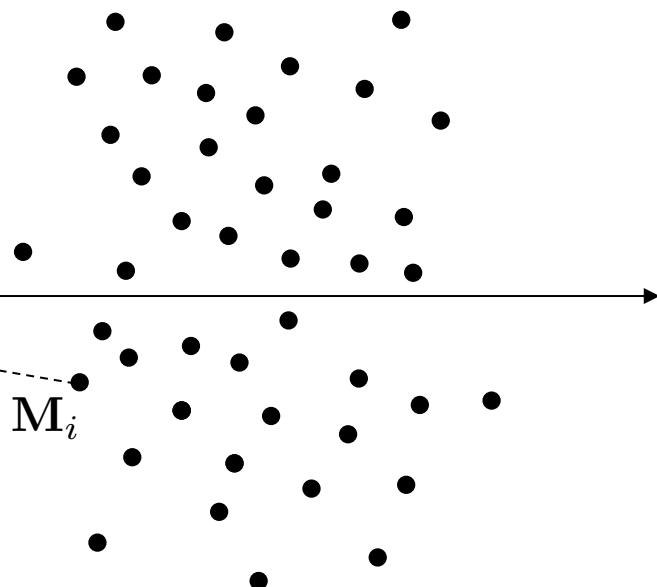
Toy Problem

True camera position at $(0, 0)$



reprojection error:

$$\text{error}(\mathbf{p}) = \frac{1}{|\mathcal{M}|} \sum_i \|\text{Eucl}(\mathbf{K}[\mathbf{R} \ \mathbf{T}] \widetilde{\mathbf{M}}_i) - \mathbf{m}_i\|^2 ,$$



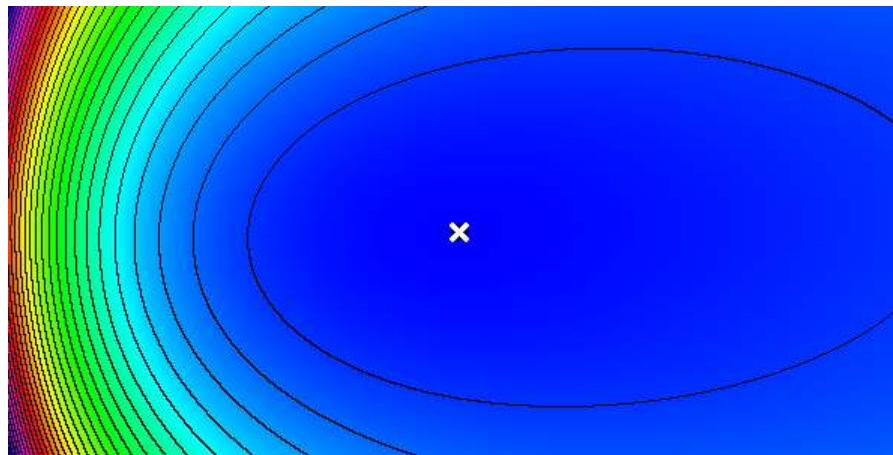
100 "3D points" randomly sampled in
 $[400;1000] \times [-500;+500]$

$$\text{with } \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{T} = \begin{bmatrix} X \\ Y \end{bmatrix}, \text{ and } \mathbf{p} = \begin{bmatrix} X \\ Y \end{bmatrix}$$

Gaussian Noise on the Projections

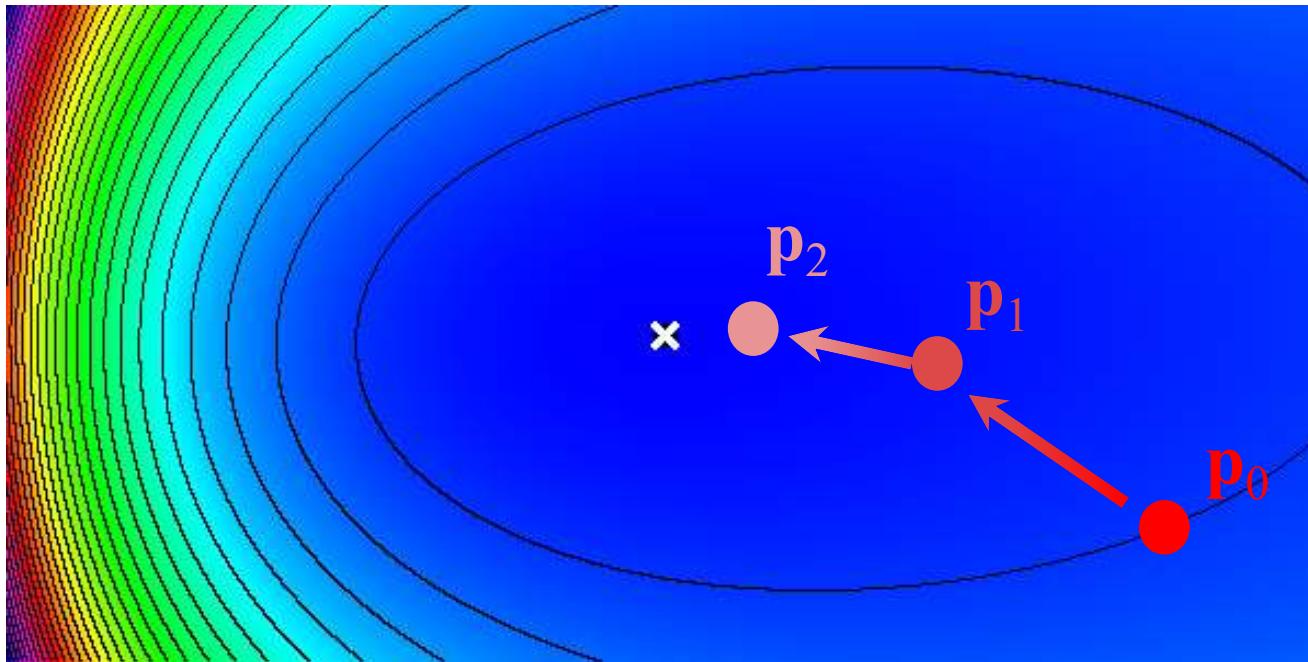
White cross: true camera position;

Black cross: global minimum of the objective function.



In that case, the global minimum of the objective function is close to the true camera pose.

Numerical Optimization



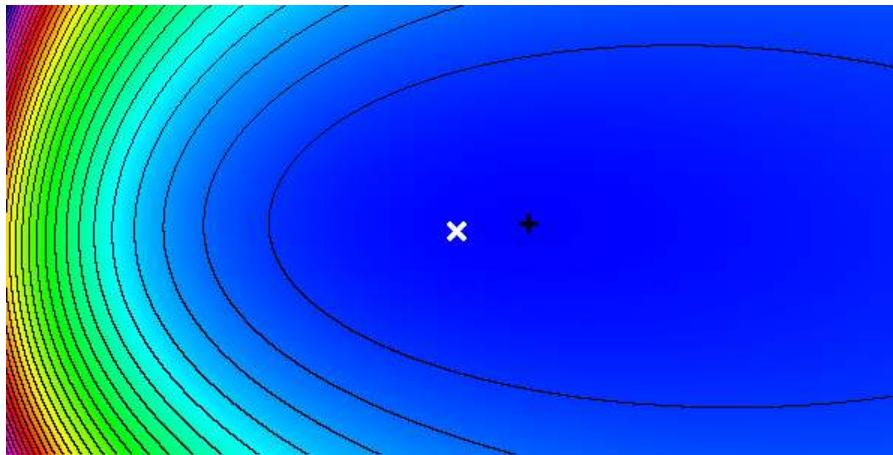
What if there are outliers?



Gaussian Noise on the Projections + 20% outliers

White cross: true camera position;

Black cross: global minimum of the objective function.



The global minimum is now far from the true camera pose.

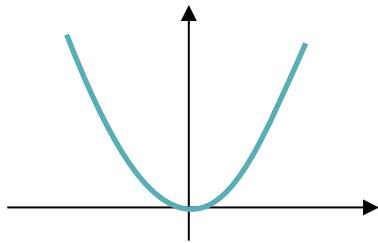
Solution: Robust Estimators

Replace: $\arg \min_{\mathbf{p}} \frac{1}{|\mathcal{M}|} \|f(\mathbf{p}) - \mathbf{b}\|^2$

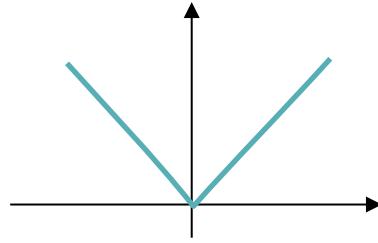
by: $\arg \min_{\mathbf{p}} \frac{1}{|\mathcal{M}|} \rho(f(\mathbf{p}) - \mathbf{b})$

where $\rho(\cdot)$ is a robust estimator

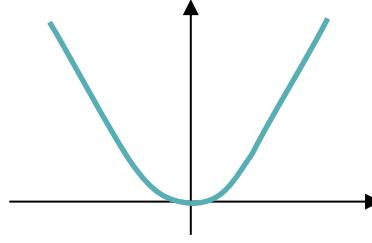
Examples of robust estimators



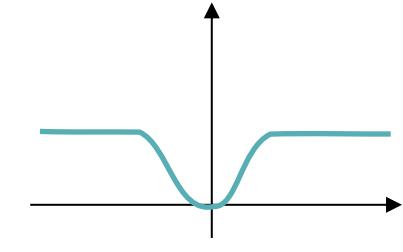
Least-squares (L2)
not a robust estimator,
here only for reference



L1



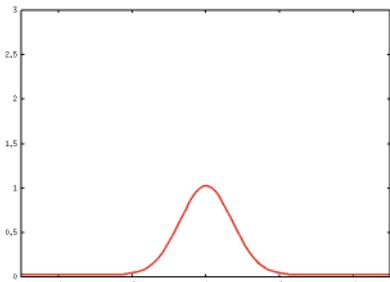
Huber
("smooth L1")



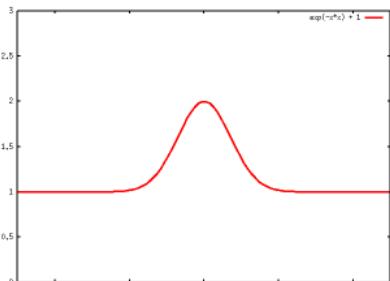
Tukey

$$\rho(x) = \begin{cases} \frac{c^2}{6} \left(1 - \left(1 - \left(\frac{x}{c}\right)^2\right)^3\right) & \text{if } |x| \leq c, \\ \frac{c^2}{6} & \text{otherwise.} \end{cases}$$

Probabilistic interpretation



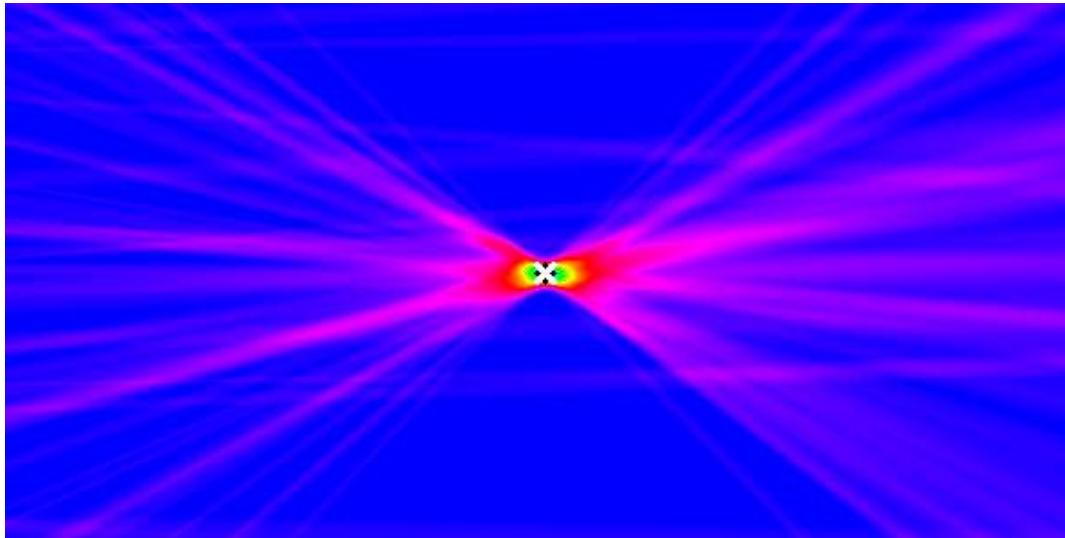
The least-squares estimator assumes the errors follow a Gaussian distribution;



The Tukey estimator assumes that the errors follow a mixture of a Gaussian distribution and a uniform distribution.

see ANNEX D.

Gaussian Noise on the Projections + 50% outliers and Tukey estimator



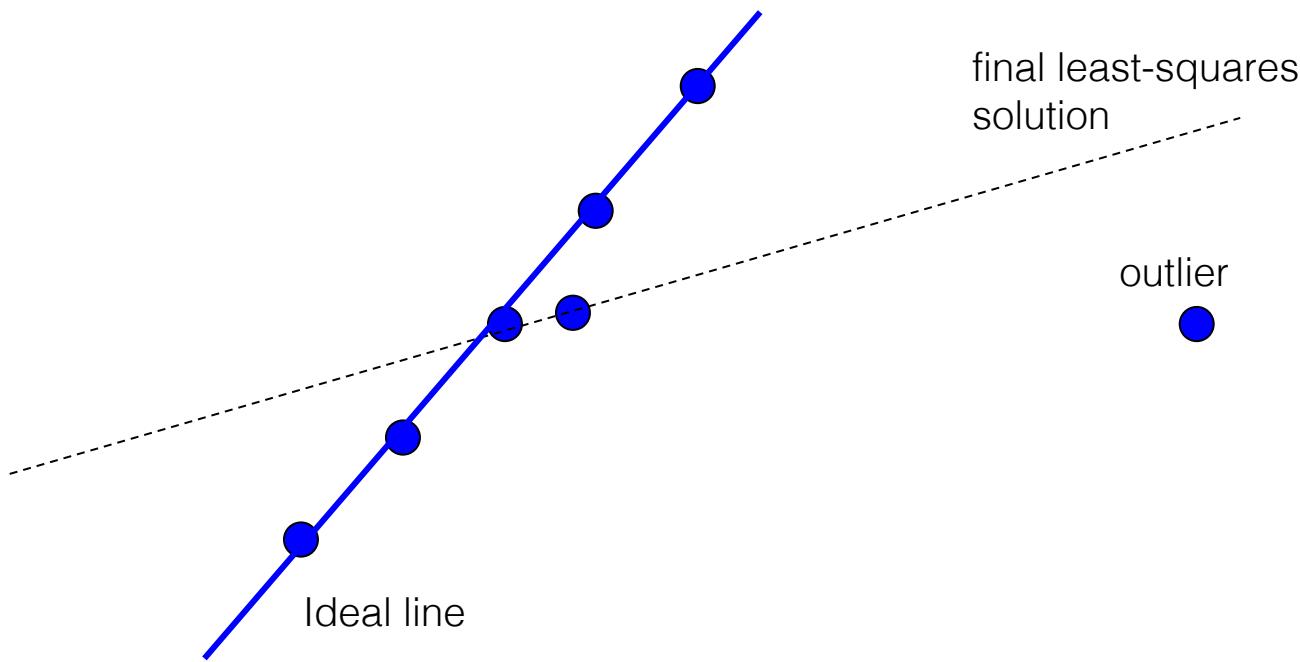
The global minimum is very close to the true camera pose again.

However, there are now many local minimums and regions where the function is flat.



How can we find a good initialization for the numerical optimization despite the outliers?

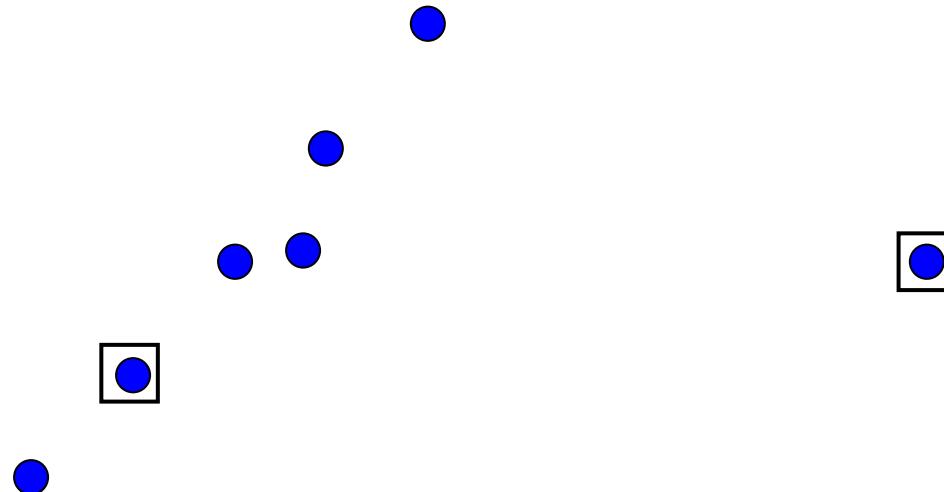
Problem on 2D line fitting



RANSAC

Idea:

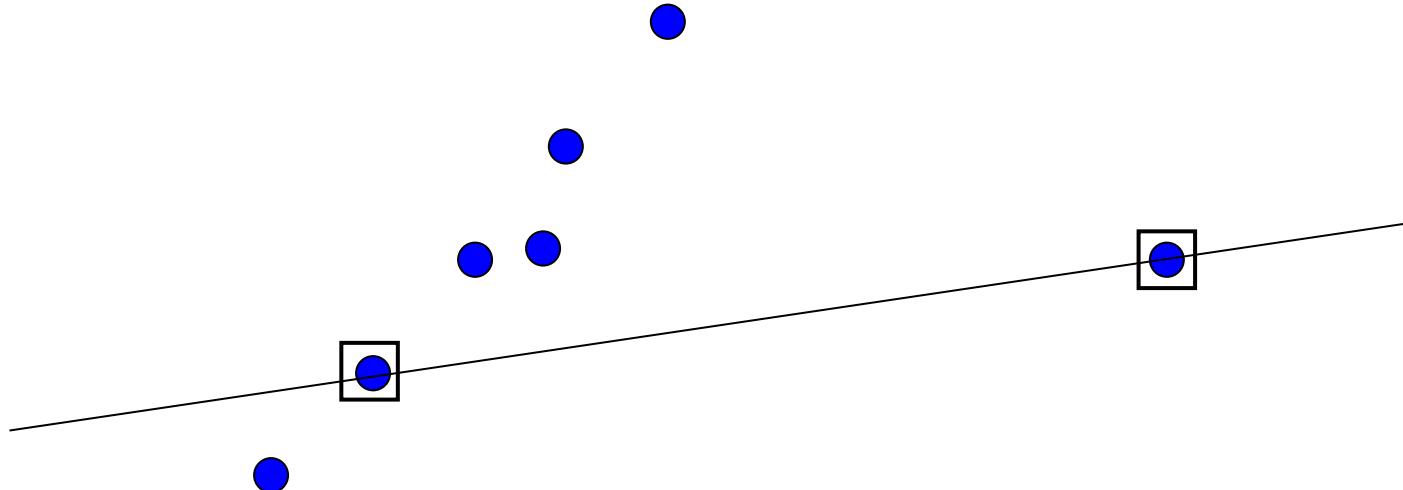
1. sample a minimal set of measures (2 here);



RANSAC

Idea:

1. sample a minimal set of measures (2 2D points here);
2. compute the model parameters from these measures (line parameters here)

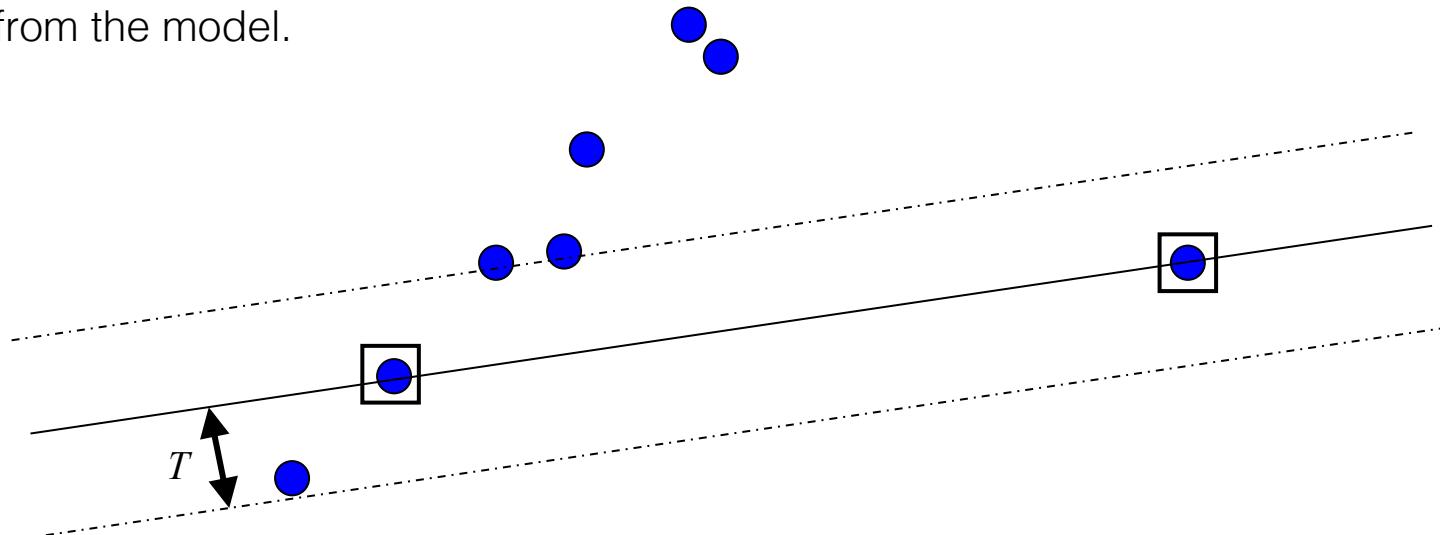


RANSAC

Idea:

1. sample a minimal set of measures (2 here);
2. compute the model parameters from these measures (line parameters here);
3. evaluate how well these parameters explain the other measures;

This is done by choosing a threshold and counting how many measures lie below this threshold from the model.

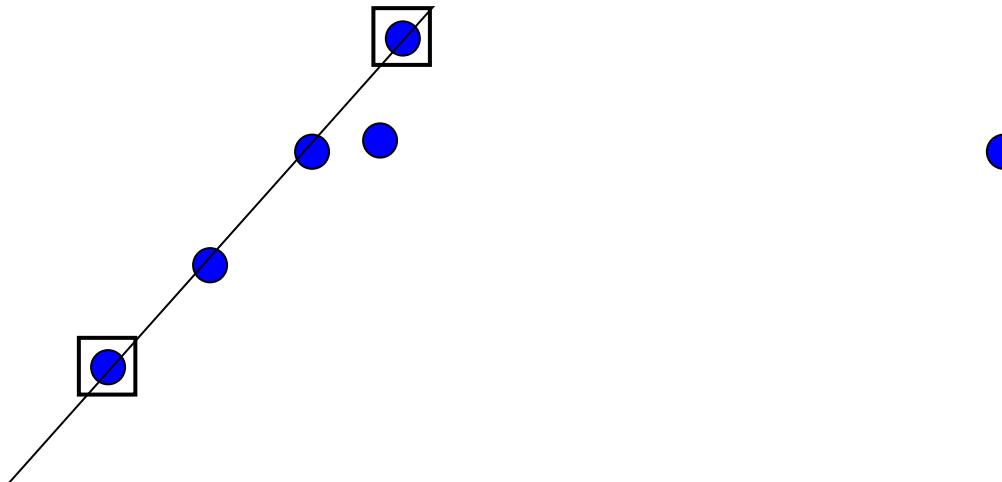


RANSAC

Idea:

1. sample a minimal set of measures (2 here);
2. compute the model parameters from these measures (line parameters here);
3. evaluate how well these parameters explain the other measures;
4. iterate and keep the model parameters that explain the most measures.

When there is no outlier in the minimal set, the computed model parameters will be close to the correct ones.



ANNEX A

Can we retrieve K from P? [R T]?

Given a projection matrix:

$$\mathbf{P} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix}$$

can we retrieve its decomposition

$$\mathbf{P} \equiv \underbrace{\begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{K}} \underbrace{\begin{pmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{pmatrix}}_{[\mathbf{R} \ \mathbf{T}]}$$

→ Yes.

Retrieving \mathbf{K} given \mathbf{P}

Consider matrix \mathbf{P}_3 made of the 3 first columns of \mathbf{P} : $\mathbf{P} = [\mathbf{P}_3 \ \mathbf{c}_4]$

We have $\mathbf{P} \equiv \mathbf{K}[\mathbf{R} \ \mathbf{T}] \equiv [\mathbf{K}\mathbf{R} \ \mathbf{K}\mathbf{T}]$, so $\mathbf{P}_3 \equiv \mathbf{K}\mathbf{R}$

What happens if we compute $\mathbf{P}_3\mathbf{P}_3^T$?

$$\mathbf{P}_3\mathbf{P}_3^T \equiv (\mathbf{K}\mathbf{R})(\mathbf{K}\mathbf{R})^T \equiv \mathbf{K}\mathbf{R}\mathbf{R}^T\mathbf{K}^T \equiv \mathbf{K}\mathbf{R}\mathbf{R}^{-1}\mathbf{K}^T \equiv \mathbf{K}\mathbf{K}^T$$

How can we use this equality?

Retrieving \mathbf{K} given \mathbf{P} (2)

We now know that $\mathbf{P}_3\mathbf{P}_3^T \equiv \mathbf{KK}^T$.

Given the coefficients of \mathbf{P} , we can compute the coefficients of $\mathbf{P}_3\mathbf{P}_3^T$:

$$\mathbf{P}_3\mathbf{P}_3^T = \begin{pmatrix} L_{11} & L_{12} & L_{13} \\ L_{12} & L_{22} & L_{23} \\ L_{13} & L_{23} & L_{33} \end{pmatrix}$$

These coefficients are the coefficients of \mathbf{KK}^T (up to a scale factor):

$$\mathbf{KK}^T = \begin{pmatrix} \alpha_u^2 + u_0^2 & u_0 v_0 & u_0 \\ u_0 v_0 & \alpha_v^2 + v_0^2 & v_0 \\ u_0 & v_0 & 1 \end{pmatrix}$$

Retrieving \mathbf{K} given \mathbf{P} (3)

$$\begin{pmatrix} L_{11} & L_{12} & L_{13} \\ L_{12} & L_{22} & L_{23} \\ L_{13} & L_{23} & L_{33} \end{pmatrix} = k \begin{pmatrix} \alpha_u^2 + u_0^2 & u_0 v_0 & u_0 \\ u_0 v_0 & \alpha_v^2 + v_0^2 & v_0 \\ u_0 & v_0 & 1 \end{pmatrix}$$

Term identification gives us a system of equations:

$$\left\{ \begin{array}{l} k(\alpha_u^2 + u_0^2) = L_{11} \\ k(u_0 v_0) = L_{12} \\ k u_0 = L_{13} \\ k(\alpha_v^2 + v_0^2) = L_{22} \\ k v_0 = L_{23} \\ k = L_{33} \end{array} \right.$$

which can be solved easily.

Retrieving \mathbf{R} and \mathbf{T} from \mathbf{P} and \mathbf{K}

$\mathbf{P} \equiv \mathbf{K}[\mathbf{R} \ \mathbf{T}]$ i.e. $k\mathbf{P} = \mathbf{K}[\mathbf{R} \ \mathbf{T}]$ with k an unknown factor, so $[\mathbf{R} \ \mathbf{T}] = k \ \mathbf{K}^{-1} \ \mathbf{P}$

k can be estimated by using the fact the norms of the columns and rows of \mathbf{R} are equal to 1.

In practice, this does not give *exactly* a rotation matrix, but it is possible to use an orthogonalization method.

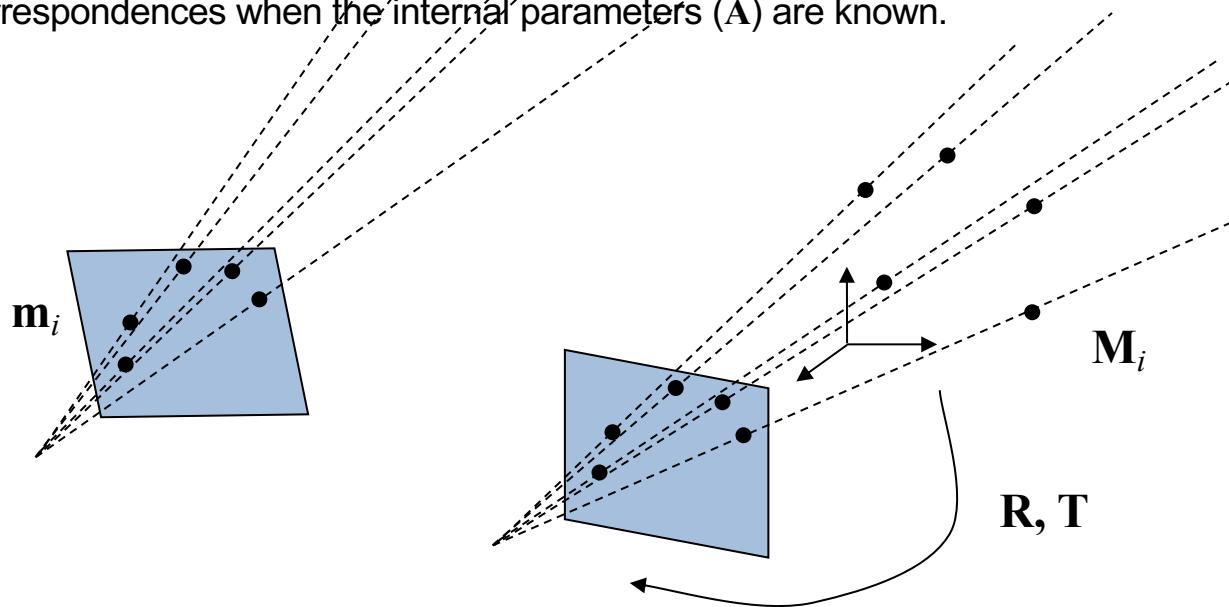
ANNEX B

Estimating a rotation and translation from pairs of 3D points and their 2D projections when K is known

P-n-P Problem

PnP: Perspective- n -Point

Estimation of the Position and Orientation (\mathbf{R} and \mathbf{T}) from 2D/3D point correspondences when the internal parameters (\mathbf{A}) are known.



P-n-P Problem

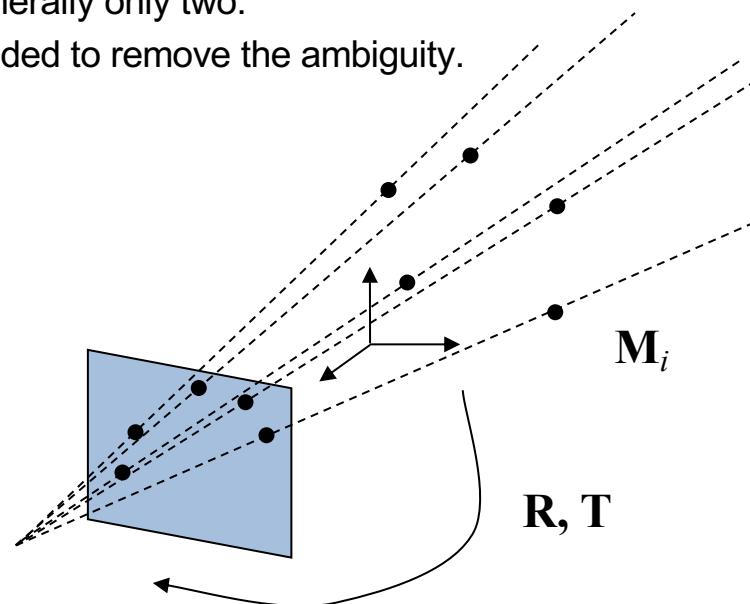
P_nP: Perspective-*n*-Point

Estimation of the Position and Orientation (\mathbf{R} and \mathbf{T}) from 2D/3D point correspondences when the internal parameters (\mathbf{A}) are known.

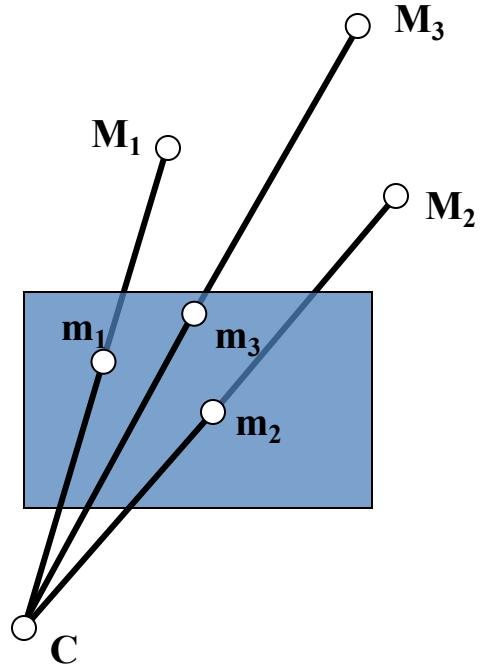
3 correspondences are sufficient (6 equations for 6 parameters),

BUT yield up to 4 solutions, generally only two.

A fourth correspondence is needed to remove the ambiguity.



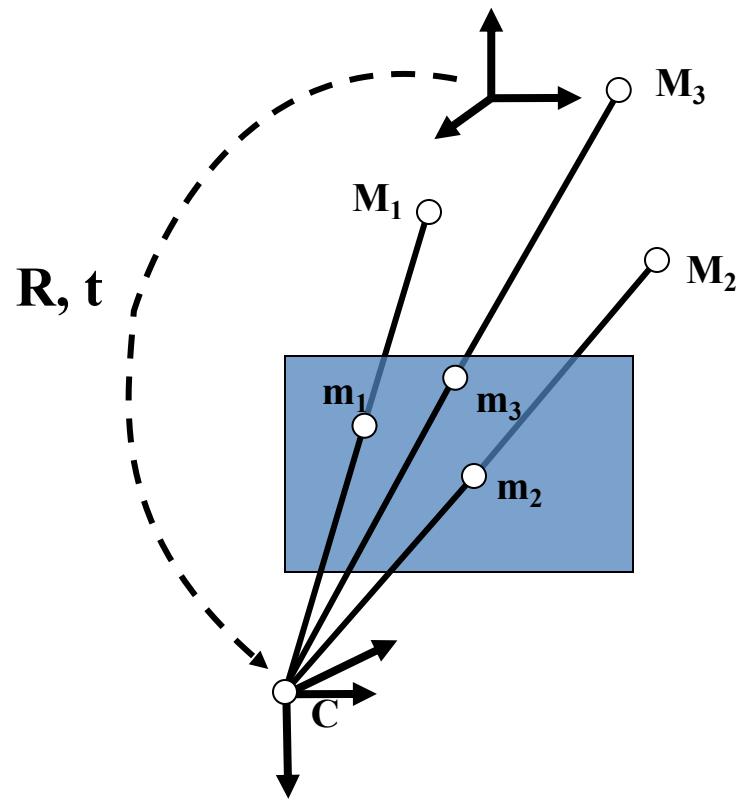
The P3P Problem



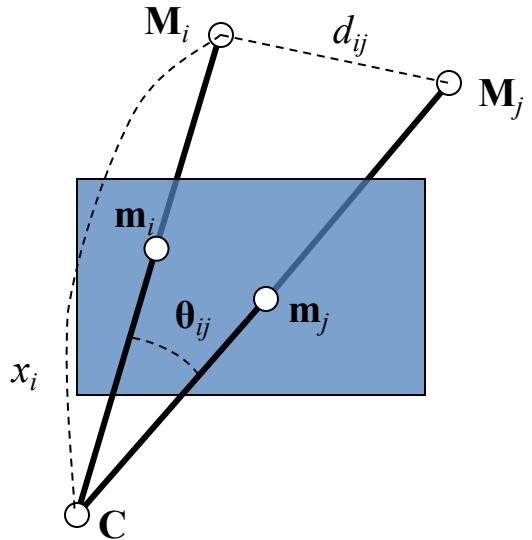
P3P: Pose estimation from 3 correspondences.

Yields up to 4 solutions, generally only two.

A fourth correspondence will be needed to remove the ambiguity.



The P3P Problem



Each pair of correspondences $\mathbf{M}_i \leftrightarrow \mathbf{m}_i$ and $\mathbf{M}_j \leftrightarrow \mathbf{m}_j$ gives a constraint on the (unknown) camera-point distances $x_i = \|\mathbf{M}_i - \mathbf{C}\|$ and $x_j = \|\mathbf{M}_j - \mathbf{C}\|$:

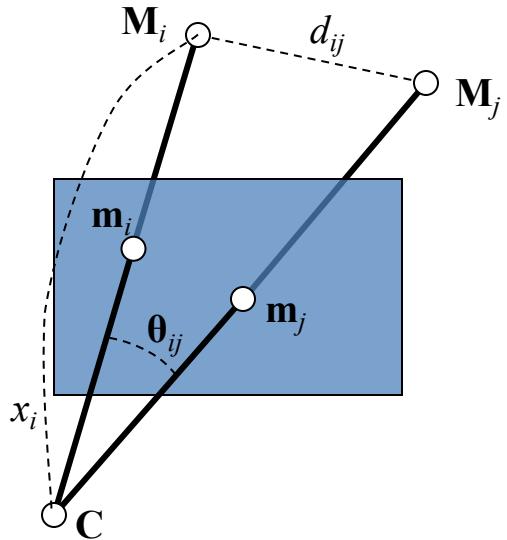
$$d_{ij}^2 = x_i^2 + x_j^2 - 2 x_i x_j \cos\theta_{ij},$$

where:

$d_{ij} = \|\mathbf{M}_i - \mathbf{M}_j\|$ is the (known) distance between \mathbf{M}_i and \mathbf{M}_j ;

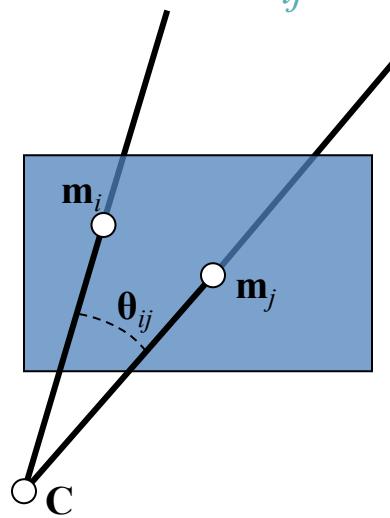
θ_{ij} is the (also known) angle subtended at the camera center by \mathbf{M}_i and \mathbf{M}_j .

Overview of the P3P



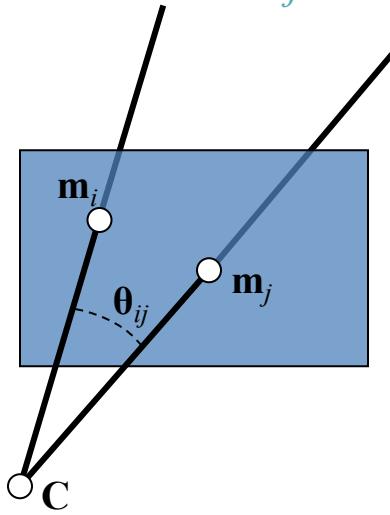
1. Solve for the distances x_i ;
2. The positions \mathbf{M}_i^C of the points \mathbf{M}_i in the camera coordinates system can be computed;
3. \mathbf{R} and \mathbf{T} are computed as the Euclidean displacement from the \mathbf{M}_i to the \mathbf{M}_i^C .

Computation of $\cos\theta_{ij}$



$\cos\theta_{ij}$ must be computed. It depends only on the (known) 2D positions m_i and m_j .

Computation of $\cos\theta_{ij}$



$\cos\theta_{ij}$ must be computed. It depends only on the (known) 2D positions \mathbf{m}_i and \mathbf{m}_j .

$$\cos\theta_{ij} = \frac{(\overrightarrow{\mathbf{Cm}}_j)^T (\overrightarrow{\mathbf{Cm}}_i)}{\|\overrightarrow{\mathbf{Cm}}_j\| \|\overrightarrow{\mathbf{Cm}}_i\|} = \frac{(\overrightarrow{\mathbf{Cm}}_j)^T (\overrightarrow{\mathbf{Cm}}_i)}{\sqrt{(\overrightarrow{\mathbf{Cm}}_j)^T (\overrightarrow{\mathbf{Cm}}_j)} \sqrt{(\overrightarrow{\mathbf{Cm}}_i)^T (\overrightarrow{\mathbf{Cm}}_i)}}$$

$$\overrightarrow{\mathbf{Cm}}_i = \mathbf{A}^{-1} \mathbf{m}_i$$

$$\cos\theta_{ij} = \frac{\mathbf{m}_j^T \omega \mathbf{m}_i}{(\mathbf{m}_j^T \omega \mathbf{m}_j)^{1/2} (\mathbf{m}_i^T \omega \mathbf{m}_i)^{1/2}} \text{ with } \omega = (\mathbf{A} \mathbf{A}^T)^{-1} \text{ the image of the absolute conic.}$$

Quadratic System

$$d_{ij}^2 = x_i^2 + x_j^2 - 2 x_i x_j \cos\theta_{ij} \rightarrow f_{ij}(x_i, x_j) = x_i^2 + x_j^2 - 2 x_i x_j \cos\theta_{ij} - d_{ij}^2 = 0$$

$$\begin{cases} f_{12}(x_1, x_2) = 0 \\ f_{13}(x_1, x_3) = 0 \\ f_{23}(x_2, x_3) = 0 \end{cases}$$

Resultant $\Rightarrow g(x_1, x_2)$ [degree 4]

Resultant $\Rightarrow h(x_1)$ [degree 8]

$h(x_1)$: Polynomial of degree 8 in x_1 with (fortunately) only even terms

Degree polynomial of degree 4 in $x=x_1^2$:

At most 4 solutions for x ;

Can be solved in closed form.

x_1 is positive and $x_1 = \sqrt{x}$. x_2 and x_3 are uniquely determined from x_1 .

To obtain a unique solution, add one more point: Solve the P3P problem for each subset of 3 points, and keep the common solution.

Resultant of Two Polynomials

Sylvester resultant (for two polynomials of a single unknown):

$$p_1(x) = a_m x^m + a_{m-1} x^{m-1} + \dots + a_1 x + a_0$$

$$p_2(x) = b_n x^n + b_{n-1} x^{n-1} + \dots + b_1 x + b_0$$

$$Syl(p_1, p_2) = \begin{bmatrix} a_m & \cdots & a_0 & 0 & \cdots & 0 \\ 0 & a_m & \cdots & a_0 & & \\ & & & \vdots & & \\ & & & & a_m & \cdots & a_0 \\ b_n & \cdots & b_0 & 0 & \cdots & 0 \\ 0 & b_n & \cdots & b_0 & & \\ & & & \vdots & & \\ & & & & b_n & \cdots & b_0 \end{bmatrix}$$

p_1 and p_2 have a common root iff $\text{determinant}(Syl(p_1, p_2)) = 0$.

$\text{determinant}(Syl(p_1, p_2))$ is called the *Sylvester resultant* of p_1 and p_2 .

System of two polynomials of two unknowns

Example:

$$p_1(x, y) = 6x^2 + 3xy - xy^2 + y + 1$$

$$p_2(x, y) = x^2y + 5x + 4y - 1$$

$$\begin{cases} p_1(x, y) = 0 \\ p_2(x, y) = 0 \end{cases}$$

System of two polynomials of two unknowns

Example:

$$p_1(x, y) = 6x^2 + 3xy - xy^2 + y + 1$$

$$p_2(x, y) = x^2y + 5x + 4y - 1$$

$$\begin{cases} p_1(x, y) = 0 \\ p_2(x, y) = 0 \end{cases}$$

Lets consider y as a constant, and write these two polynomials as polynomials in x :

$$p_1(x, y) = 6x^2 + (3y - y^2)x + (y + 1)$$

$$p_2(x, y) = yx^2 + 5x + (4y - 1)$$

$$\begin{vmatrix} 6 & (3y - y^2) & (y + 1) & 0 \\ 0 & 6 & (3y - y^2) & (y + 1) \\ y & 5 & (4y - 1) & 0 \\ 0 & y & 5 & (4y - 1) \end{vmatrix} = 0$$

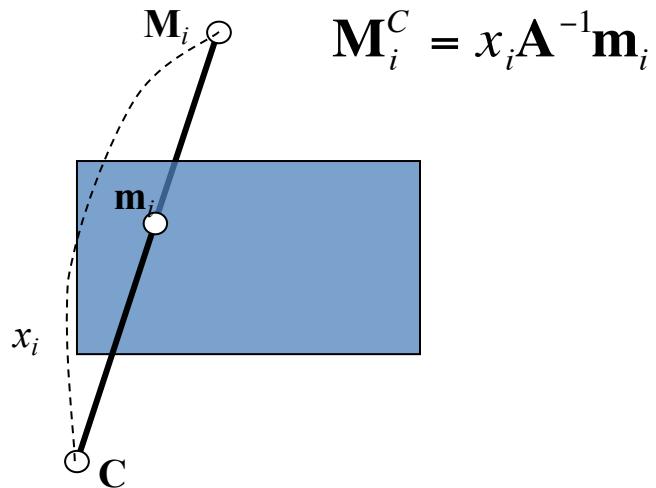
determinant(Syl(p_1, p_2)) = $4y^6 - 20y^5 + 153y^4 - 310y^3 + 781y^2 - 276y + 36$ is a polynomial in y only !

→ First, solve $4y^6 - 20y^5 + 153y^4 - 310y^3 + 781y^2 - 276y + 36 = 0$

Once y is known, x can be found from $p_1(x, y)$ or $p_2(x, y)$.

3D points in the camera coordinates system

The coordinates of the 3D points \mathbf{M}_i in the **camera coordinates system** can now be computed:

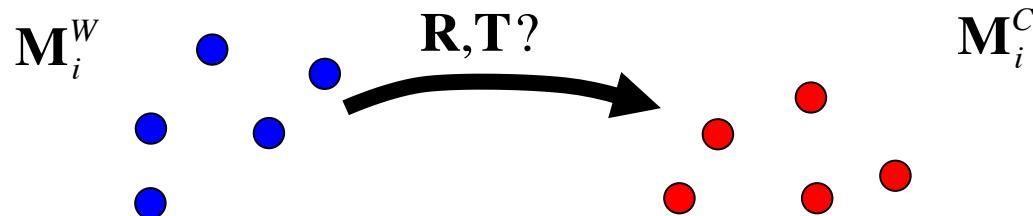


Estimating R and T

We now know:

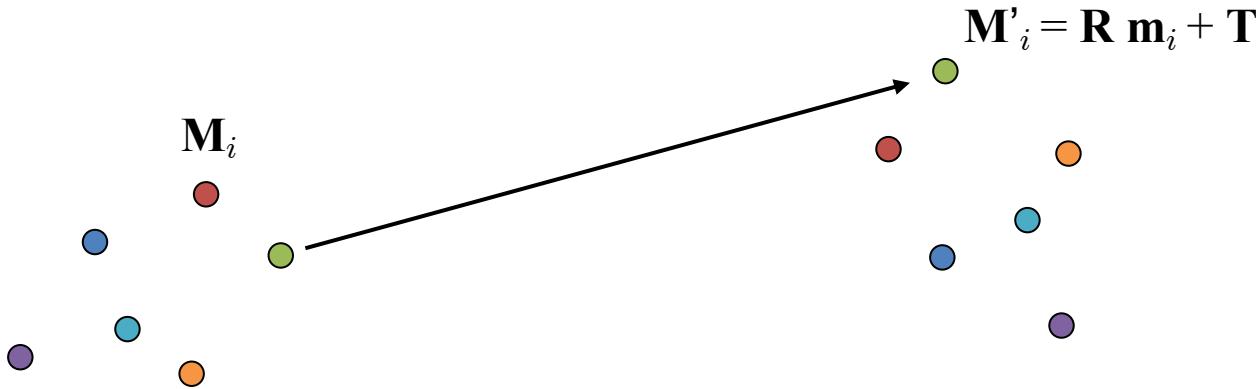
- the coordinates of the 3D points \mathbf{M}_i in the world coordinate system (this is given as input);
- their coordinates in the camera coordinate system (we just computed them).

We can compute R and T as the rotation and translation that transform the world coordinate system into the camera coordinate system (see ANNEX C).



ANNEX C

Estimating a rotation and translation from pairs of 3D points



$$\text{error}(\mathbf{R}, \mathbf{T}) = \frac{1}{|\mathcal{M}|} \sum_i \|\mathbf{RM}_i + \mathbf{T} - \mathbf{M}'_i\|^2$$

$$\arg \min_{\mathbf{R} \in \text{SO}(3), \mathbf{T}} \text{error}(\mathbf{R}, \mathbf{T})$$

$$\text{error}(\mathbf{R}, \mathbf{T}) = \frac{1}{|\mathcal{M}|} \sum_i \|\mathbf{RM}_i + \mathbf{T} - \mathbf{M}'_i\|^2 \quad \arg \min_{\mathbf{R} \in \text{SO}(3), \mathbf{T}} \text{error}(\mathbf{R}, \mathbf{T})$$

From $\frac{\partial \text{error}}{\partial \mathbf{T}} = \mathbf{0}$, we get: $\mathbf{T} = \overline{\mathbf{M}'} - \mathbf{R}\overline{\mathbf{M}}$

$$\overline{\mathbf{M}} = \text{Mean}(\{\mathbf{M}_i\})$$

$$\overline{\mathbf{M}'} = \text{Mean}(\{\mathbf{M}'_i\})$$

→ Once we know \mathbf{R} , we can compute \mathbf{T} .

By plugging expression $\mathbf{T} = \overline{\mathbf{M}'} - \mathbf{R}\overline{\mathbf{M}}$ into $\text{error}(\mathbf{R}, \mathbf{T})$, and by introducing

$$\mathbf{N}_i = \mathbf{M}_i - \overline{\mathbf{M}} \quad \mathbf{N}'_i = \mathbf{M}'_i - \overline{\mathbf{M}'}$$

we obtain:

$$\mathbf{R} = \arg \min_{\mathbf{R} \in \text{SO}(3)} \sum_i \|\mathbf{RN}_i - \mathbf{N}'_i\|^2$$

After some computations:

$$\|\mathbf{R}\mathbf{N}_i - \mathbf{N}'_i\|^2 = .. = \mathbf{N}_i^T \mathbf{N}_i - 2(\mathbf{N}'_i)^T \mathbf{R} \mathbf{N}_i + (\mathbf{N}'_i)^T \mathbf{N}'_i$$

the first and last terms are constant wrt \mathbf{R} , and thus have no influence on the optimization problem and we can ignore them:

$$\mathbf{R} = \arg \max_{\mathbf{R} \in \text{SO}(3)} \sum_i (\mathbf{N}'_i)^T \mathbf{R} \mathbf{N}_i$$

It can be shown that (just expand the two expressions):

$$\sum_i (\mathbf{N}'_i)^T \mathbf{R} \mathbf{N}_i = \text{tr} (\mathbf{R} (\mathbf{N}')^T \mathbf{N})$$

with \mathbf{N} and \mathbf{N}' the matrices of the \mathbf{N}_i and \mathbf{N}'_i

we are thus now looking for

$$\arg \max_{\mathbf{R} \in \text{SO}(3)} \text{tr}(\mathbf{RL}) \text{ with } \mathbf{L} = (\mathbf{N}')^T \mathbf{N}$$

By taking the SVD of \mathbf{L} : $\mathbf{L} = \mathbf{U}\Sigma\mathbf{V}^T$

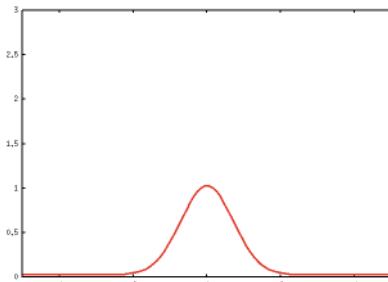
then $\mathbf{R} = \mathbf{UV}^T$

ANNEX D

Least-squares optimization assumes that the errors on the measures follow a Gaussian distribution and are independent:

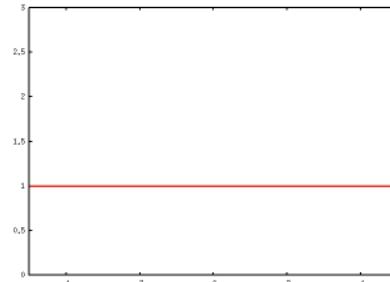
$$\begin{aligned}
& \arg \min_{\mathbf{P}} \frac{1}{|\mathcal{M}|} \sum_i \|\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i\|^2 \\
&= \arg \min_{\mathbf{P}} \sum_i (\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i)^T (\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i) \\
&= \arg \min_{\mathbf{P}} \sum_i (\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i)^T \Sigma (\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i) \text{ with } \Sigma = \begin{bmatrix} \sigma^{-1} & 0 \\ 0 & \sigma^{-1} \end{bmatrix} \\
&= \arg \max_{\mathbf{P}} \sum_i -(\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i)^T \Sigma (\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i) \\
&= \arg \max_{\mathbf{P}} \exp \left(\sum_i -(\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i)^T \Sigma (\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i) \right) \\
&= \arg \max_{\mathbf{P}} \left(\prod_i \exp(\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i)^T \Sigma (\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i) \right) \\
&= \arg \max_{\mathbf{P}} \prod_i \exp \left(\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i \right)^T \Sigma (\text{Eucl}(\mathbf{PM}_i) - \mathbf{m}_i) \\
&= \arg \max_{\mathbf{P}} \prod_i \mathcal{N}(\mathbf{m}_i \mid \text{Eucl}(\mathbf{PM}_i), \Sigma)
\end{aligned}$$

Normal distribution
(inliers)



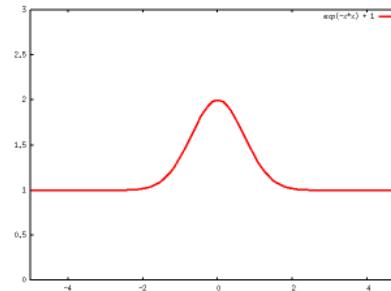
+

Uniform distribution
(outliers)

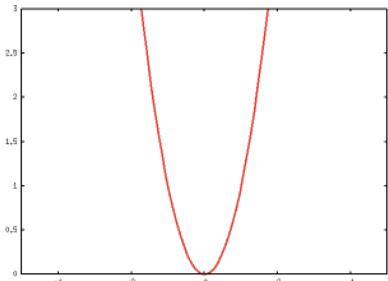


=

Mixture

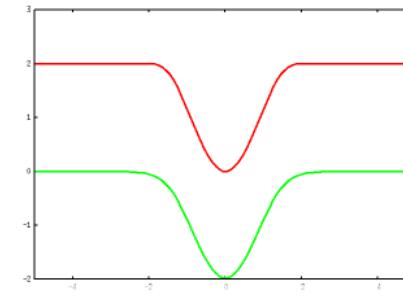


-log(.)



Least-squares

-log(.)



Tukey estimator

