Technischen Universität München
Winter Semester 2009/2010

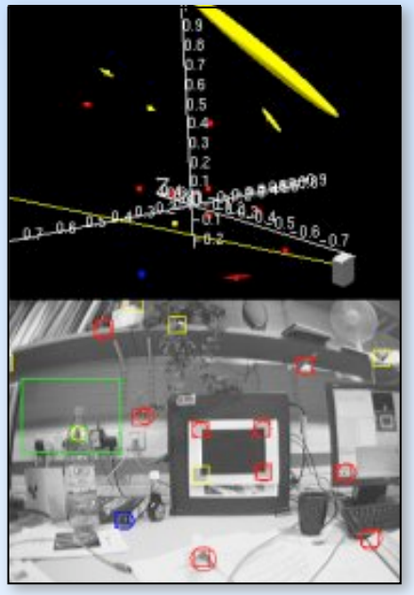
TRACKING and DETECTION in COMPUTER VISION

Camera models and pose estimation

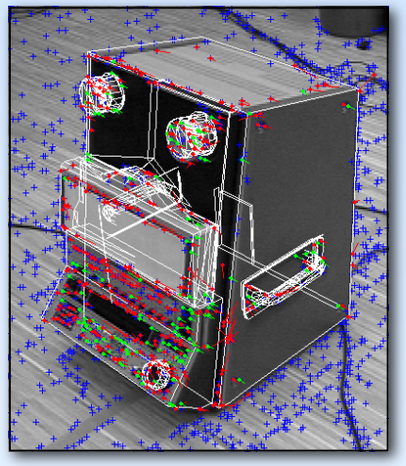
Slobodan Ilić

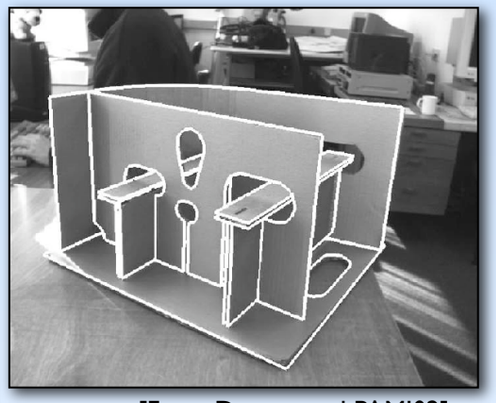
Slides adapted from V.Lepetit, EPFL

Different Approaches to Vision-based 3D Tracking

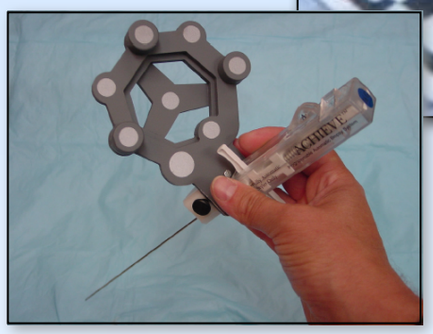


[From Davison ICCV01]





[From Drummond PAMI02]



[From D.Wagner]

Consider natural features

No *a priori* 3D knowledge

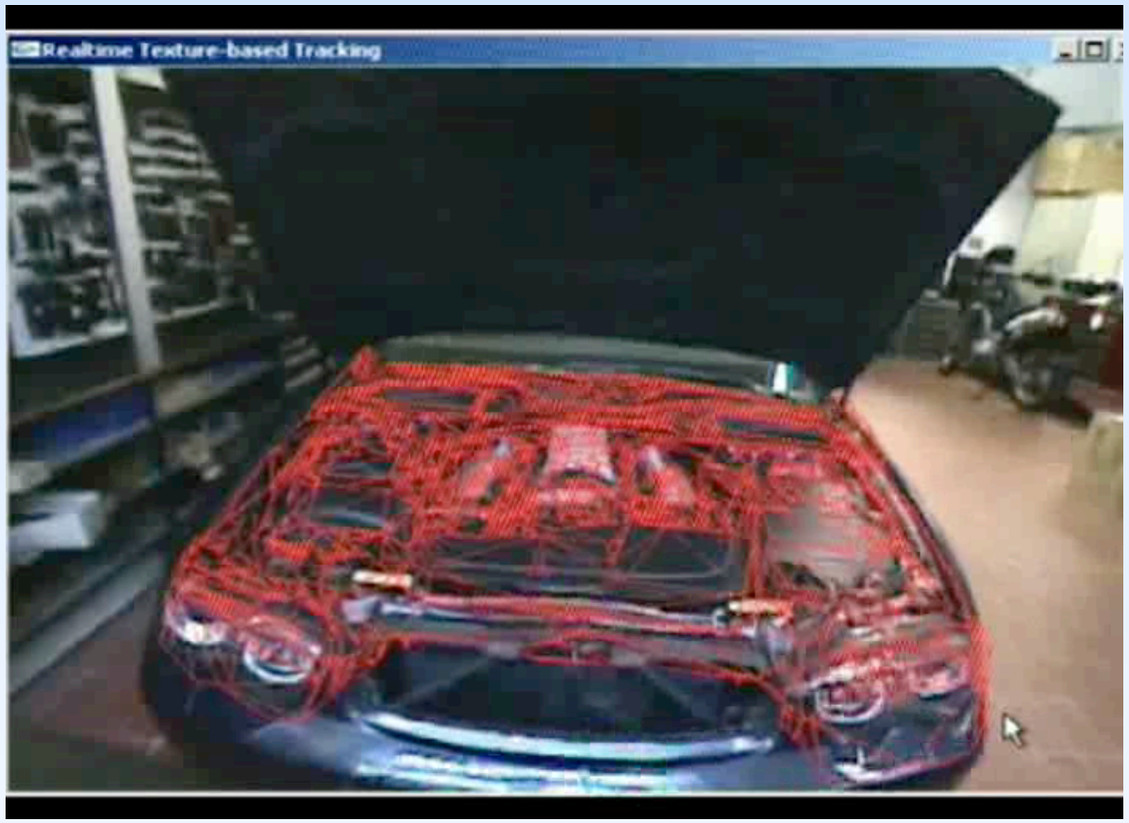
Consider natural features

Use some 3D knowledge

Make use of visual markers

Use some 3D knowledge

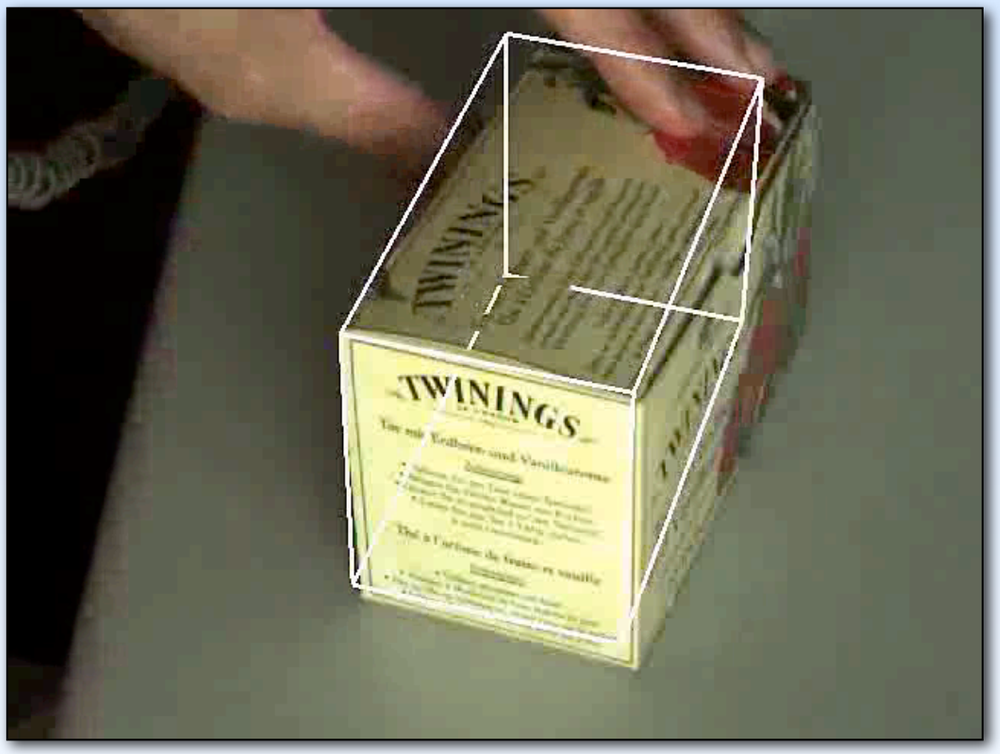
Classical Problems in Vision-Based Tracking



Initialization is tricky!

Tracking can easily get lost!

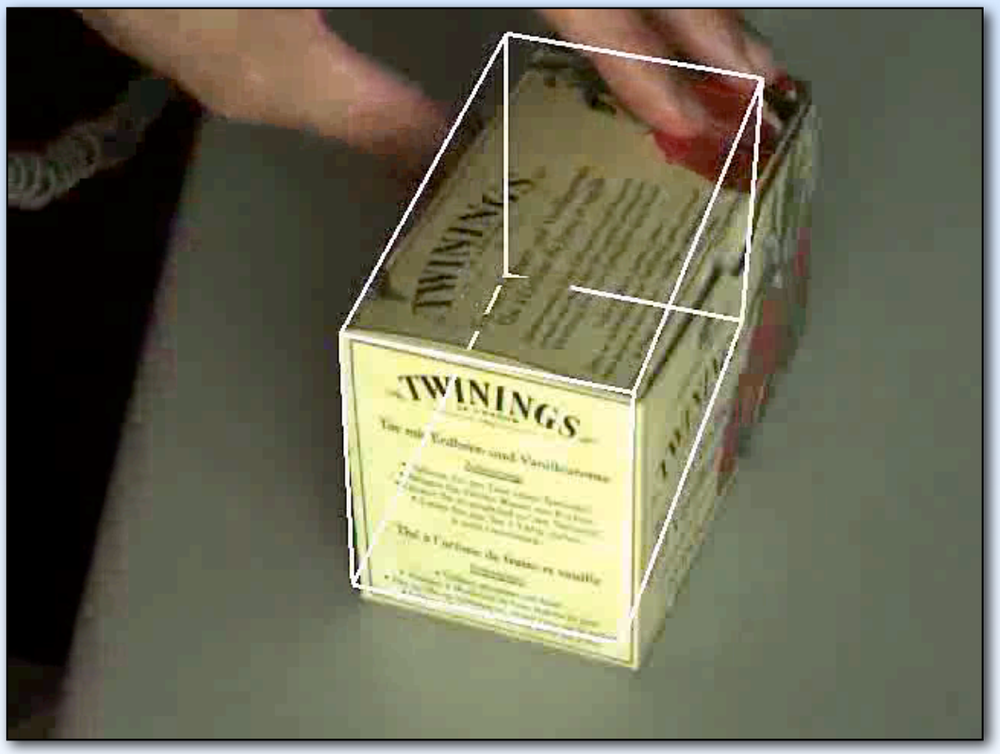
Classical Problems in Vision-Based Tracking



Drift: Error accumulation

Jittering: Inaccuracies

Classical Problems in Vision-Based Tracking



Drift: Error accumulation

Jittering: Inaccuracies

Classical Problems in Vision-Based Tracking

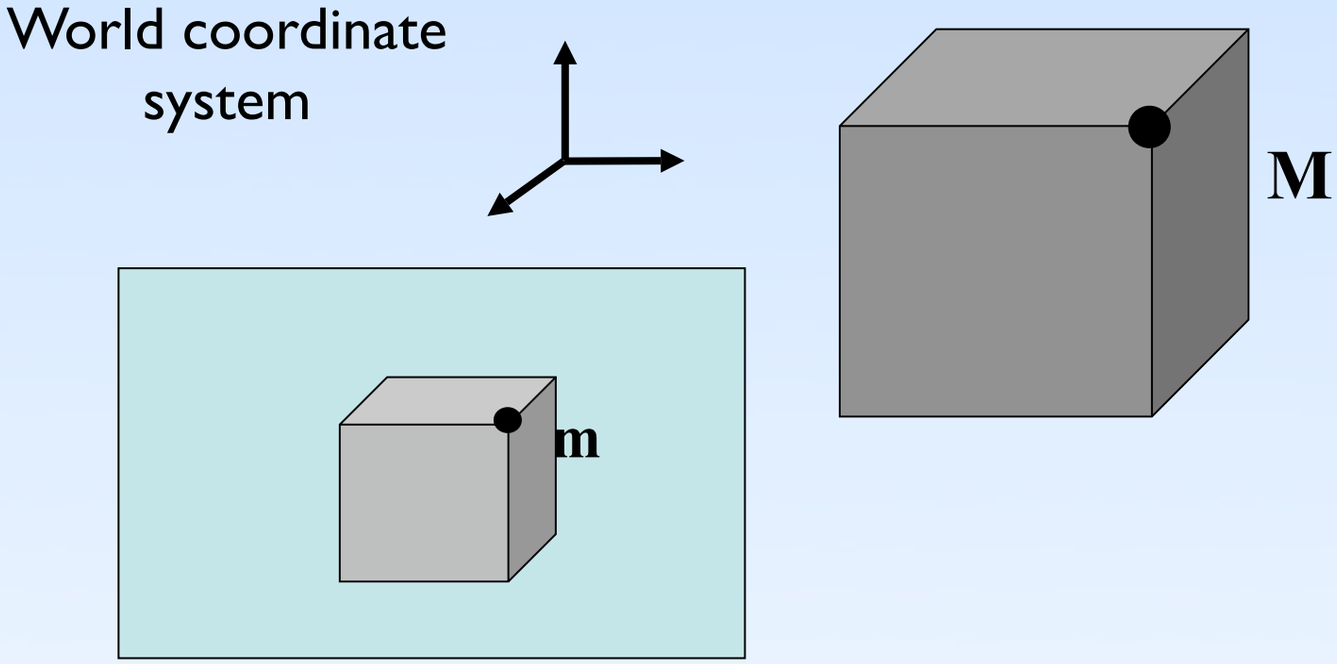
Drift: Error accumulation

Jittering: Inaccuracies

Camera Model

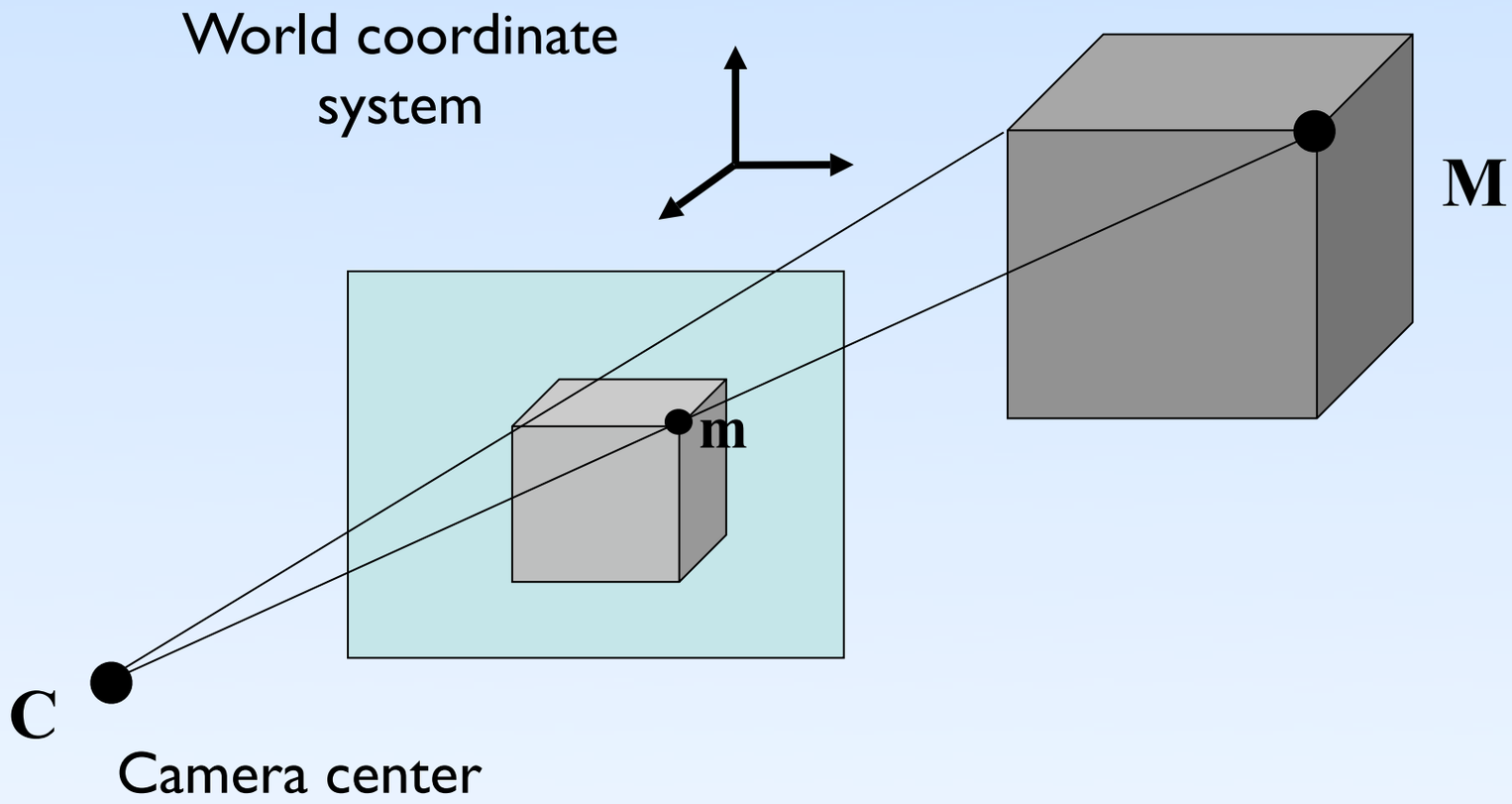
From World to Images

From the World to the Image



What is the relation between the 3D coordinates of a point **M** and its correspondent **m** in the image captured by the camera ?

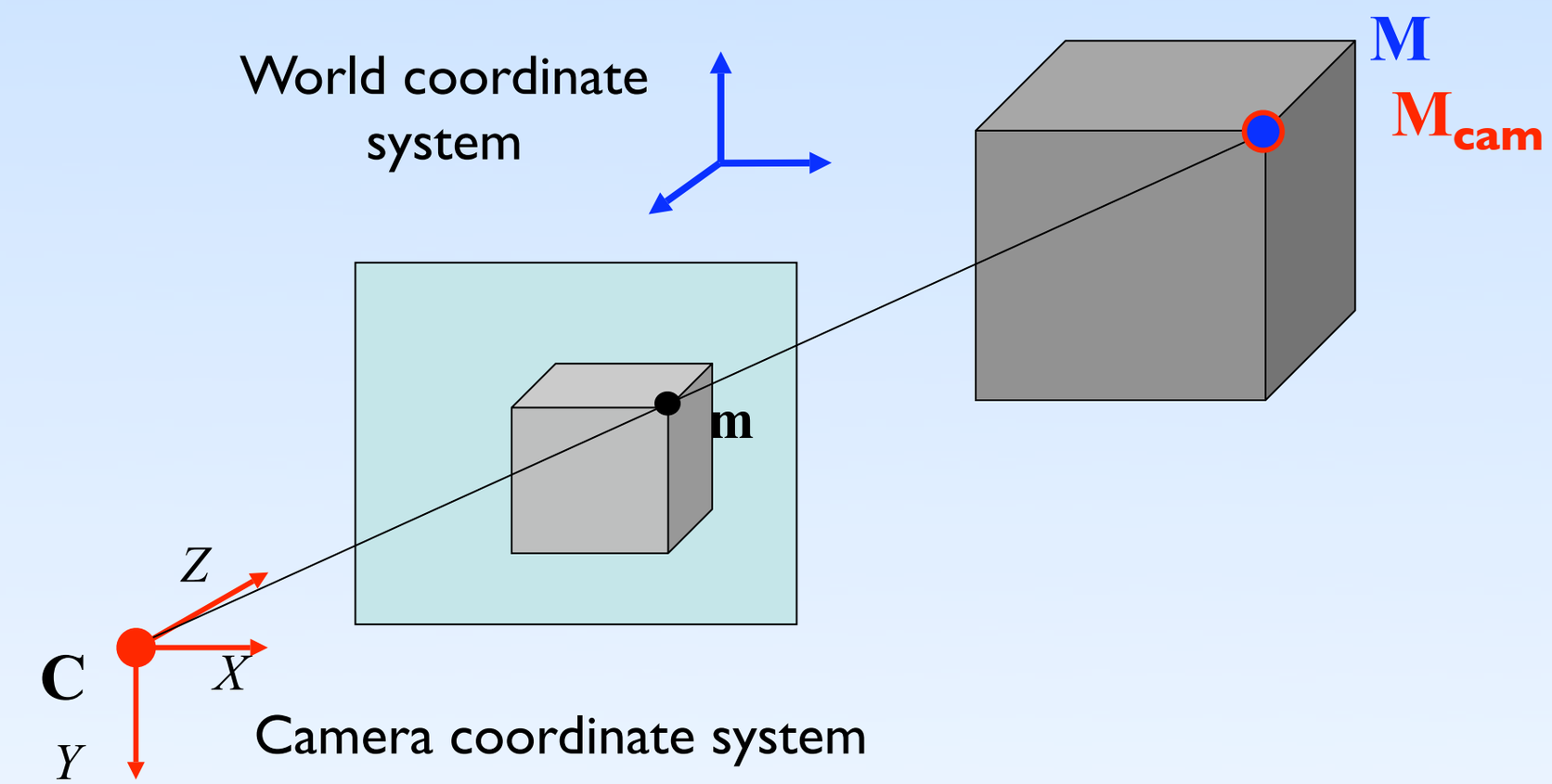
Perspective Projection



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

The rays passing through a 3D points **M** and their correspondents **m** in the image, intersect at a single point **C**, called the camera center.

Expressing **M** in the Camera Coordinates



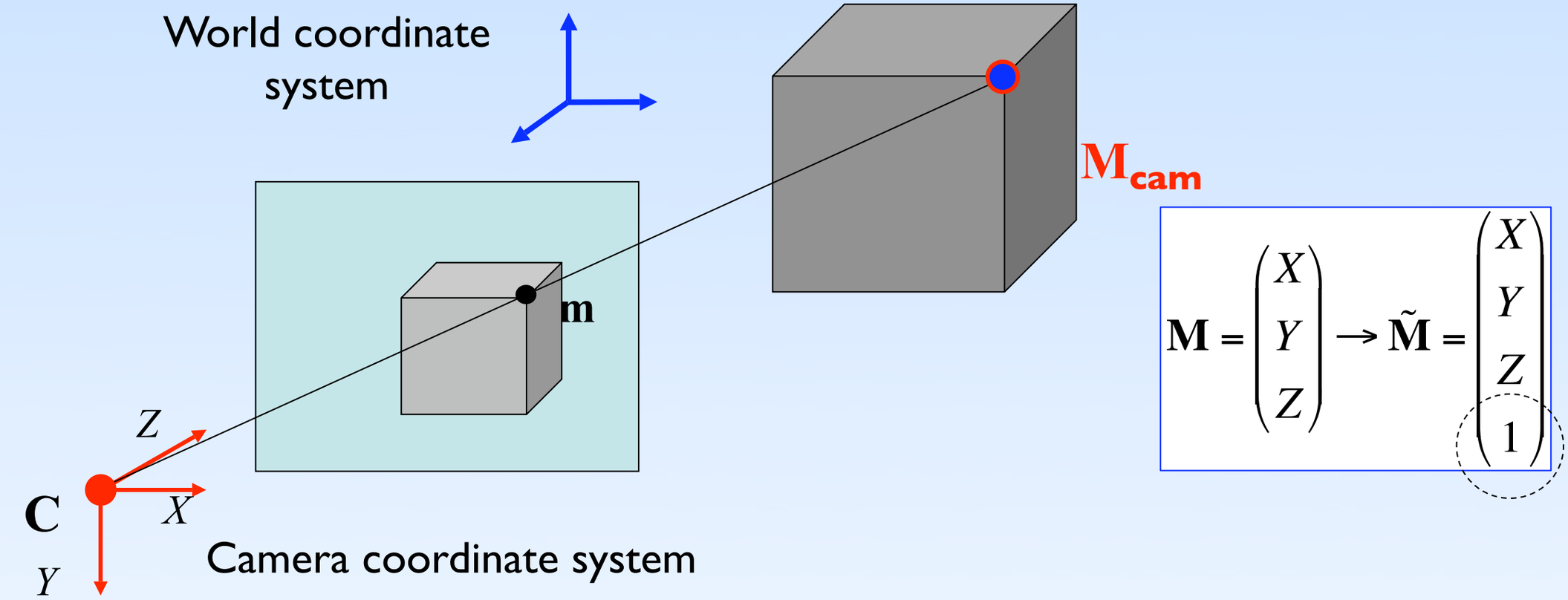
Step 1: Express the coordinates of **M** in the camera coordinates system as $\mathbf{M}_{cam}$.

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

$$\mathbf{M}_{cam} = \mathbf{RM} + \mathbf{T}$$

where: **R** is a 3x3 rotation matrix, and **T** is a 3- vector.

Homogeneous Coordinates



$$\mathbf{M} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \rightarrow \tilde{\mathbf{M}} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Lets replace $\mathbf{M}$ by the 4- homogeneous vector $\tilde{\mathbf{M}}$: Just add a 1 as the fourth coordinate.

Now, the Euclidean displacement can be expressed as an linear transformation instead of an affine:

$$\mathbf{M}_{\text{cam}} = \mathbf{R}\mathbf{M} + \mathbf{T} \rightarrow \begin{pmatrix} X_{\text{cam}} \\ Y_{\text{cam}} \\ Z_{\text{cam}} \end{pmatrix} = \mathbf{R}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{T} \rightarrow \begin{pmatrix} X_{\text{cam}} \\ Y_{\text{cam}} \\ Z_{\text{cam}} \end{pmatrix} = (\mathbf{R} \mid \mathbf{T})\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \mathbf{M}_{\text{cam}} = (\mathbf{R} \mid \mathbf{T})\tilde{\mathbf{M}}$$

$(\mathbf{R} \mid \mathbf{T})$ is a 3x4 matrix.

The External Calibration

The matrix

$$(\mathbf{R}\,|\,\mathbf{T}) = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{13} & \mathbf{R}_{13} & \mathbf{T}_1 \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \mathbf{R}_{23} & \mathbf{T}_2 \\ \mathbf{R}_{31} & \mathbf{R}_{32} & \mathbf{R}_{33} & \mathbf{T}_3 \end{pmatrix}$$

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 parameterization of rotations in the 3D space.

Homogeneous Coordinates

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

A 3- vector expressed in homogeneous coordinates becomes a 4-vector:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \rightarrow \begin{pmatrix} kX \\ kY \\ kZ \\ k \end{pmatrix}$$

To go back to the 3-vector, one just has to divide the 3 first coordinates by the last one.

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

All the homogeneous vectors of the form $\begin{pmatrix} kX \\ kY \\ kZ \\ k \end{pmatrix}$ correspond to the same $3-$vector $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$.

The homogeneous vector $\begin{pmatrix} u \\ v \\ w \\ t \end{pmatrix}$ corresponds to the $3-$vector $\begin{pmatrix} \frac{u}{t} \\ \frac{v}{t} \\ \frac{w}{t} \end{pmatrix}$.

Homogeneous

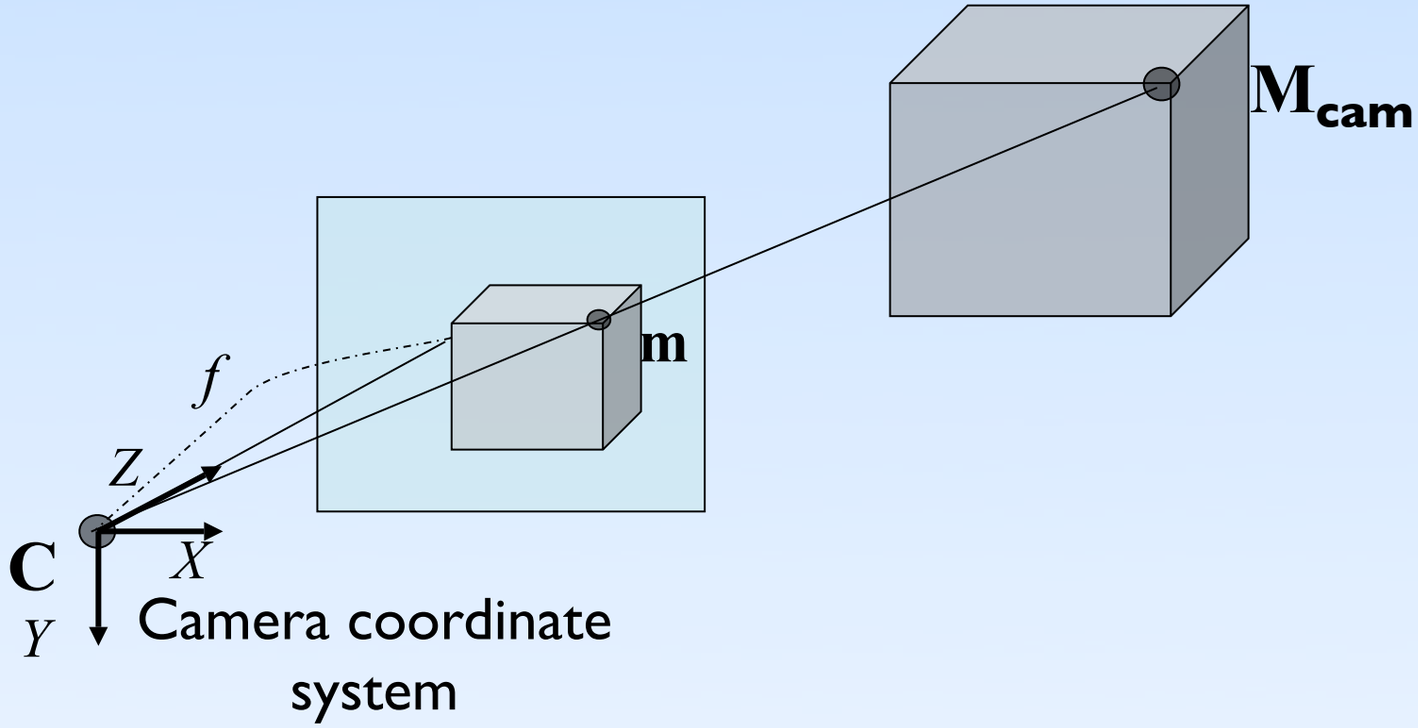
In homogeneous coordinates, an Euclidean displacement can be written using a 4x4 matrix:

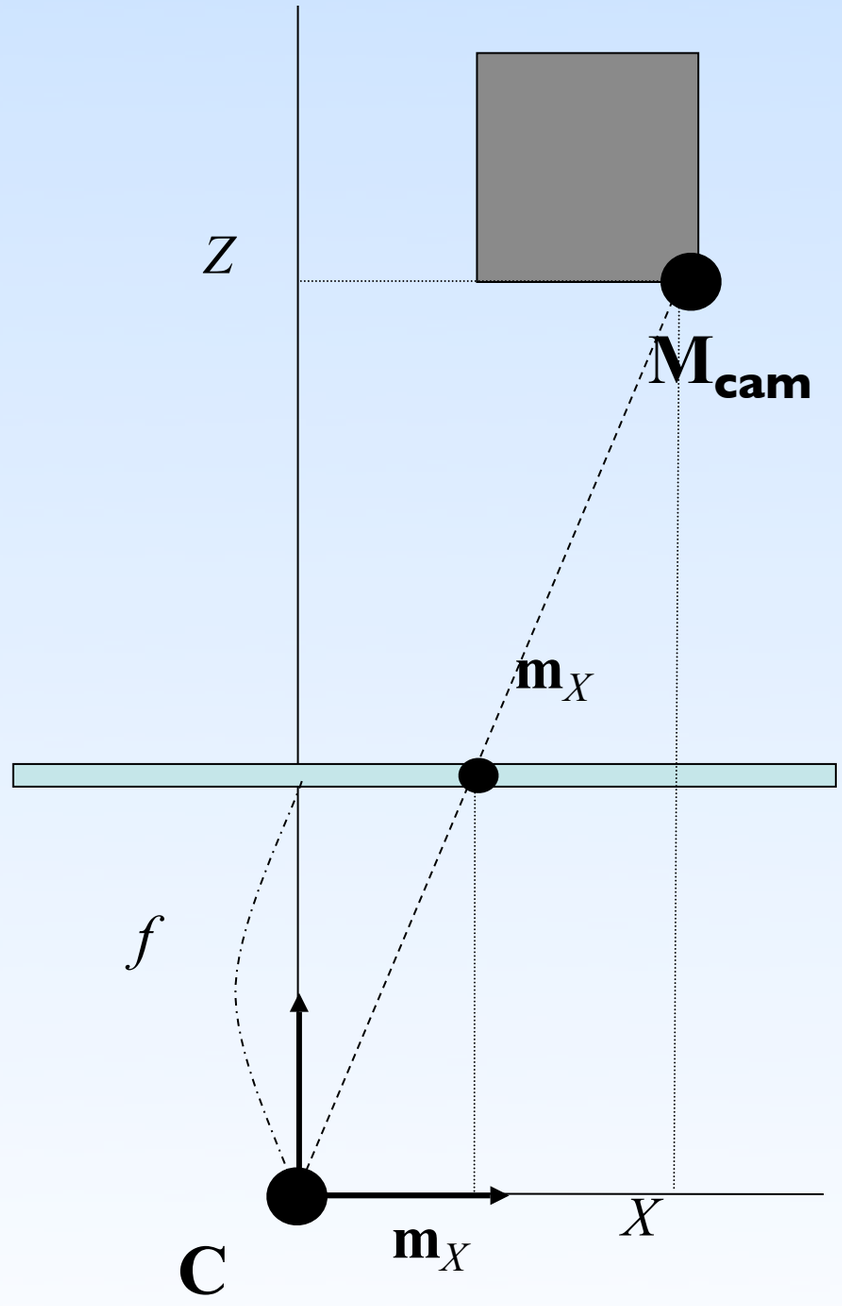
$$\begin{pmatrix} k'X' \\ k'Y' \\ k'Z' \\ k' \end{pmatrix} = \begin{pmatrix} & \mathbf{R} & & \mathbf{T} \\ & & & \\ & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} kX \\ kY \\ kZ \\ k \end{pmatrix}$$

The Computer Vision community uses a 3x4 matrix and considers that the 4th coordinate of $\tilde{\mathbf{M}}$ is 1:

$$\begin{pmatrix} X_{\text{cam}} \\ Y_{\text{cam}} \\ Z_{\text{cam}} \end{pmatrix} = (\mathbf{R} \mid \mathbf{T}) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Projection





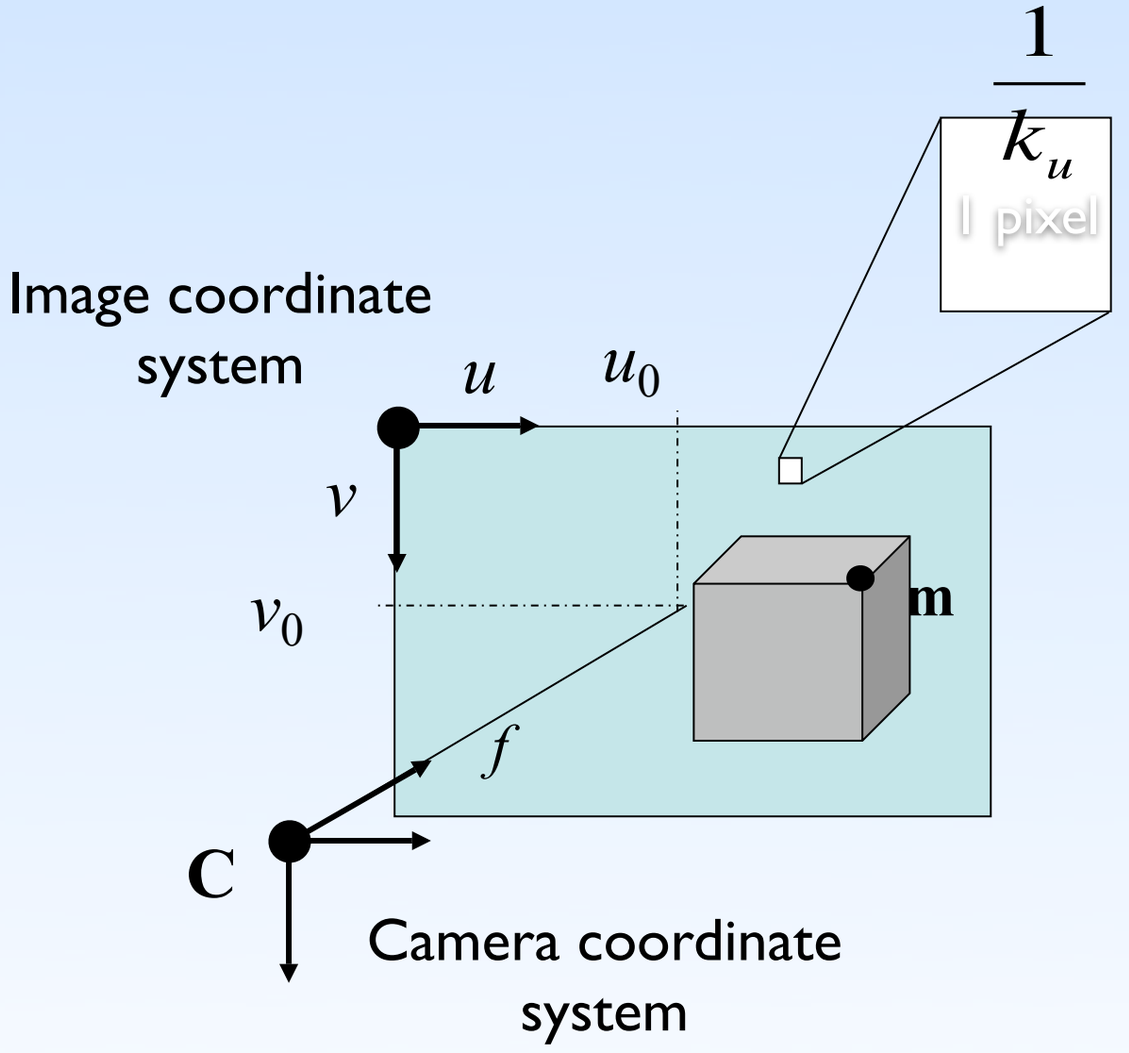
Computation of the coordinates of **m** in the image plane, from $\mathbf{M_{cam}}$ (expressed in the camera coordinates system):

Simply use Thales' theorem:

$$\frac{\mathbf{m}_X}{f} = \frac{X}{Z} \quad \rightarrow \quad \mathbf{m}_X = f\frac{X}{Z}$$

From Projection to Image

Coordinates of $\mathbf{m}$ in pixels ?



$$\mathbf{m}_X = f\frac{X}{Z}, \quad \mathbf{m}_Y = f\frac{Y}{Z}$$

$$\mathbf{m}_u = ?, \quad \mathbf{m}_v = ?$$

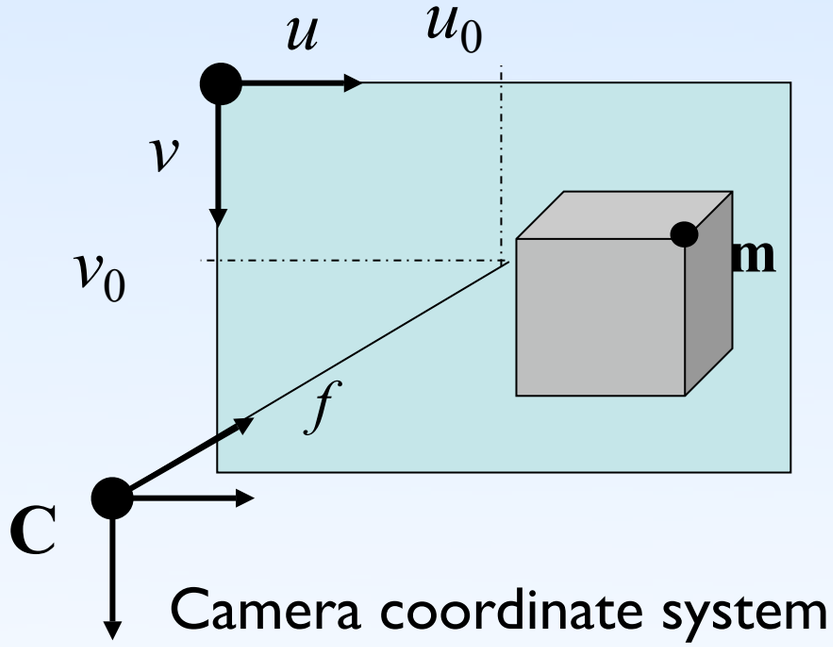
From Projection to Image

- (u_0, v_0): Projection of the camera center $\mathbf{C}$ in the image coordinate system.

- k_u : Inverse of the size of one pixel along X axis.

- k_v : Same for Y axis.

Image coordinate system



$$\mathbf{m}_X = f\frac{X}{Z}, \quad \mathbf{m}_Y = f\frac{Y}{Z}$$

$$\mathbf{m}_u = u_0 + k_u\mathbf{m}_X, \quad \mathbf{m}_v = v_0 + k_v\mathbf{m}_Y$$

From Projection to Image

Putting
- the perspective projection and
- the transformation into pixel coordinates

together:

$$\mathbf{m}_X = f\frac{X}{Z}, \quad \mathbf{m}_Y = f\frac{Y}{Z}$$

$$\mathbf{m}_u = u_0 + k_u\mathbf{m}_X, \quad \mathbf{m}_v = v_0 + k_v\mathbf{m}_Y$$

In matrix form:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$\begin{pmatrix} u \\ v \\ w \end{pmatrix}$ defines $\mathbf{m}$ in homogeneous coordinates $\rightarrow$

$$\begin{cases} \mathbf{m}_u = \dfrac{u}{w} = u_0 + k_u f\dfrac{X}{Z} \\ \mathbf{m}_v = \dfrac{v}{w} = v_0 + k_v f\dfrac{Y}{Z} \end{cases}$$

The Internal Calibration Matrix

The 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 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 can be reduced under some assumptions:

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

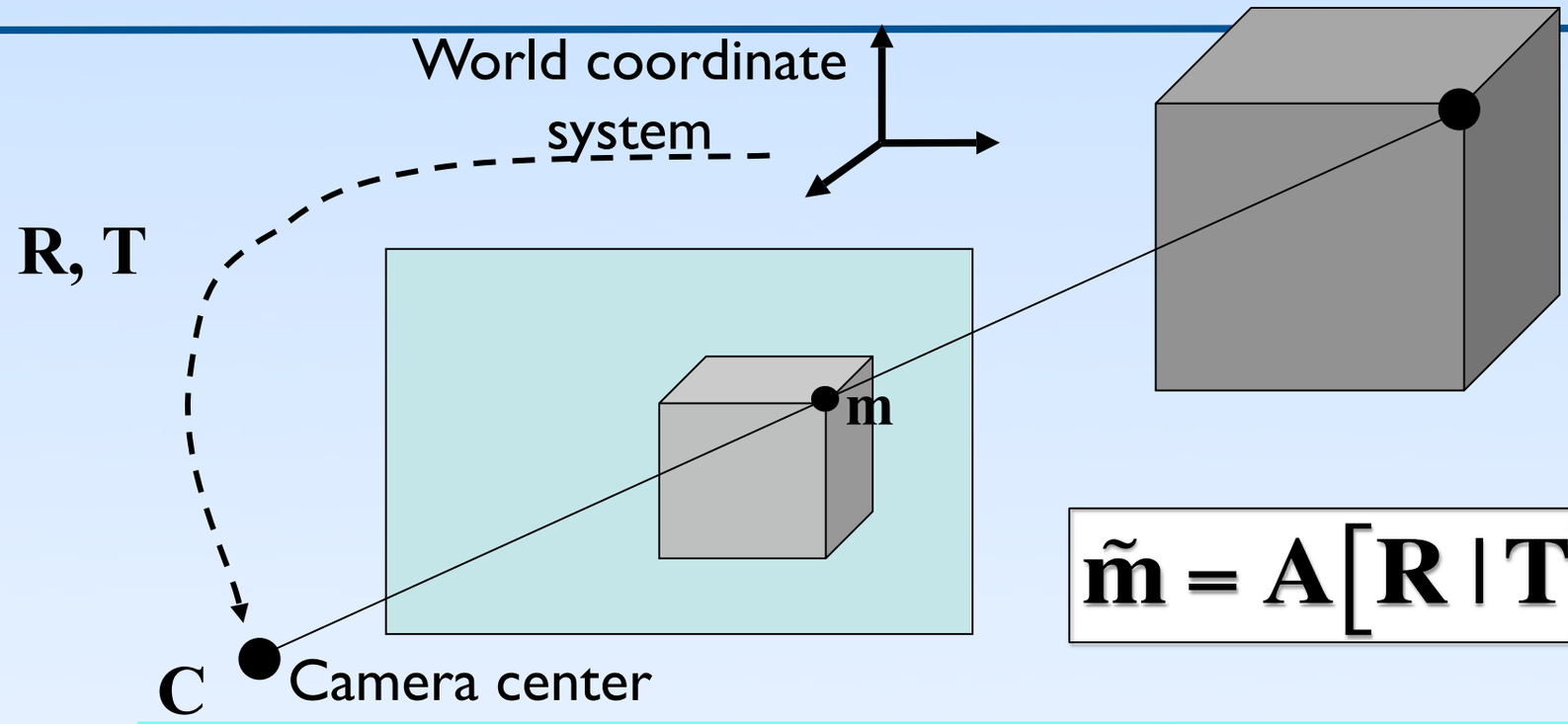
The Full Transformation

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

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{13} & \mathbf{R}_{13} & \mathbf{T}_1 \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \mathbf{R}_{23} & \mathbf{T}_2 \\ \mathbf{R}_{31} & \mathbf{R}_{32} & \mathbf{R}_{33} & \mathbf{T}_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix}}_{\text{projection matrix}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Notations



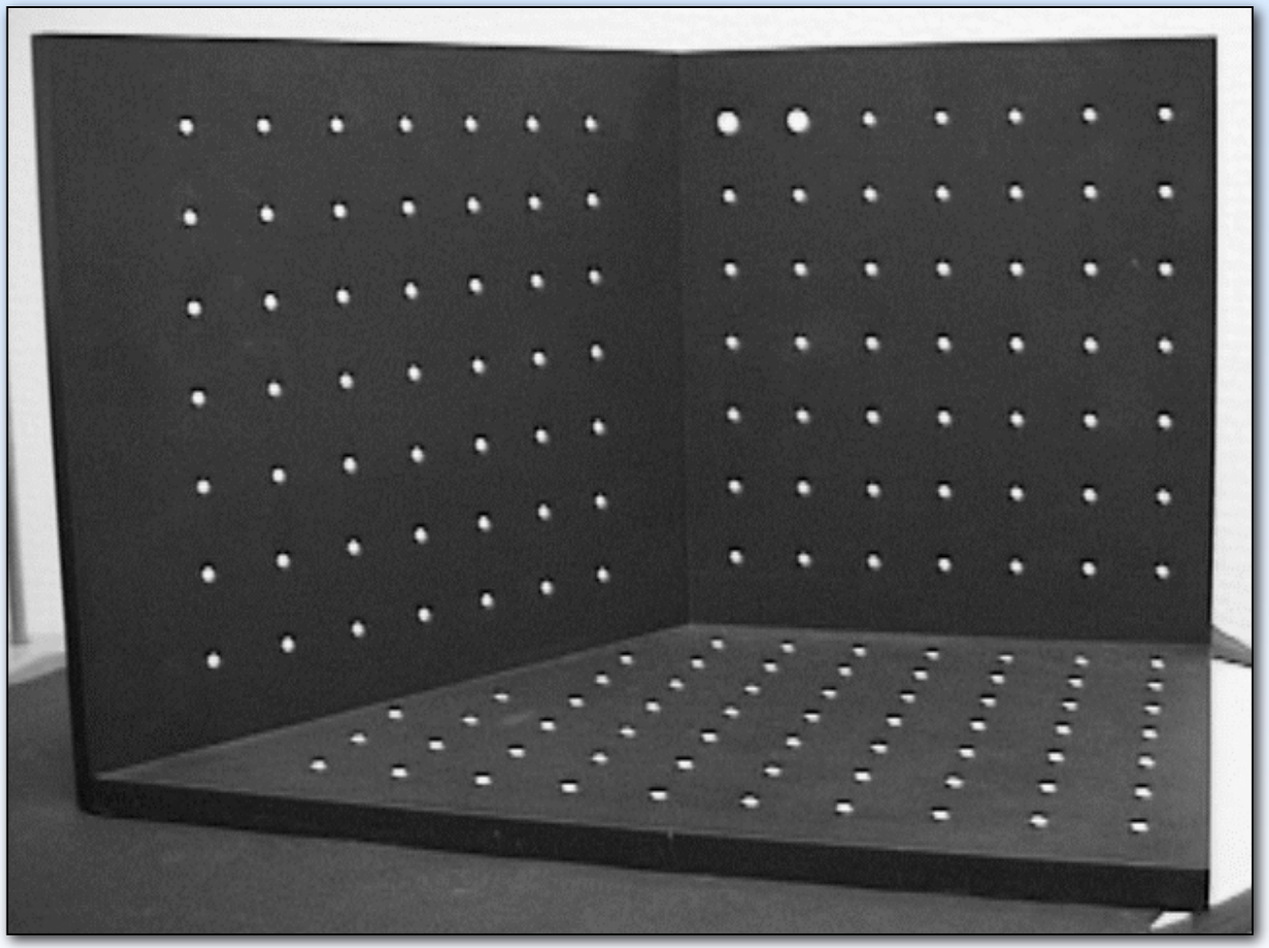
$$\tilde{\mathbf{m}} = \mathbf{A}\left[\mathbf{R} \mid \mathbf{T}\right]\tilde{\mathbf{M}} = \mathbf{P}\tilde{\mathbf{M}}$$

- $\mathbf{A}$: Internal calibration matrix (3x3);
- $\mathbf{R}$: Rotation matrix (3x3);
- $\mathbf{T}$: Translation vector (3x1);
- $\mathbf{P}$: Projection matrix (3x4);
- $\tilde{\mathbf{M}}$: world 3D point in homogenous coordinates (4x1);
- $\tilde{\mathbf{m}}$: image 2D point in homogenous coordinates (3x1), projection of $\tilde{\mathbf{M}}$ in the image;

Internal Parameters (A)

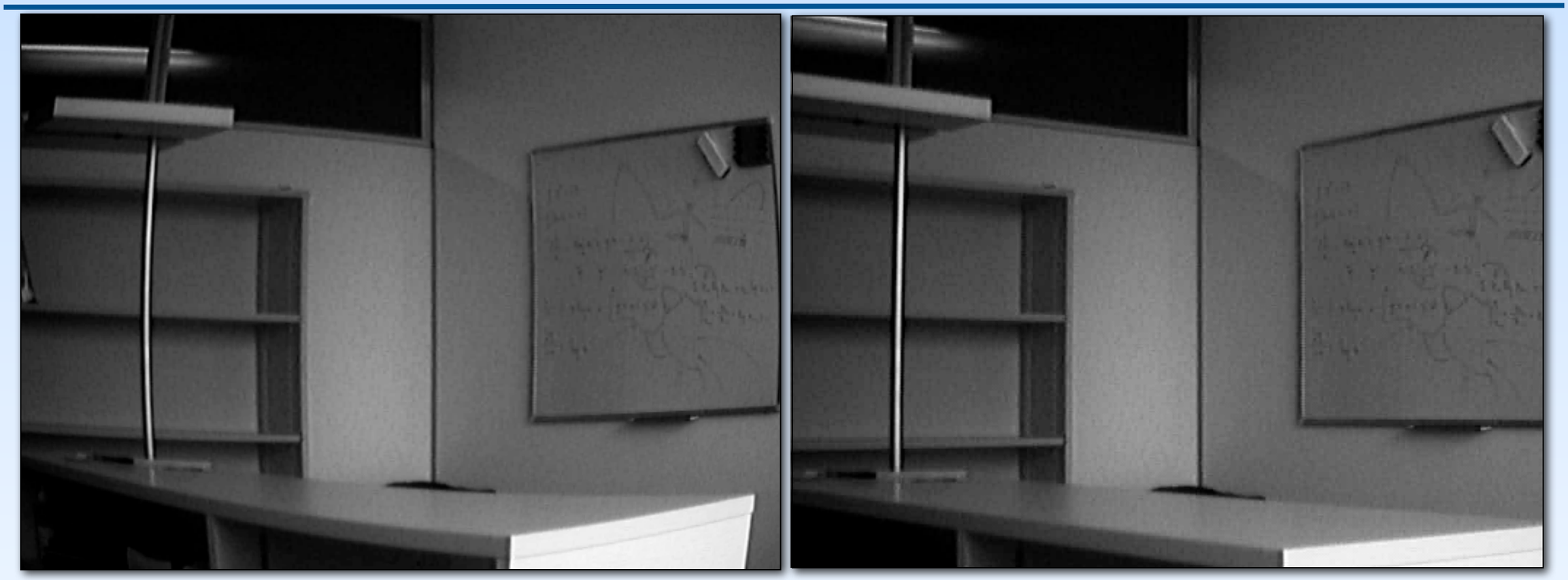
A : Often assumed to be known in the following.

Can be estimated beforehand using a calibration grid.



$$\begin{cases} \cdots \\ \mathbf{m}_i = \mathbf{A}[\mathbf{R} \mid \mathbf{T}]\mathbf{M}_i \\ \cdots \end{cases}$$

Camera Distortion

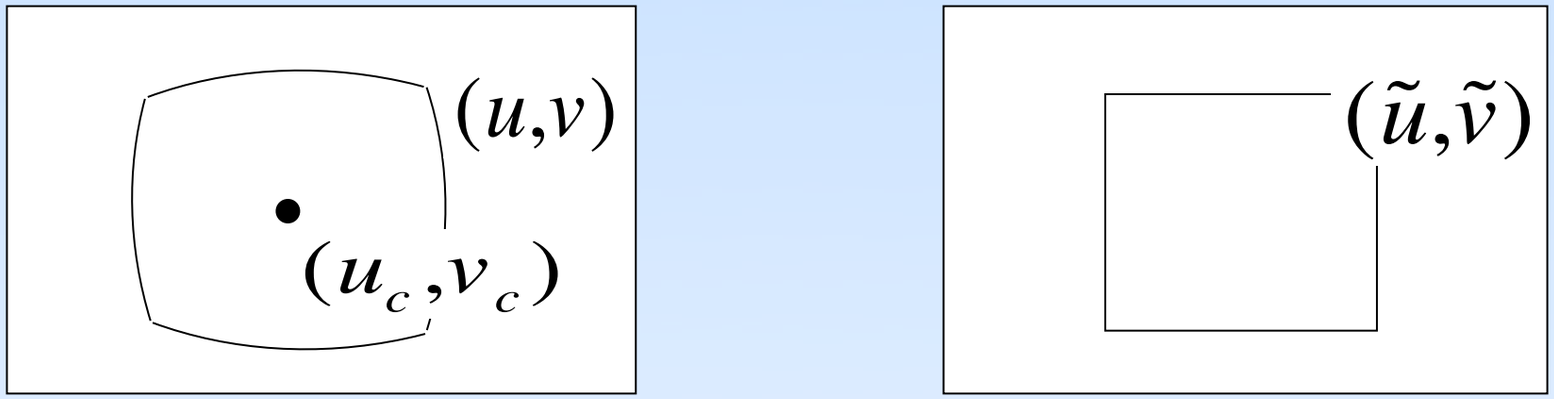


•The previous camera model does not take into account distortions due to fish eye lenses or bad quality lenses.

• Fish eye lenses are interesting because images have a larger field of view.

•Distortion can be removed before applying perspective projection

Distortion Estimation



Radial (important) + tangential distortions (has much less influence, so it is usually neglected).

(u_c,v_c) : Center of radial distortion.

The correction is written:

$$\begin{cases}\tilde{u} = u_c + L(r)\left(u - u_c\right)\\ \tilde{v} = v_c + L(r)\left(v - v_c\right)\end{cases} \quad \text{with} \quad r^2 = \left(u - u_c\right)^2 + \left(v - v_c\right)^2 \text{ and } L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \ldots$$

Once the distortion parameters are estimated, the distortion can be compensated in two ways:

- Undistort the images, using a look-up table for efficiency;
- "Distort" the model projection when computing a re-projection error.

Camera Pose (Rotation + translation)

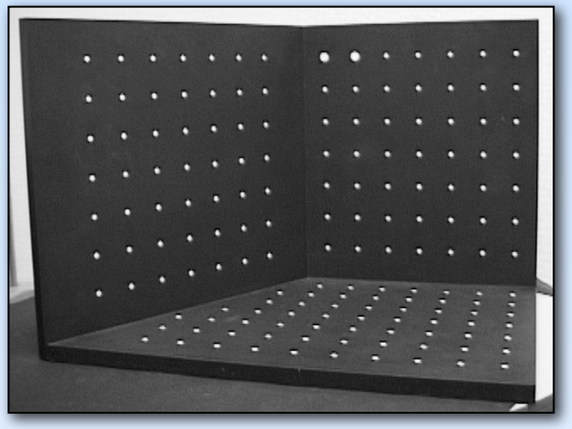
From 3D-2D points correspondences:

- DLT algorithm + Extraction of **A**, **R**, **t**;
- P3P algorithm;
- POSIT and others;
- Non-linear minimization.

From a planar structure.

Linear Estimation of P from 3D/2D point correspondences (DLT algorithm)

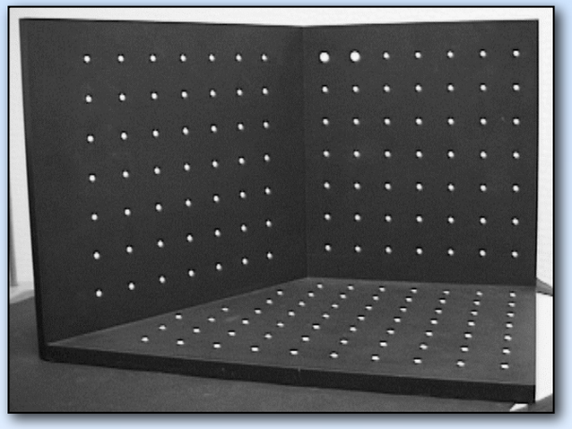
Each correspondence gives us 2 equations:



Stacking all the equations into matrix $\mathbf{B}$ yields the linear system:

Linear Estimation of P from 3D/2D point correspondences (DLT algorithm)

Each correspondence gives us 2 equations:

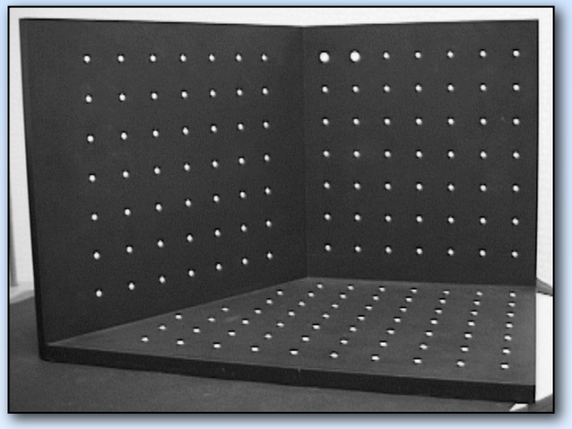


$$\tilde{\mathbf{m}}_i = \mathbf{P}\tilde{\mathbf{M}}_i$$

Stacking all the equations into matrix $\mathbf{B}$ yields the linear system:

Linear Estimation of P from 3D/2D point correspondences (DLT algorithm)

Each correspondence gives us 2 equations:



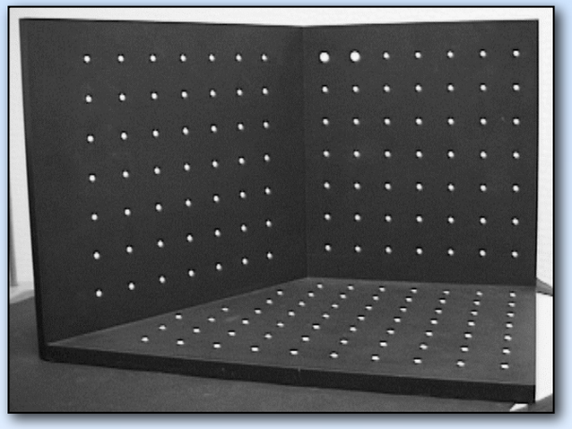
$$\tilde{\mathbf{m}}_i = \mathbf{P}\tilde{\mathbf{M}}_i$$

$$\begin{cases} u_i = \dfrac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \\ v_i = \dfrac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \end{cases}$$

Stacking all the equations into matrix $\mathbf{B}$ yields the linear system:

Linear Estimation of P from 3D/2D point correspondences (DLT algorithm)

Each correspondence gives us 2 equations:



$$\tilde{\mathbf{m}}_i = \mathbf{P}\tilde{\mathbf{M}}_i$$

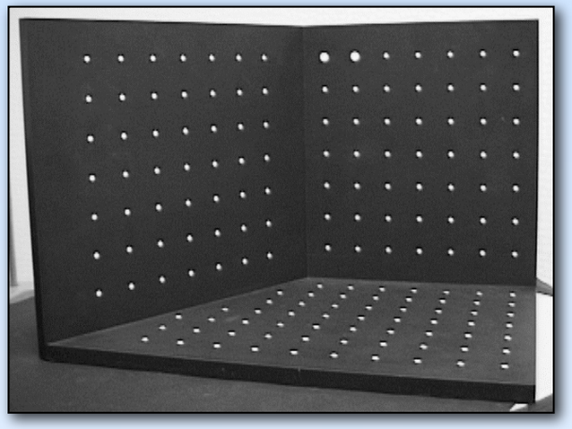
$$\begin{cases} u_i = \dfrac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \\ v_i = \dfrac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \end{cases}$$

$$\begin{cases} P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14} - P_{31}X_i u_i - P_{32}Y_i u_i - P_{33}Z_i u_i - P_{34}u_i = 0 \\ P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24} - P_{31}X_i v_i - P_{32}Y_i v_i - P_{33}Z_i v_i - P_{34}v_i = 0 \end{cases}$$

Stacking all the equations into matrix **B** yields the linear system:

Linear Estimation of P from 3D/2D point correspondences (DLT algorithm)

Each correspondence gives us 2 equations:



$$\tilde{\mathbf{m}}_i = \mathbf{P}\tilde{\mathbf{M}}_i$$

$$\begin{cases} u_i = \dfrac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \\ v_i = \dfrac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \end{cases}$$

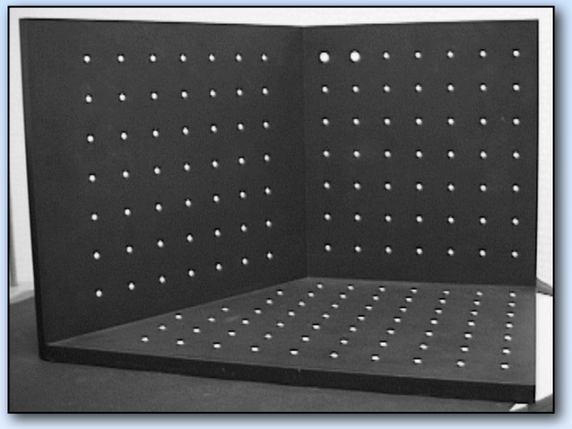
$$\begin{cases} P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14} - P_{31}X_i u_i - P_{32}Y_i u_i - P_{33}Z_i u_i - P_{34}u_i = 0 \\ P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24} - P_{31}X_i v_i - P_{32}Y_i v_i - P_{33}Z_i v_i - P_{34}v_i = 0 \end{cases}$$

$$\begin{pmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -X_i u_i & -Y_i u_i & -Z_i u_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -X_i v_i & -Y_i v_i & -Z_i v_i & -v_i \end{pmatrix} \begin{pmatrix} P_{11} \\ \dots \\ P_{34} \end{pmatrix} = \mathbf{0}$$

Stacking all the equations into matrix **B** yields the linear system:

Linear Estimation of P from 3D/2D point correspondences (DLT algorithm)

Each correspondence gives us 2 equations:



$$\tilde{\mathbf{m}}_i = \mathbf{P}\tilde{\mathbf{M}}_i$$

$$\begin{cases} u_i = \dfrac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \\ v_i = \dfrac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \end{cases}$$

$$\begin{cases} P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14} - P_{31}X_iu_i - P_{32}Y_iu_i - P_{33}Z_iu_i - P_{34}u_i = 0 \\ P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24} - P_{31}X_iv_i - P_{32}Y_iv_i - P_{33}Z_iv_i - P_{34}v_i = 0 \end{cases}$$

$$\begin{pmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -X_iu_i & -Y_iu_i & -Z_iu_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -X_iv_i & -Y_iv_i & -Z_iv_i & -v_i \end{pmatrix} \begin{pmatrix} P_{11} \\ \dots \\ P_{34} \end{pmatrix} = \mathbf{0}$$

Stacking all the equations into matrix **B** yields the linear system:

$$\mathbf{B}\begin{pmatrix} P_{11} \\ \dots \\ P_{34} \end{pmatrix} = \mathbf{0}$$

DLT algorithm

$$\mathbf{B}\begin{pmatrix} P_{11} \\ \dots \\ P_{34} \end{pmatrix} = \mathbf{0} \qquad \Rightarrow \begin{pmatrix} P_{11} \\ \dots \\ P_{34} \end{pmatrix}$$

is the eigen vector of **B** corresponding to the smallest eigenvalue of **B**.

Question: Why is the null vector not a valid solution ?

11 coefficients to estimate; each 3D-2D correspondence gives 2 equations: 6 3D-2D correspondences must be known.

In practice, much more are needed to get a good estimation.

Extraction of **A**, **R**, and **T** from **P**

1. **A** is not known. -- DLT algorithm
2. **A** is known. -- PnP algorithm

Trick:

Consider the matrix made of the first 3 columns of **P**:

$$\mathbf{P} = (\mathbf{P}_3 \,|\, \mathbf{c}_4) \quad \mathbf{P}_3 = \mathbf{AR}$$

And compute

$$\mathbf{P}_3\mathbf{P}_3^T \qquad \mathbf{P}_3\mathbf{P}_3^T = (\mathbf{AR})(\mathbf{AR})^T = \mathbf{ARR}^T\mathbf{A}^T = \mathbf{AA}^T$$

→ since **A** is a upper-triangular matrix, it can be found using Choleski factorization of $\mathbf{P}_3\mathbf{P}_3^T$

Extraction of **A**, **R**, and **T** from **P**

1. **A** is not known. -- DLT algorithm
2. **A** is known. -- PnP algorithm

Trick:

Consider the matrix made of the first 3 columns of **P**:

$$\mathbf{P} = (\mathbf{P}_3 \,|\, \mathbf{c}_4) \quad \mathbf{P}_3 = \mathbf{AR}$$

And compute

$$\mathbf{P}_3\mathbf{P}_3^T \qquad \mathbf{P}_3\mathbf{P}_3^T = (\mathbf{AR})(\mathbf{AR})^T = \mathbf{ARR}^T\mathbf{A}^T = \mathbf{AA}^T$$

→ since **A** is a upper-triangular matrix, it can be found using Choleski factorization of $\mathbf{P}_3\mathbf{P}_3^T$

$\left(\mathbf{AA}^T\right)^{-1}$ is called the *image of the absolute conic*.

Extraction of $\mathbf{R}$, and $\mathbf{T}$

Once $\mathbf{A}$ is known:

$$[\mathbf{R}|\mathbf{t}] \propto \mathbf{A}^{-1}\mathbf{P}$$

No guarantee that $\mathbf{R}$ is a rotation matrix:

$\mathbf{R}$ still must be "ortho-normalized".

Ortho-Normalization of $\mathbf{R}$

Lets $\mathbf{Q} = \mathbf{A}^{-1}\mathbf{P}_3$ a first approximation of $\mathbf{R}$.

No guarantee that $\mathbf{Q}$ is a rotation matrix.

How to find the "best" rotation matrix that approximates $\mathbf{Q}$?

1. Singular Value Decomposition of $\mathbf{Q}$: $\mathbf{Q} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, where

   $\mathbf{S} = diag(s_1, s_2, s_3)$

   $\mathbf{U}$ and $\mathbf{V}$ have orthogonal columns, so that

   $$\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$$

2. $\mathbf{R} = \mathbf{U}\mathbf{V}^T$

[From Zhang TR98]

Ortho-Normalization of $\mathbf{R}$: Proof

Lets $\mathbf{Q} = \mathbf{A}^{-1}\mathbf{P}_3$ a first approximation of $\mathbf{R}$. No guarantee that $\mathbf{Q}$ is a rotation matrix.

How to find the "best" rotation matrix that approximates $\mathbf{Q}$?

- Take "best" in the sense of the smallest Frobenius norm of the difference $\mathbf{R} - \mathbf{Q}$.

$$\min_{\mathbf{R}} \|\mathbf{R} - \mathbf{Q}\|_F^2 \quad \text{subject to } \mathbf{R}\mathbf{R}^T = \mathbf{I}$$

$$\|\mathbf{R} - \mathbf{Q}\|_F^2 = trace\left((\mathbf{R} - \mathbf{Q})^T(\mathbf{R} - \mathbf{Q})\right) \qquad \text{(definition of Frobenius norm)}$$

$$= trace(\mathbf{R}^T\mathbf{R} - \mathbf{Q}^T\mathbf{R} - \mathbf{R}^T\mathbf{Q} + \mathbf{Q}^T\mathbf{Q})$$

$$= 3 + trace\left(\mathbf{Q}^T\mathbf{Q}\right) - 2trace\left(\mathbf{R}^T\mathbf{Q}\right)$$

Minimizing $||\mathbf{R} - \mathbf{Q}||^2$ is equivalent to maximize $trace(\mathbf{R}^T\mathbf{Q})$)

Let the SVD of $\mathbf{Q}$ be $\mathbf{U}\mathbf{S}\mathbf{V}^T$, where $\mathbf{S} = diag(s_1, s_2, s_3)$:

$$trace(\mathbf{R}^T\mathbf{Q}) = trace(\mathbf{R}^T\mathbf{U}\mathbf{S}\mathbf{V}^T) = trace(\mathbf{V}^T\mathbf{R}^T\mathbf{U}\mathbf{S})$$

$$trace(\mathbf{Z}\mathbf{S}) = \sum_i z_i i s_i \leq \sum_i si$$

with $\mathbf{Z} = \mathbf{V}^T\mathbf{R}^T\mathbf{U}$. Maximum reached with $\mathbf{Z} = \mathbf{I} \Rightarrow \mathbf{R} = \mathbf{U}\mathbf{V}^T$

DLT : Summary

Not a good idea
when the internal parameters are known.

P-*n*-P Problem

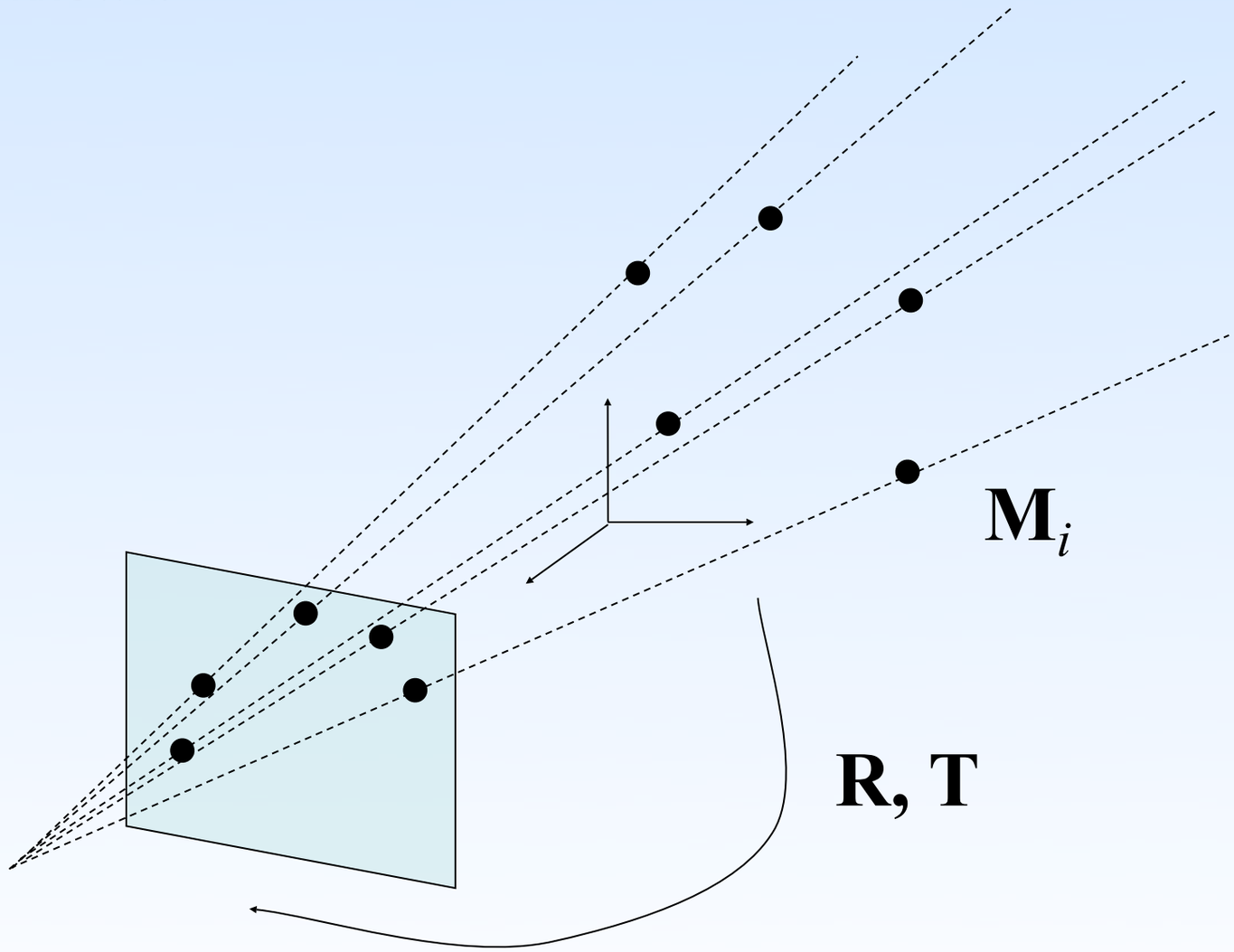
P*n*P: Perspective-*n*-Point

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

P-*n*-P Problem

P*n*P: Perspective-*n*-Point

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



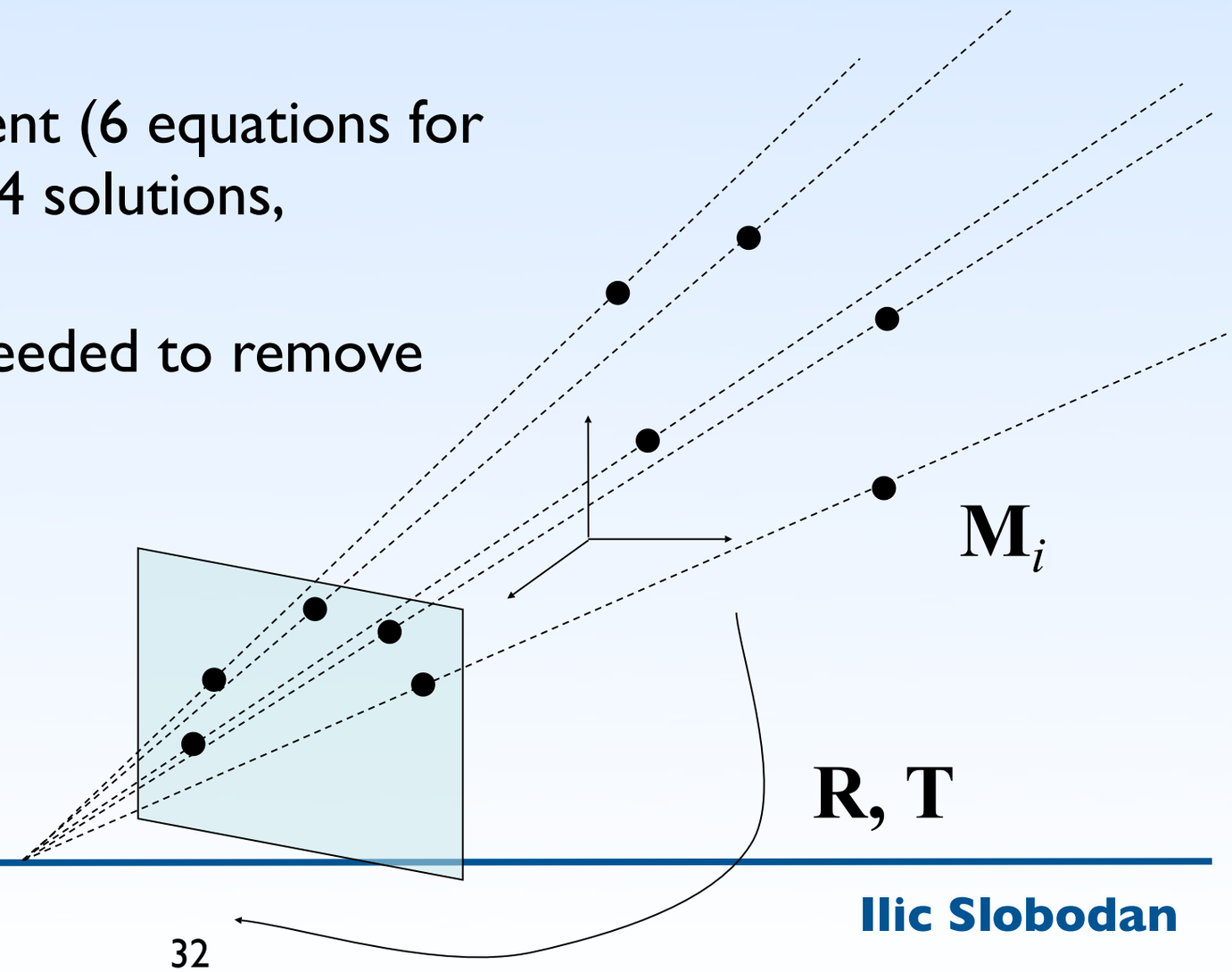
P-*n*-P Problem

P*n*P: Perspective-*n*-Point

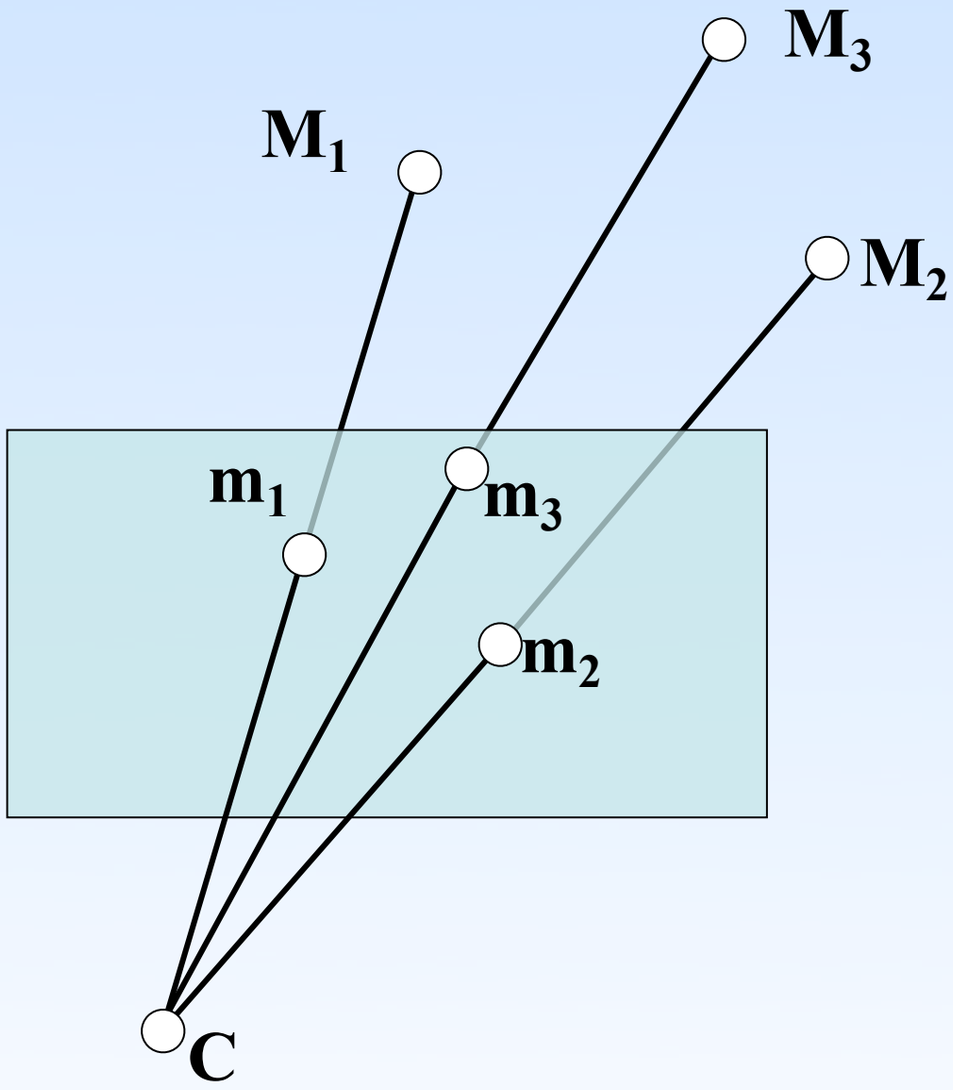
Estimation of the Position and Orientation (**R** and **T**) from 2D/3D point correspondences when the internal parameters (**A**) are known.

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

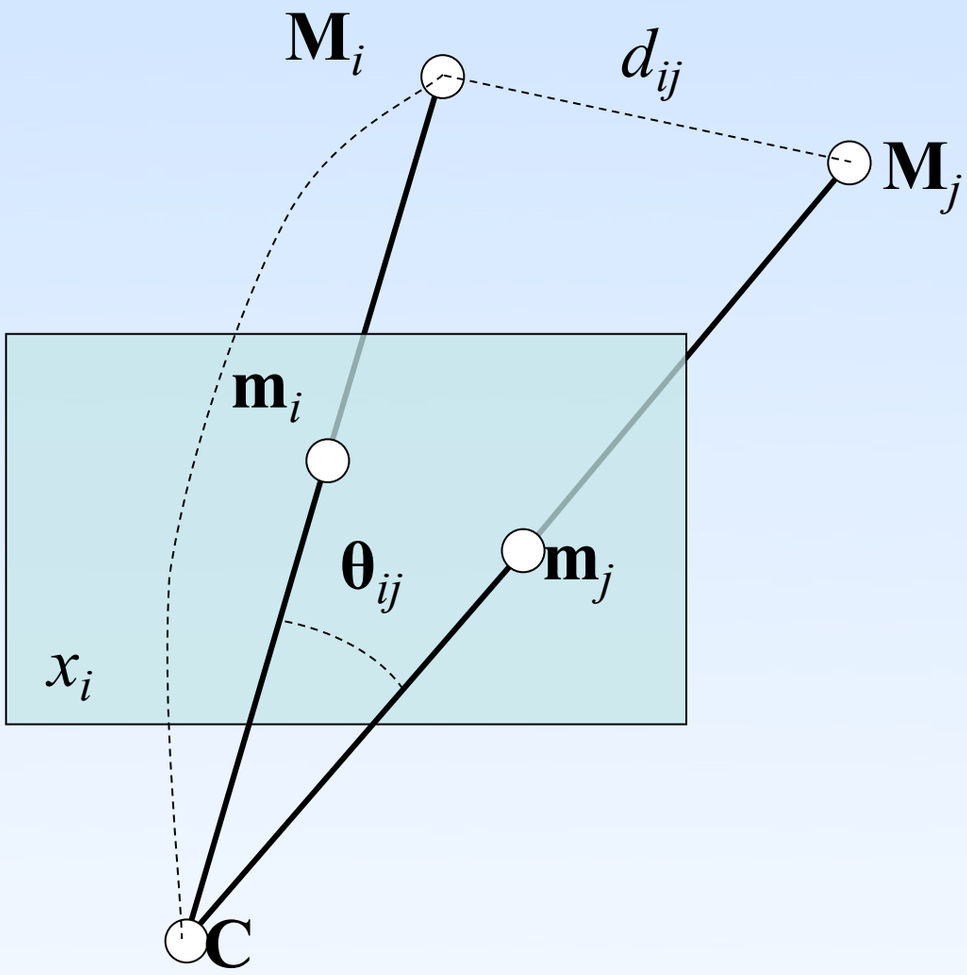


P3P: Pose estimation from 3 correspondences.

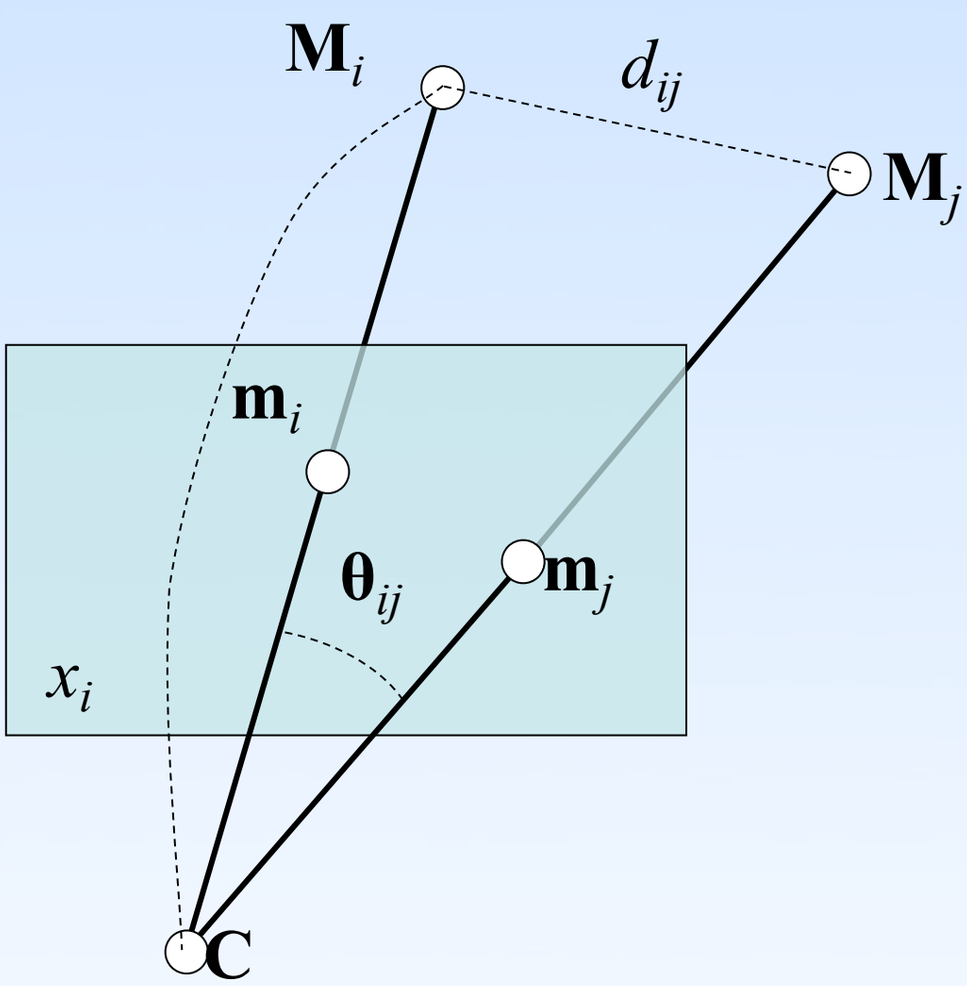
Yield up to 4 solutions, generally only two.

A fourth correspondence will be needed to remove the ambiguity.

The P3P Problem

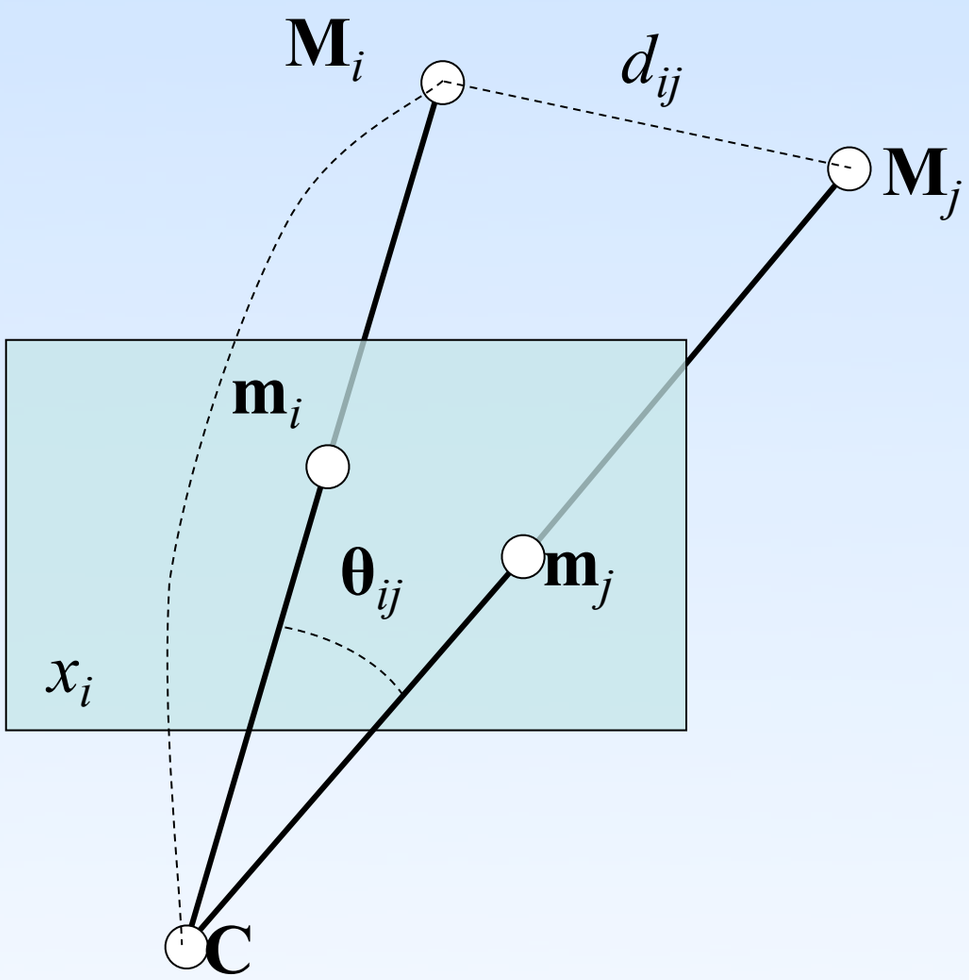


The P3P Problem



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

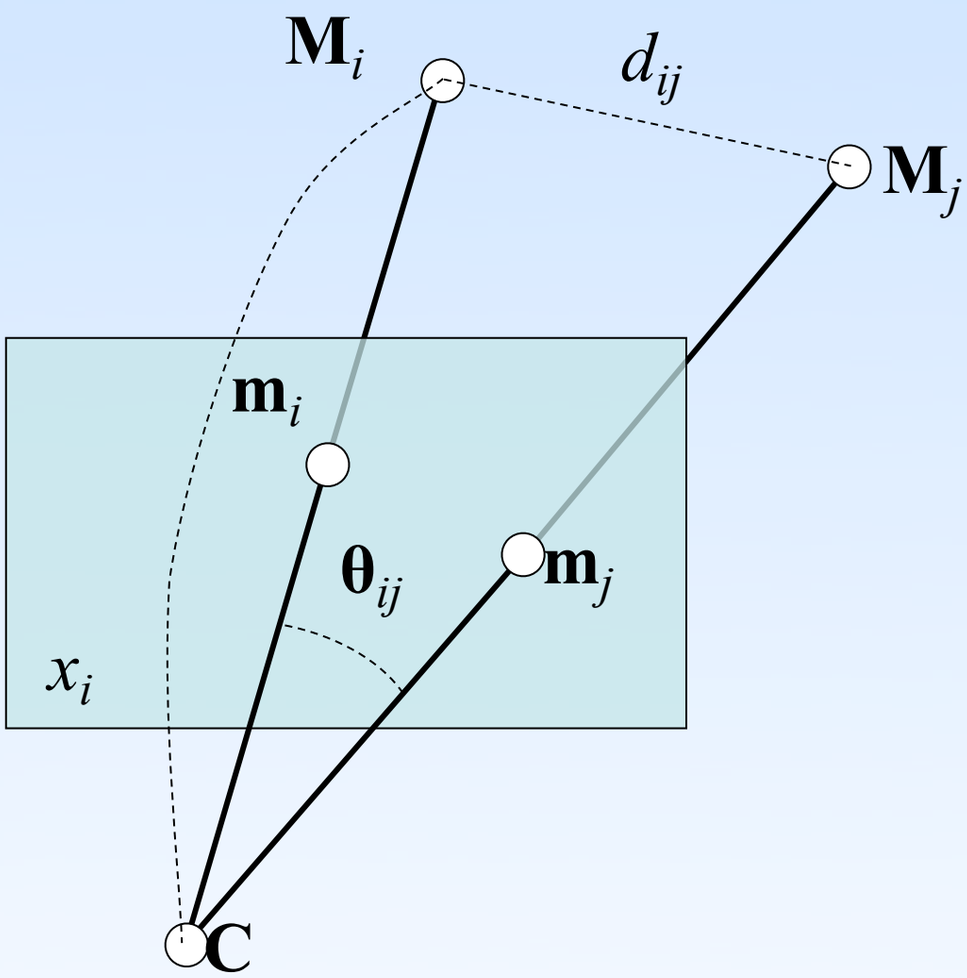
The P3P Problem



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 $x_j = ||\mathbf{M}_j - \mathbf{C}||$:

The P3P Problem

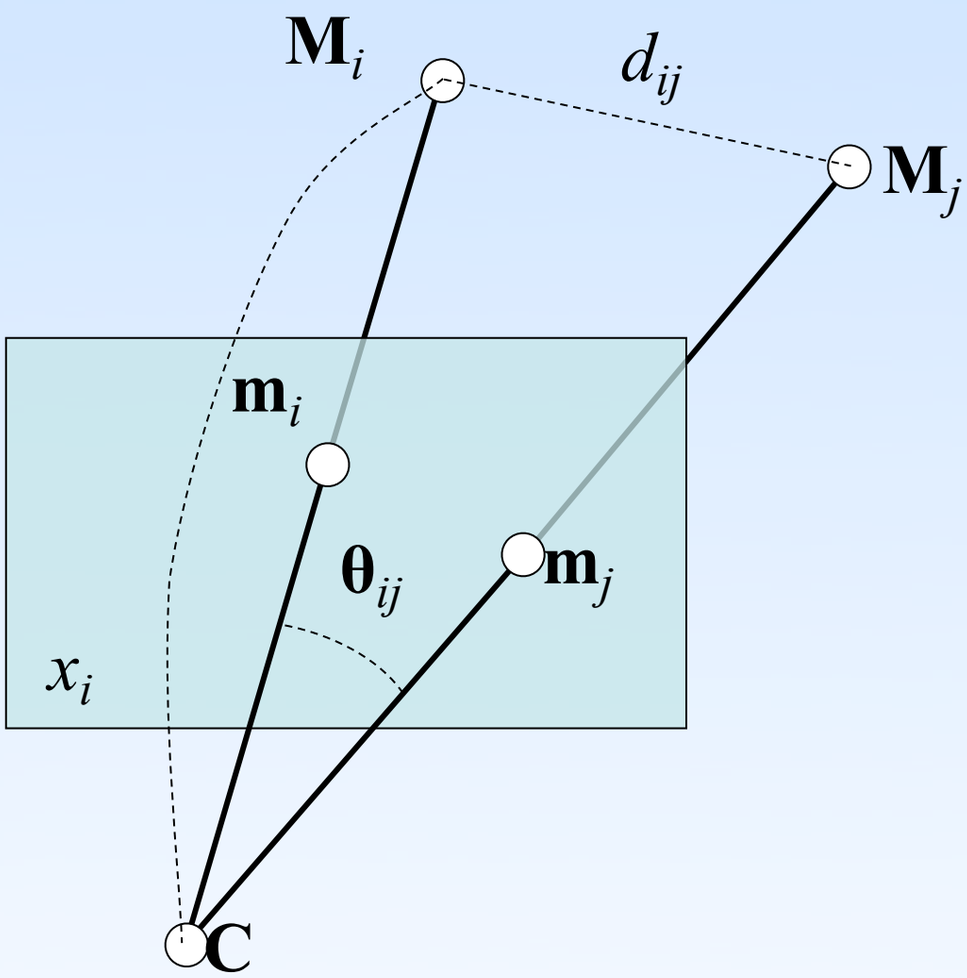


Each pair of correspondences $\mathbf{M}_i \leftrightarrow \mathbf{m}_i$ and $\mathbf{M}_j \leftrightarrow \mathbf{m}_j$ gives a constraint on the (unknown) camera-point distances

$x_i = ||\mathbf{M}_i - \mathbf{C}||$ and $x_j = ||\mathbf{M}_j - \mathbf{C}||$:

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

The P3P Problem



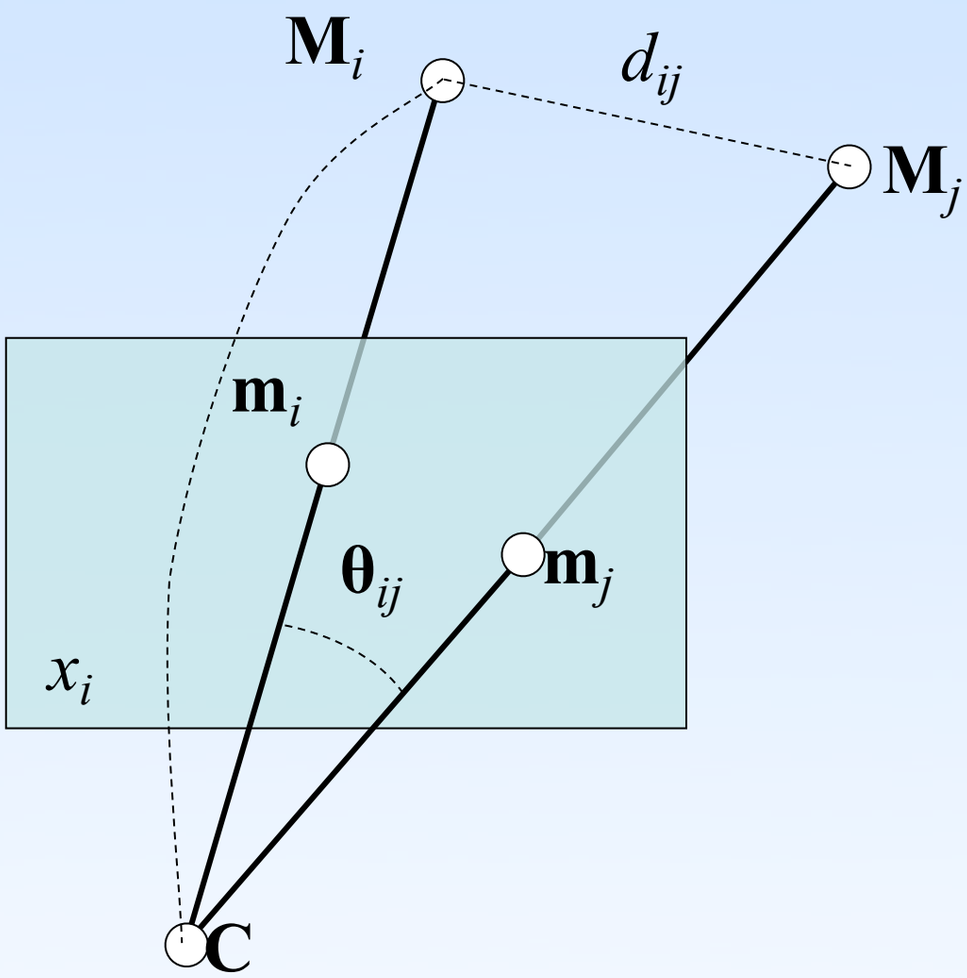
Each pair of correspondences $\mathbf{M}_i \leftrightarrow \mathbf{m}_i$ and $\mathbf{M}_j \leftrightarrow \mathbf{m}_j$ gives a constraint on the (unknown) camera-point distances

$x_i = ||\mathbf{M}_i - \mathbf{C}||$ 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:

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

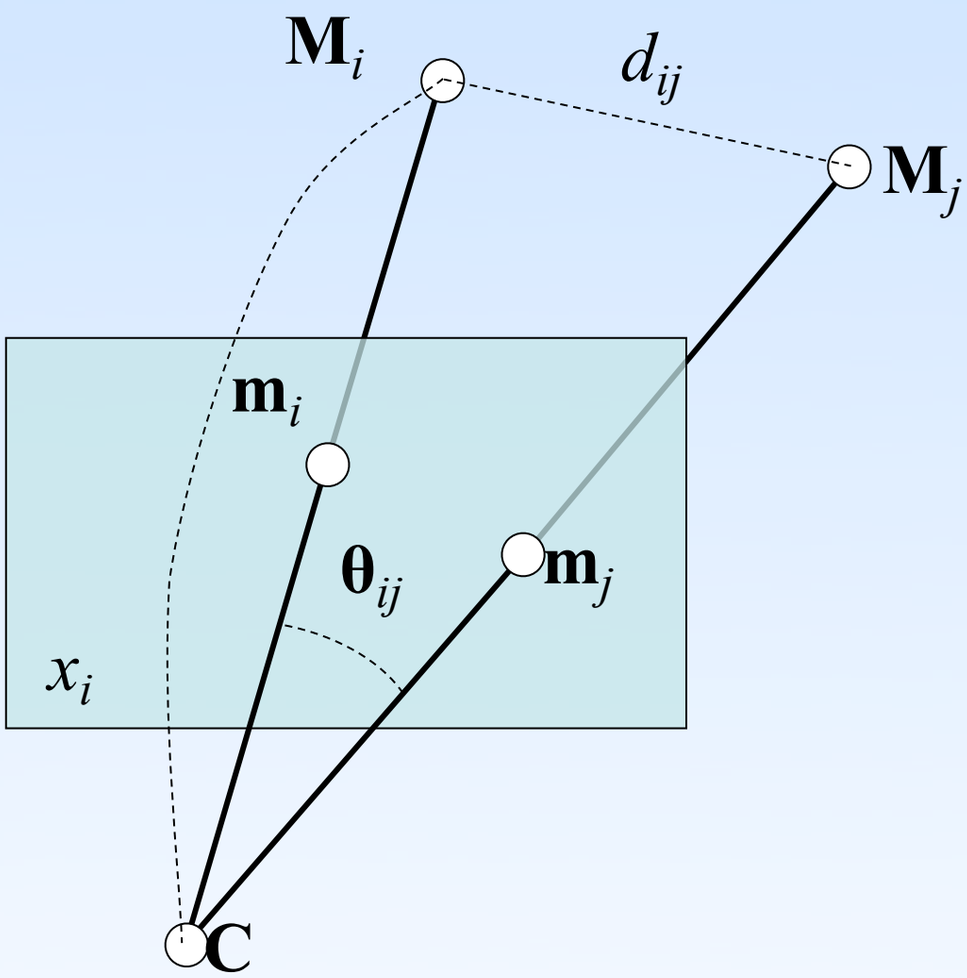
$x_i = \|\mathbf{M}_i - \mathbf{C}\|$ and $x_j = \|\mathbf{M}_j - \mathbf{C}\|$:

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

where:

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

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 $x_j = ||\mathbf{M}_j - \mathbf{C}||$:

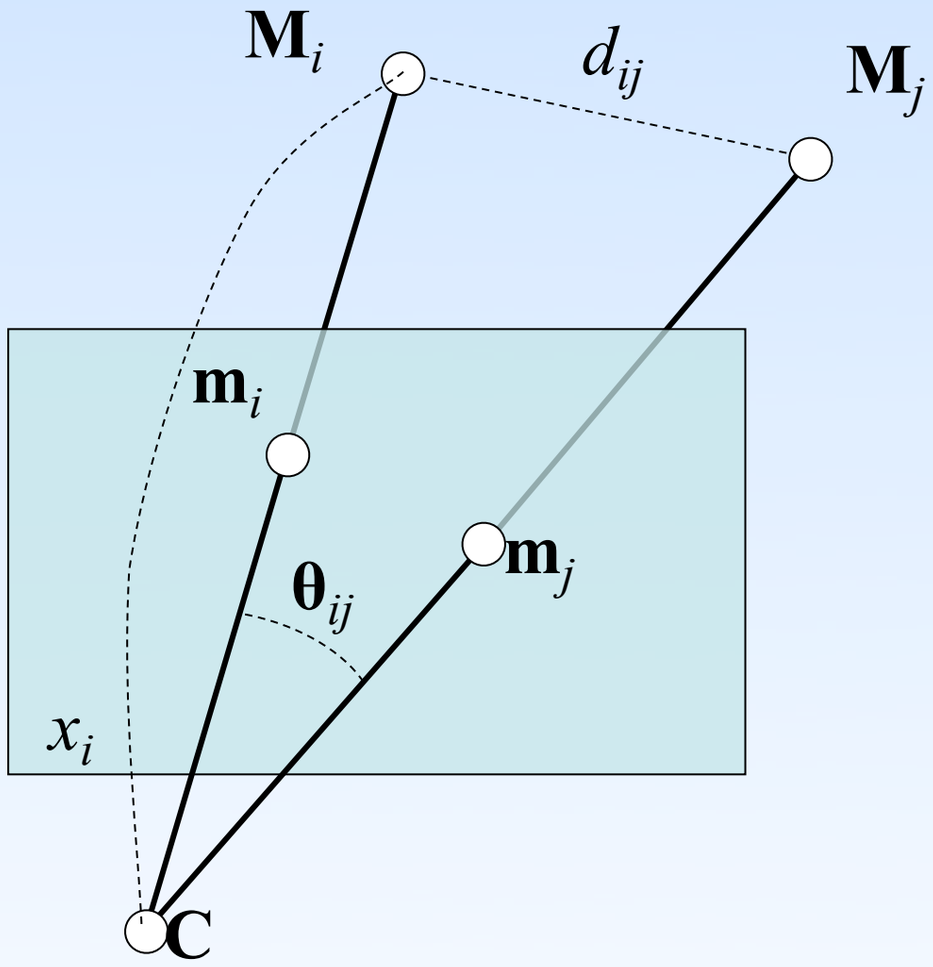
$$d_{ij}^2 = x_i^2 + x_j^2 - 2\, x_i\, x_j \cos\theta_{ij},$$

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 suspended at the camera center by $\mathbf{M}_i$ and $\mathbf{M}_j$.

Overview of the P3P

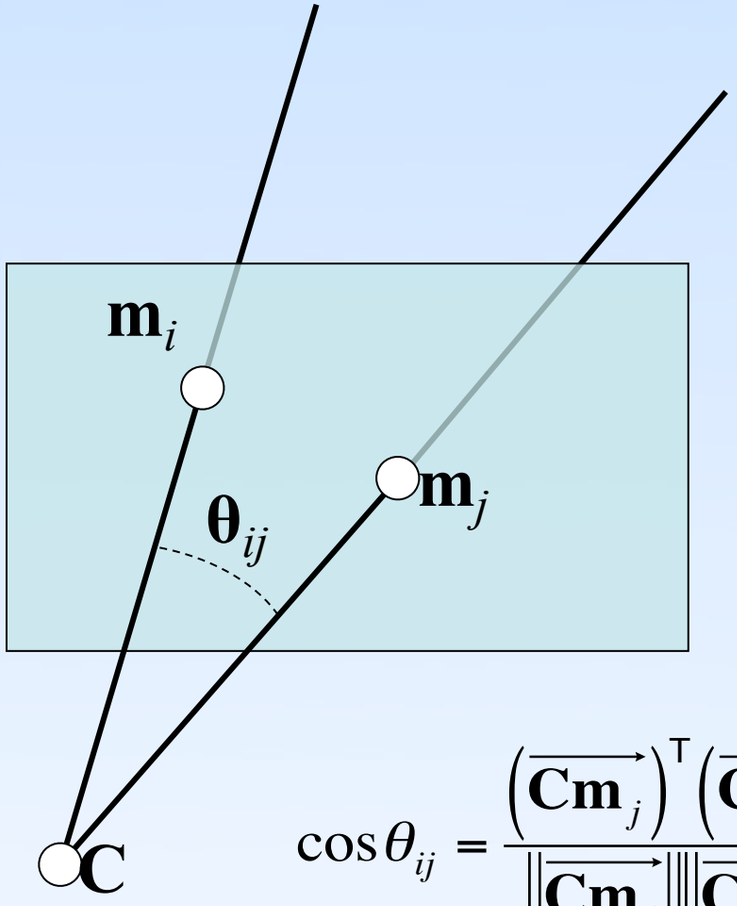


1. Solve for the distances x_i ;

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 $\mathbf{m}_i$ and $\mathbf{m}_j$.

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

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

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

Quadratic System

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

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

$h(x_1)$: 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.

Quadratic System

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

$$\begin{cases} f_{12}(x_1,x_2) = 0 \\ f_{13}(x_1,x_3) = 0 \\ f_{23}(x_2,x_3) = 0 \end{cases} \xrightarrow{\text{Resultant}} g(x_1,x_2) \quad \text{[degree 4]}$$

$h(x_1)$: 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;

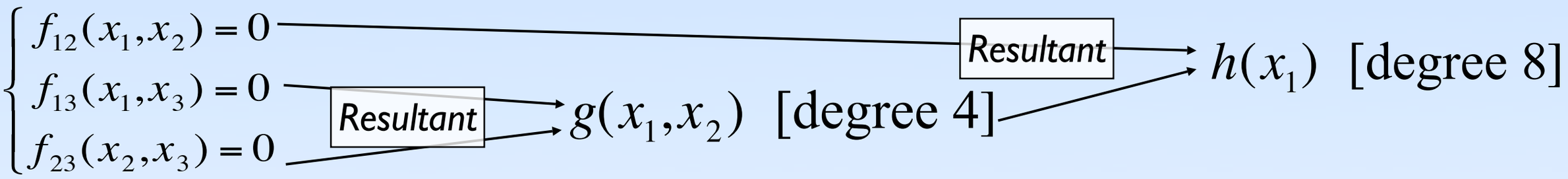
Can be solved in closed form.

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.

Quadratic System

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



$h(x_1)$: 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):

$$p_1(x) = a_m x^m + a_{m\text{-}1} x^{m\text{-}1} + \ldots + a_1 x + a_0$$
$$p_2(x) = b_n x^n + b_{n\text{-}1} x^{n\text{-}1} + \ldots + b_1 x + b_0$$

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

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

Resultant of Two Polynomials

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

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

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

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

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

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:

$$p_1(x, y) = 6x^2 + 3xy - xy^2 + y + 1$$
$$p_2(x, y) = x^2y + 5x + 4y - 1$$

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

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

$$p_1(x, y) = 6x^2 + (3y - y^2)x + (y + 1)$$
$$p_2(x, y) = y\,x^2 + 5x + (4y - 1)$$

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

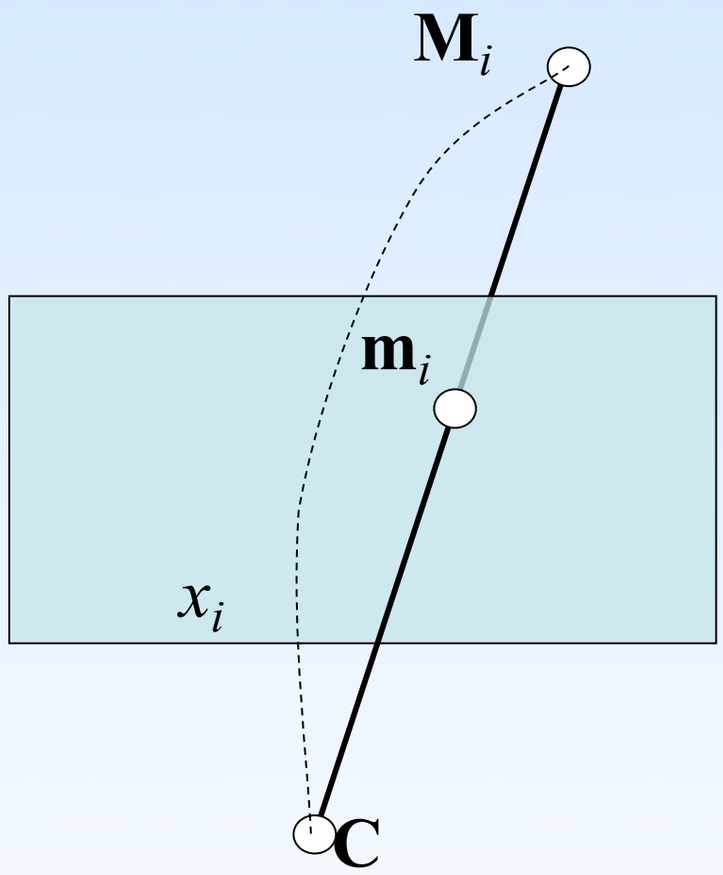
determinant(Syl(p_1, p_2)) = $4y^6 - 20y^5 + 153y^4 - 310y^3 + 781y^2 - 276\,y + 36$ is a polynomial in y only !

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

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

Estimating **R** and **T**

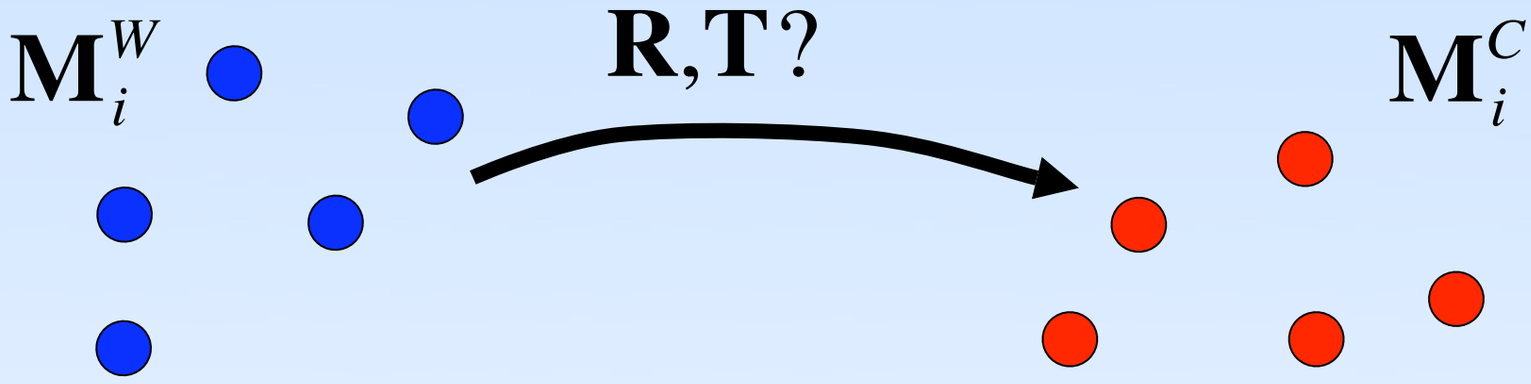
The coordinates of the 3D points $\mathbf{M}_i$ in the **camera coordinates system** can now be computed:



$$\mathbf{M}_i^C = x_i \mathbf{A}^{-1} \mathbf{m}_i$$

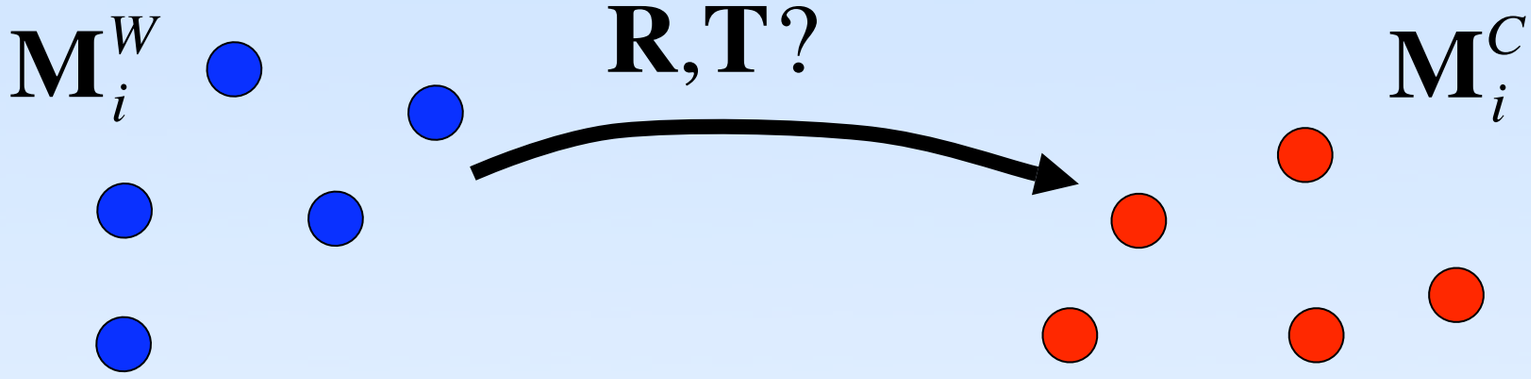
Estimating **R** and **T**

The positions $\mathbf{M}_i^C$ of the points $\mathbf{M}_i$ in the camera coordinates system can now be computed:



Estimating **R** and **T**

The positions $\mathbf{M}_i^C$ of the points $\mathbf{M}_i$ in the camera coordinates system can now be computed:

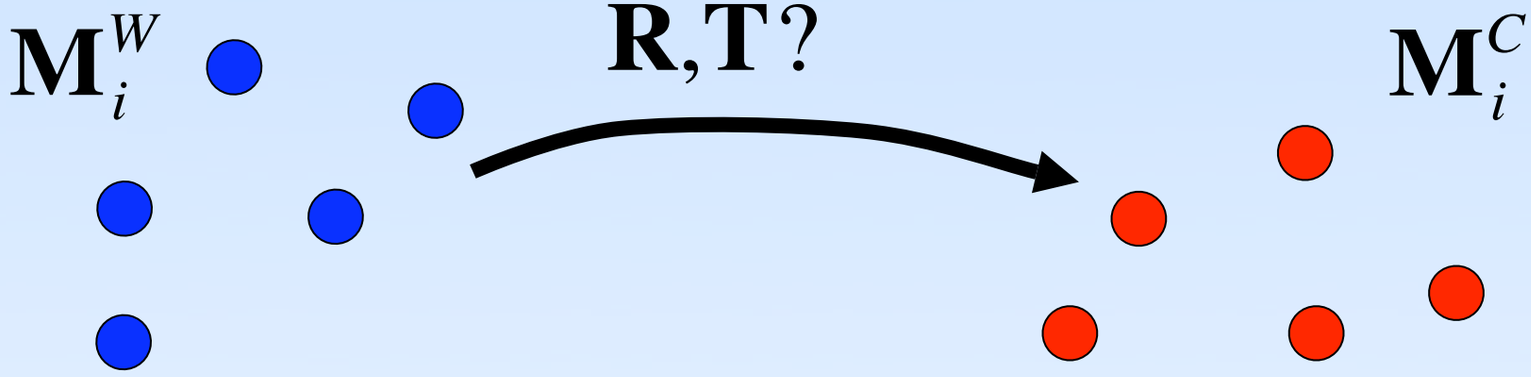


$$\overline{\mathbf{M}}^W = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}_i^W \rightarrow \mathbf{N}_i^W = \mathbf{M}_i^W - \overline{\mathbf{M}}^W$$

$$\overline{\mathbf{M}}^C = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}_i^C \rightarrow \mathbf{N}_i^C = \mathbf{M}_i^C - \overline{\mathbf{M}}^C$$

Estimating **R** and **T**

The positions $\mathbf{M}_i^C$ of the points $\mathbf{M}_i$ in the camera coordinates system can now be computed:

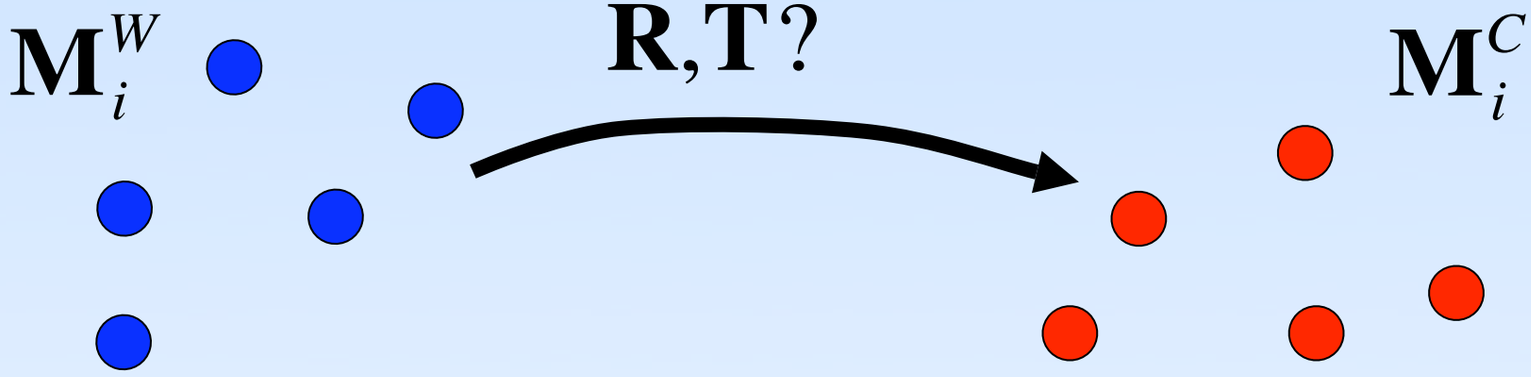


$$\overline{\mathbf{M}}^W = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}_i^W \rightarrow \mathbf{N}_i^W = \mathbf{M}_i^W - \overline{\mathbf{M}}^W \qquad \overline{\mathbf{M}}^C = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}_i^C \rightarrow \mathbf{N}_i^C = \mathbf{M}_i^C - \overline{\mathbf{M}}^C$$

$$\max_{\mathbf{R}\text{ rotation matrix}} \sum_i \left(\mathbf{N}_i^C\right)^t \cdot \left(\mathbf{R}\mathbf{N}_i^W\right)$$

Estimating **R** and **T**

The positions $\mathbf{M}_i^C$ of the points $\mathbf{M}_i$ in the camera coordinates system can now be computed:



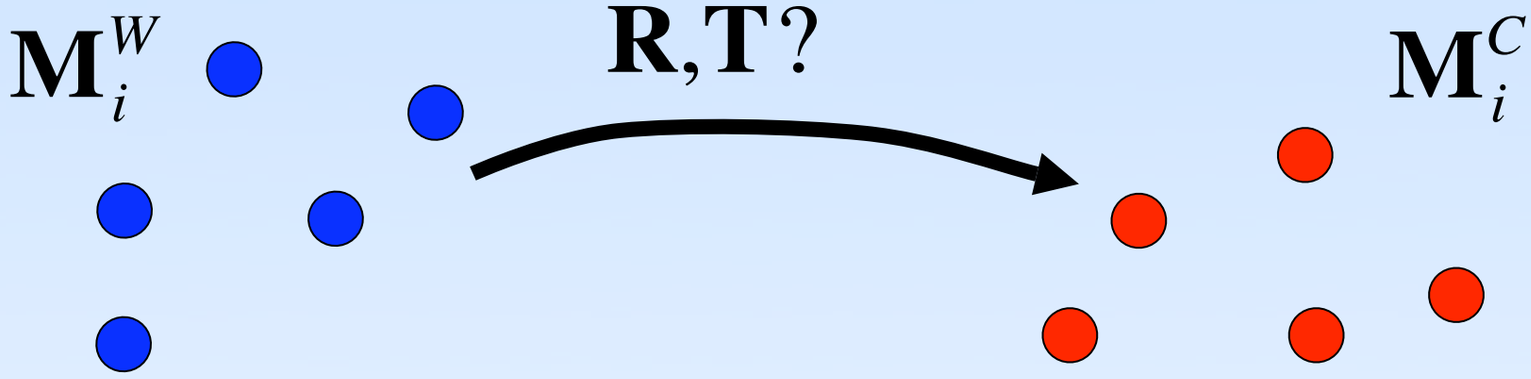
$$\overline{\mathbf{M}}^W = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}_i^W \rightarrow \mathbf{N}_i^W = \mathbf{M}_i^W - \overline{\mathbf{M}}^W \qquad \overline{\mathbf{M}}^C = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}_i^C \rightarrow \mathbf{N}_i^C = \mathbf{M}_i^C - \overline{\mathbf{M}}^C$$

$$\max_{\mathbf{R}\text{ rotation matrix}} \sum_i \left(\mathbf{N}_i^C\right)^t .\left(\mathbf{R}\mathbf{N}_i^W\right)$$

$$\mathbf{v}^t .\left(\mathbf{R}\mathbf{u}\right) = \text{trace}\left(\mathbf{R}^t\mathbf{u}\mathbf{v}^t\right) \rightarrow \sum_i \left(\mathbf{N}_i^C\right)^t .\left(\mathbf{R}\mathbf{N}_i^W\right) = \text{trace}\left(\mathbf{R}^t \sum_i \left(\mathbf{N}_i^C\right)^t .\mathbf{N}_i^W\right) = \text{trace}\left(\mathbf{R}^t\mathbf{L}\right) \text{ where } \mathbf{L} = \sum_i \left(\mathbf{N}_i^C\right)^t .\mathbf{N}_i^W$$

Estimating **R** and **T**

The positions $\mathbf{M}_i^C$ of the points $\mathbf{M}_i$ in the camera coordinates system can now be computed:



$$\overline{\mathbf{M}}^W = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}_i^W \rightarrow \mathbf{N}_i^W = \mathbf{M}_i^W - \overline{\mathbf{M}}^W \qquad \overline{\mathbf{M}}^C = \frac{1}{n}\sum_{i=1}^{n}\mathbf{M}_i^C \rightarrow \mathbf{N}_i^C = \mathbf{M}_i^C - \overline{\mathbf{M}}^C$$

$$\max_{\mathbf{R}\text{ rotation matrix}} \sum_i \left(\mathbf{N}_i^C\right)^t .\left(\mathbf{R}\mathbf{N}_i^W\right)$$

$$\mathbf{v}^t .(\mathbf{R}\mathbf{u}) = \text{trace}\left(\mathbf{R}^t\mathbf{u}\mathbf{v}^t\right) \rightarrow \sum_i \left(\mathbf{N}_i^C\right)^t .\left(\mathbf{R}\mathbf{N}_i^W\right) = \text{trace}\left(\mathbf{R}^t \sum_i \left(\mathbf{N}_i^C\right)^t .\mathbf{N}_i^W\right) = \text{trace}\left(\mathbf{R}^t\mathbf{L}\right) \text{ where } \mathbf{L} = \sum_i \left(\mathbf{N}_i^C\right)^t .\mathbf{N}_i^W$$

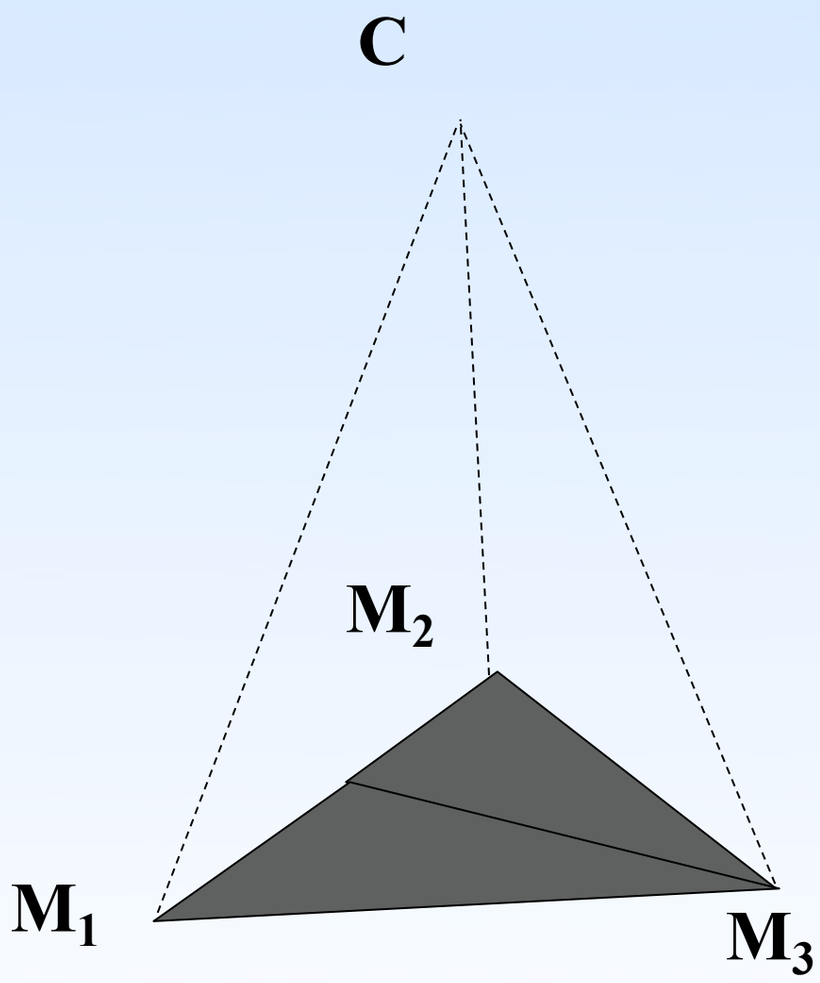
$$\mathbf{R} = \underset{\mathbf{R}\text{ rotation matrix}}{\arg\max}\ \text{trace}(\mathbf{R}^t\mathbf{L})$$

$$\rightarrow \mathbf{R} = \mathbf{U}\mathbf{V}^t \text{ where } \mathbf{L} = \mathbf{U}\mathbf{S}\mathbf{V}^t$$

$$\mathbf{T} = \overline{\mathbf{M}}^C - \mathbf{R}\overline{\mathbf{M}}^W$$

What are the possible solutions?

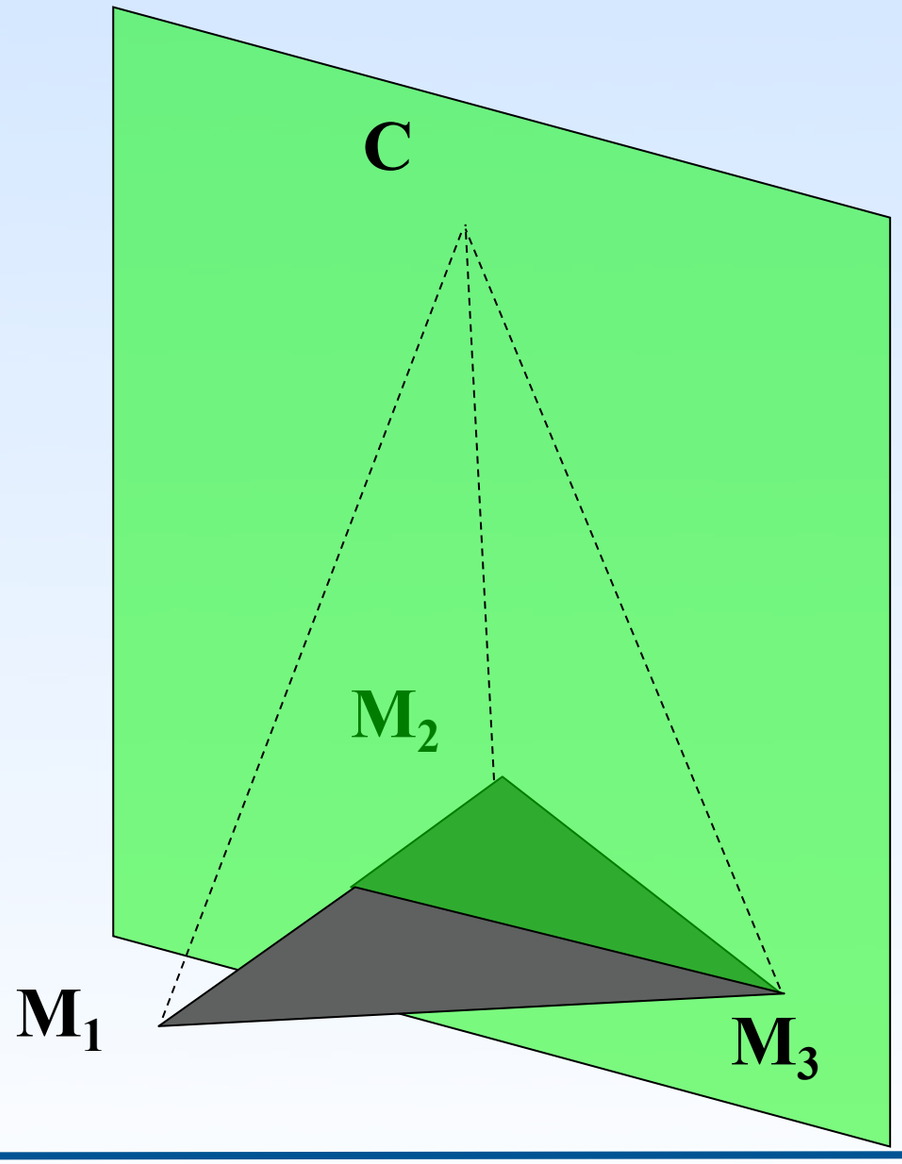
Special case: the center of projection lies on the plane perpendicular to the triangle:



[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

What are the possible solutions?

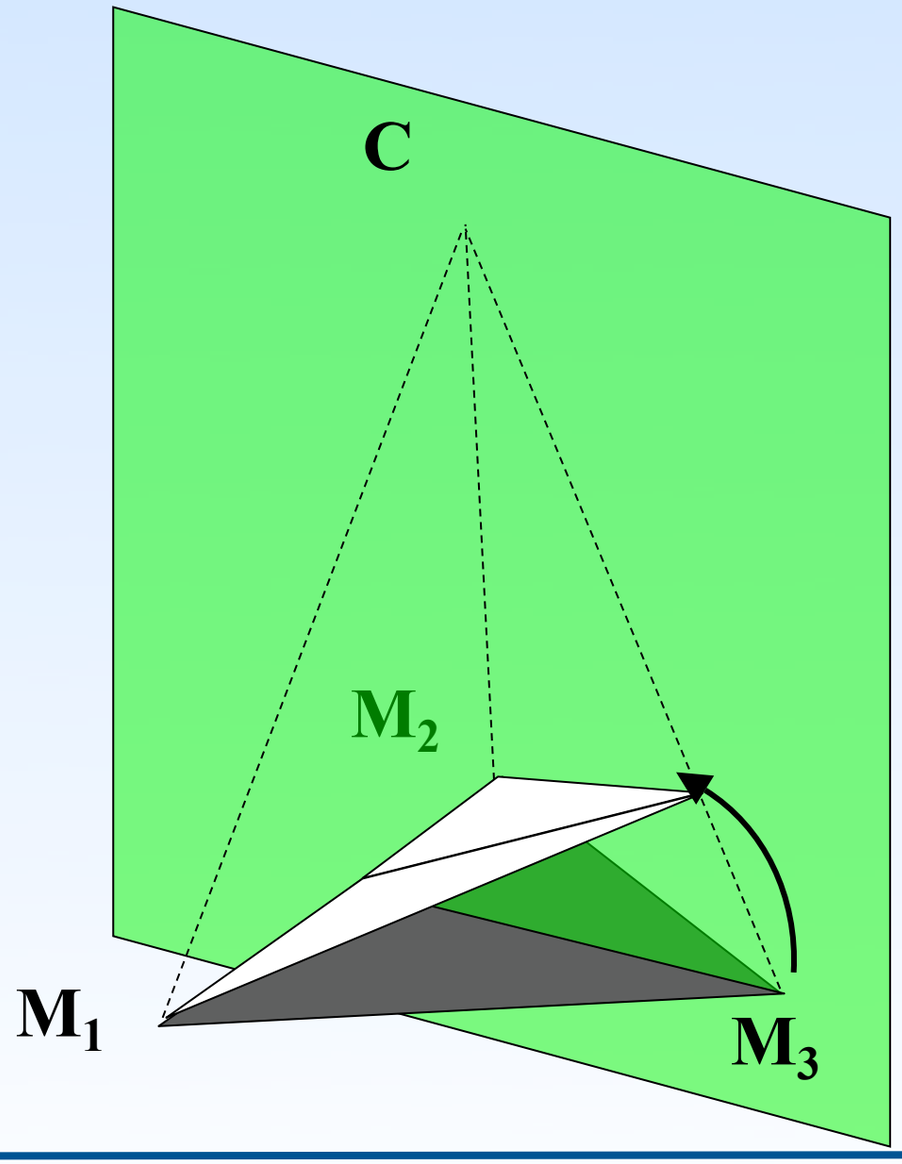
Special case: the center of projection lies on the plane perpendicular to the triangle:



[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

What are the possible solutions?

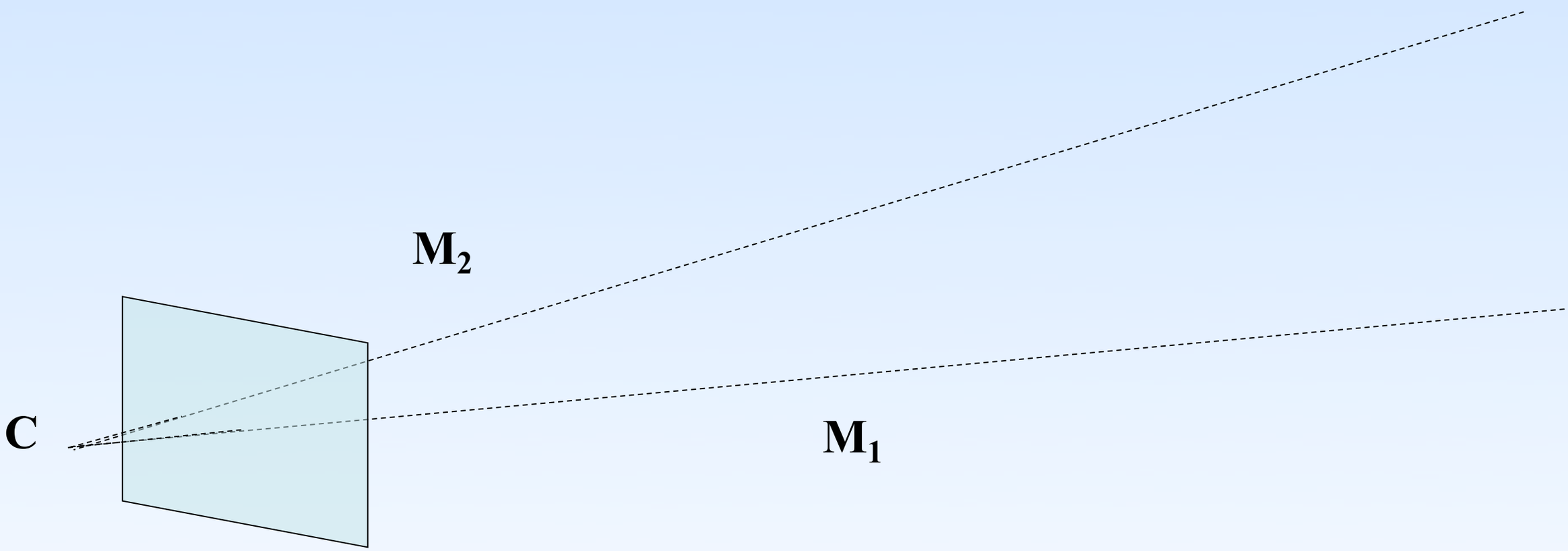
Special case: the center of projection lies on the plane perpendicular to the triangle:



[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

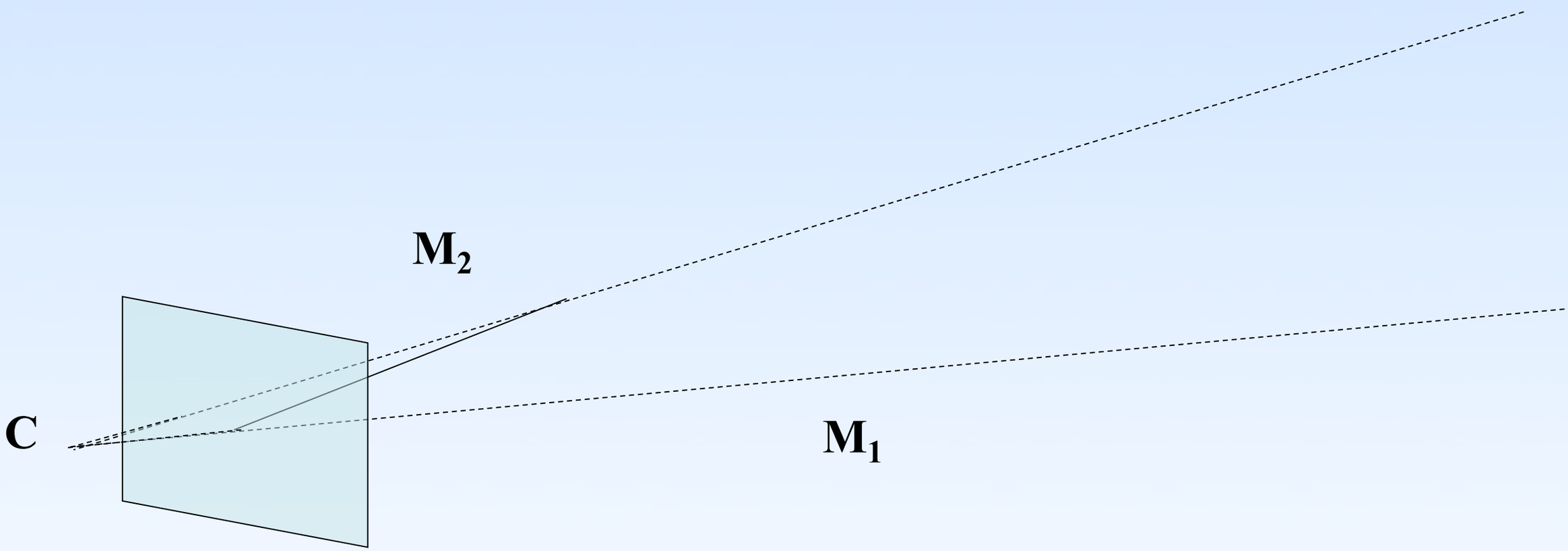
What are the possible solutions?

General case:



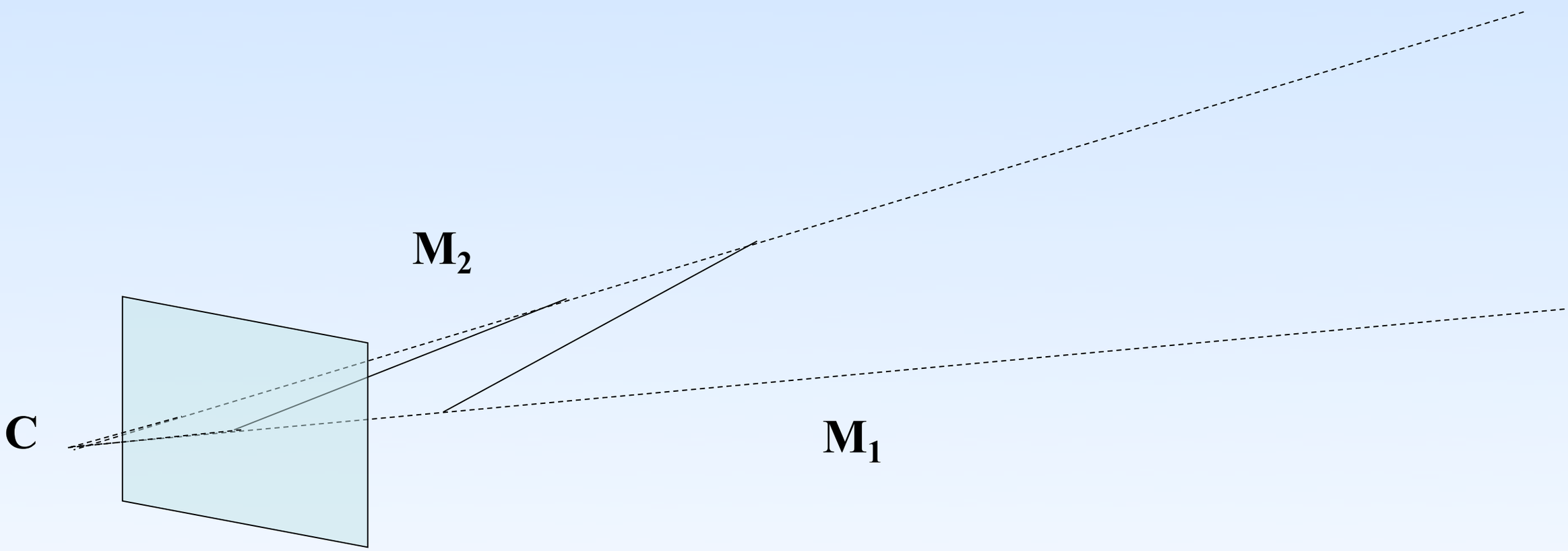
What are the possible solutions?

General case:



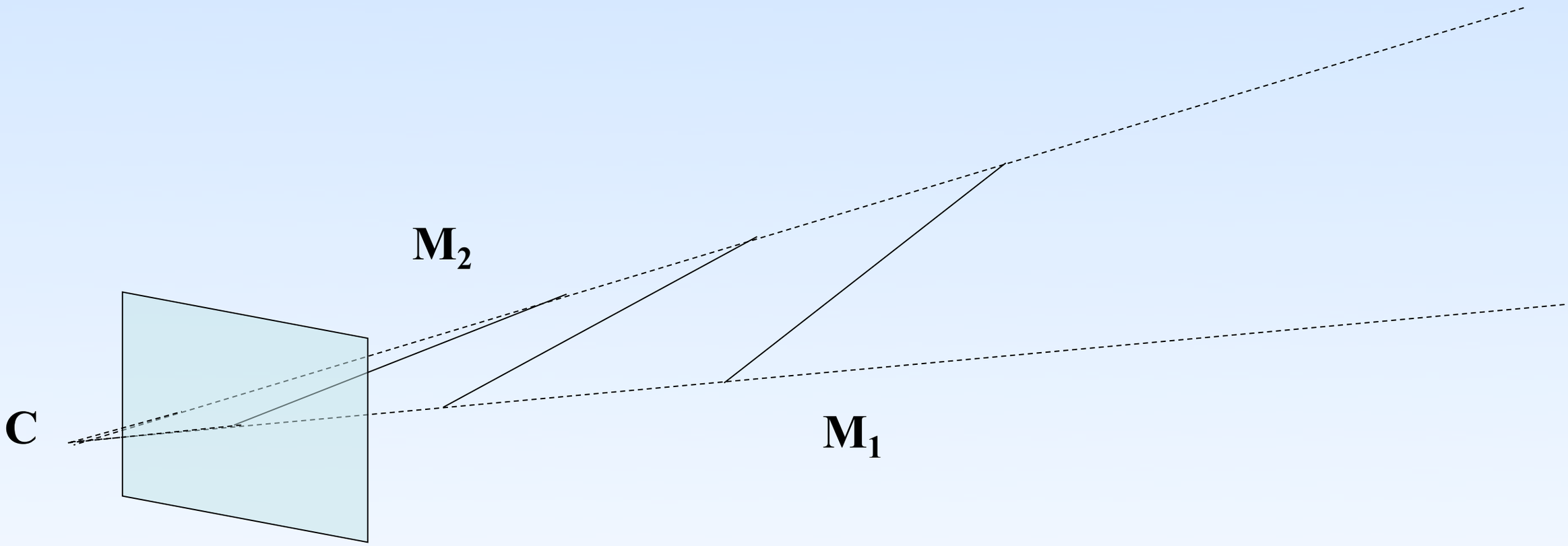
What are the possible solutions?

General case:



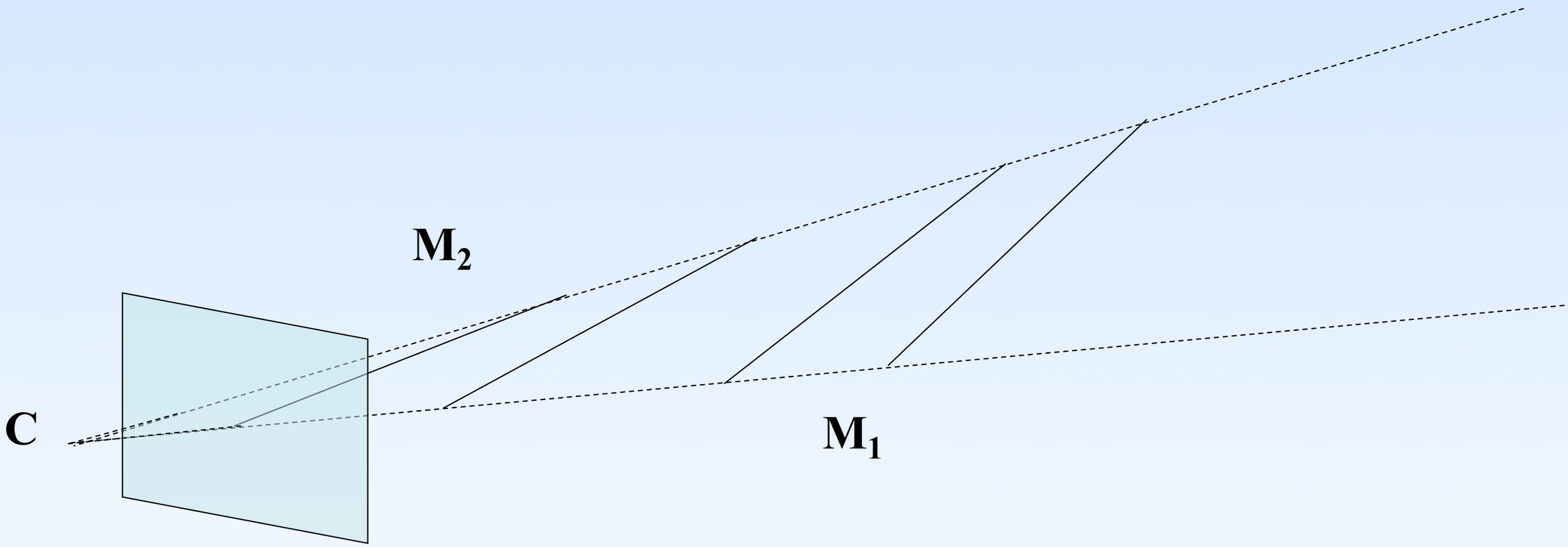
What are the possible solutions?

General case:



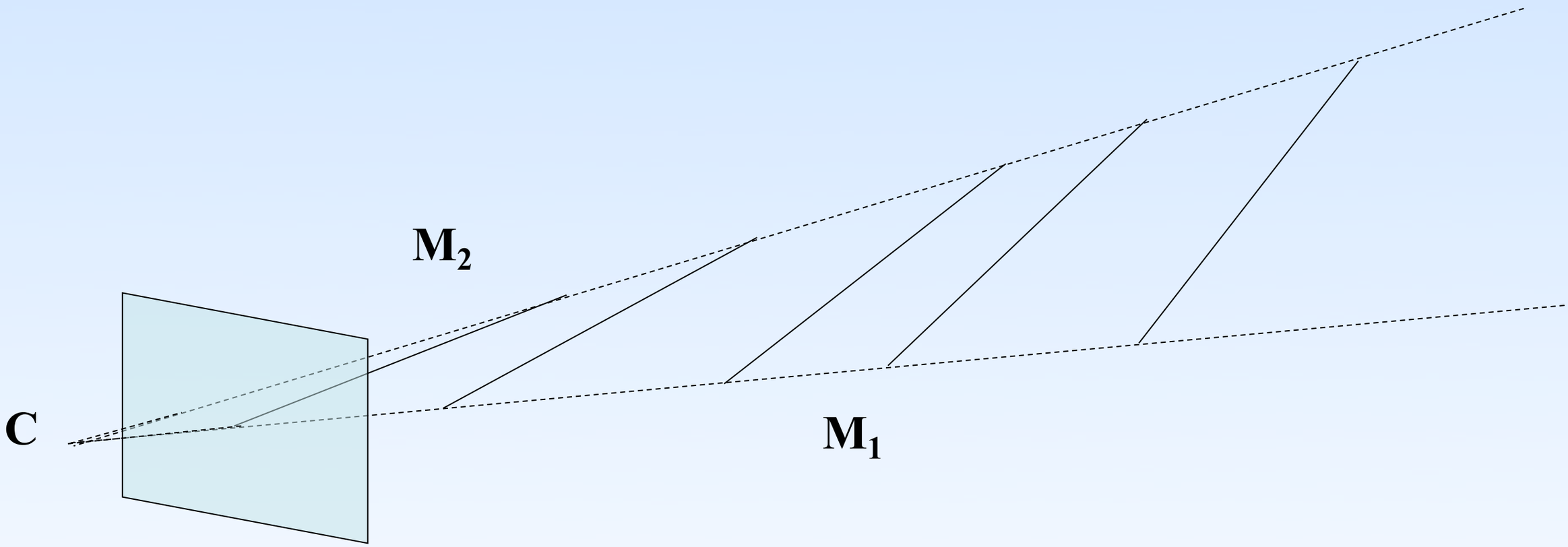
What are the possible solutions?

General case:



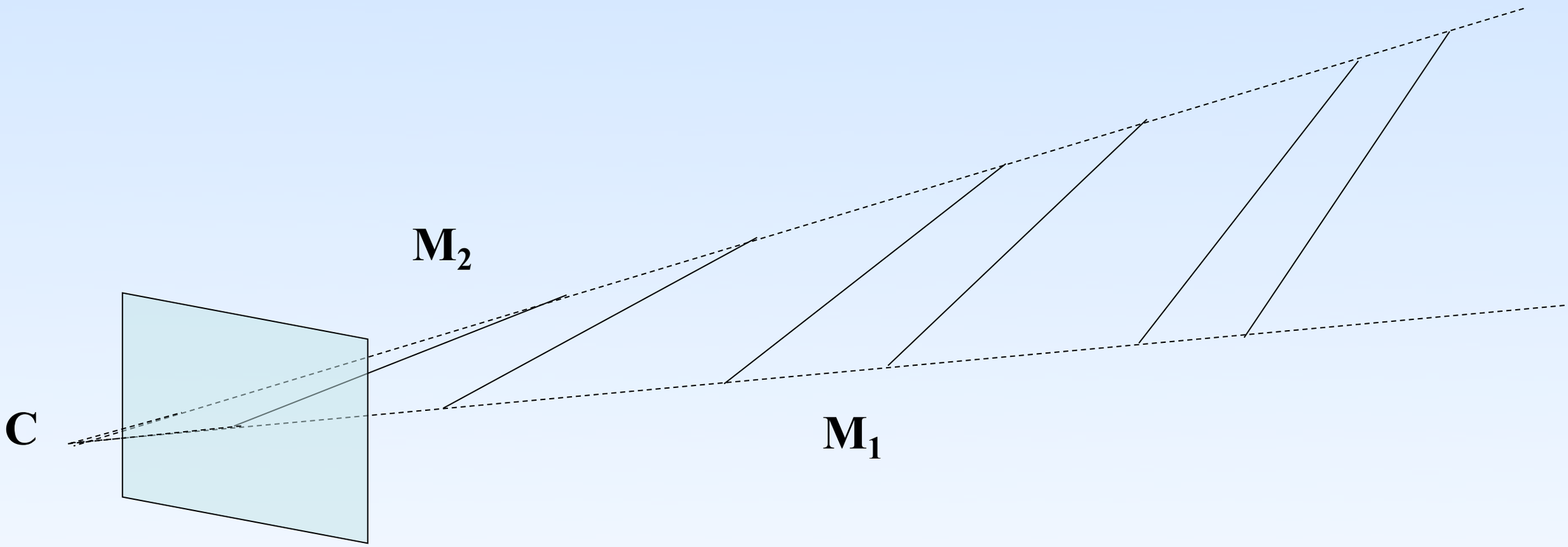
What are the possible solutions?

General case:



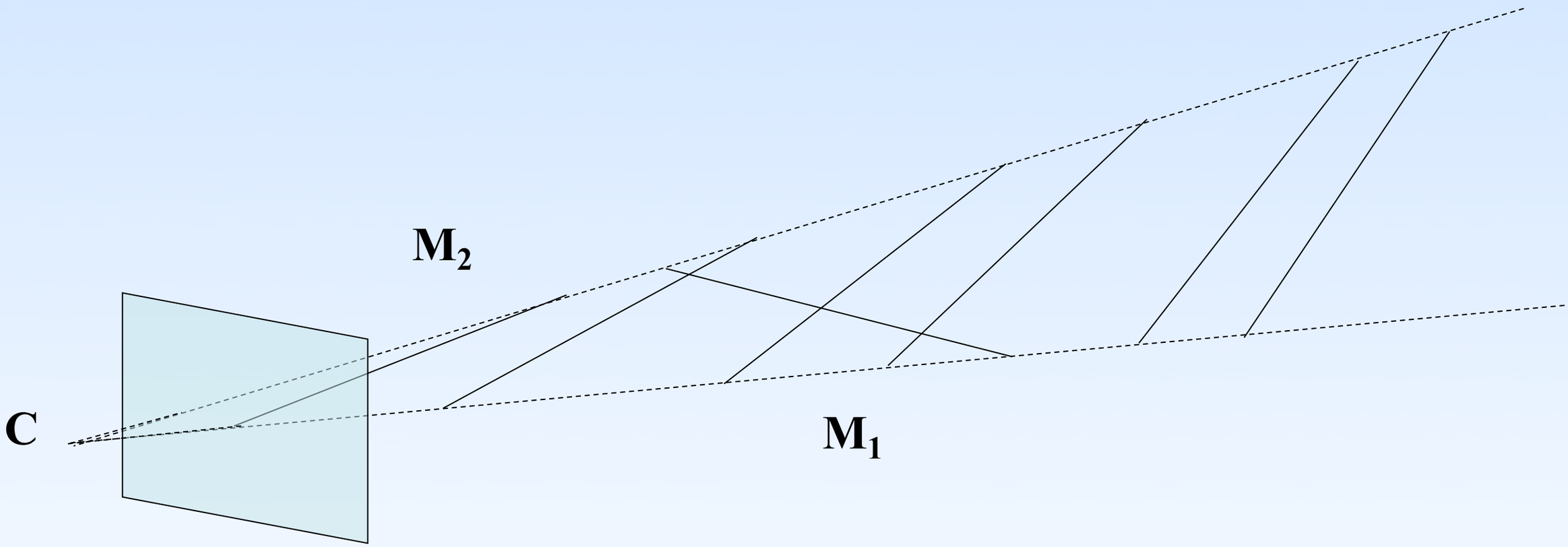
What are the possible solutions?

General case:



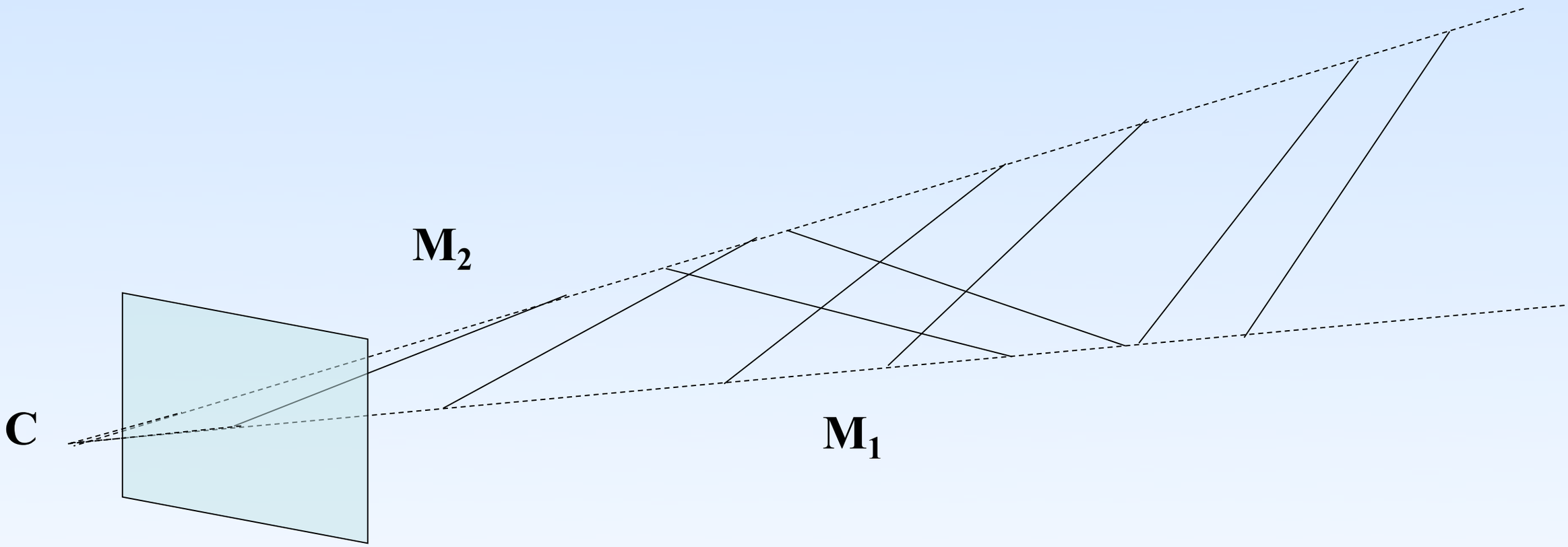
What are the possible solutions?

General case:



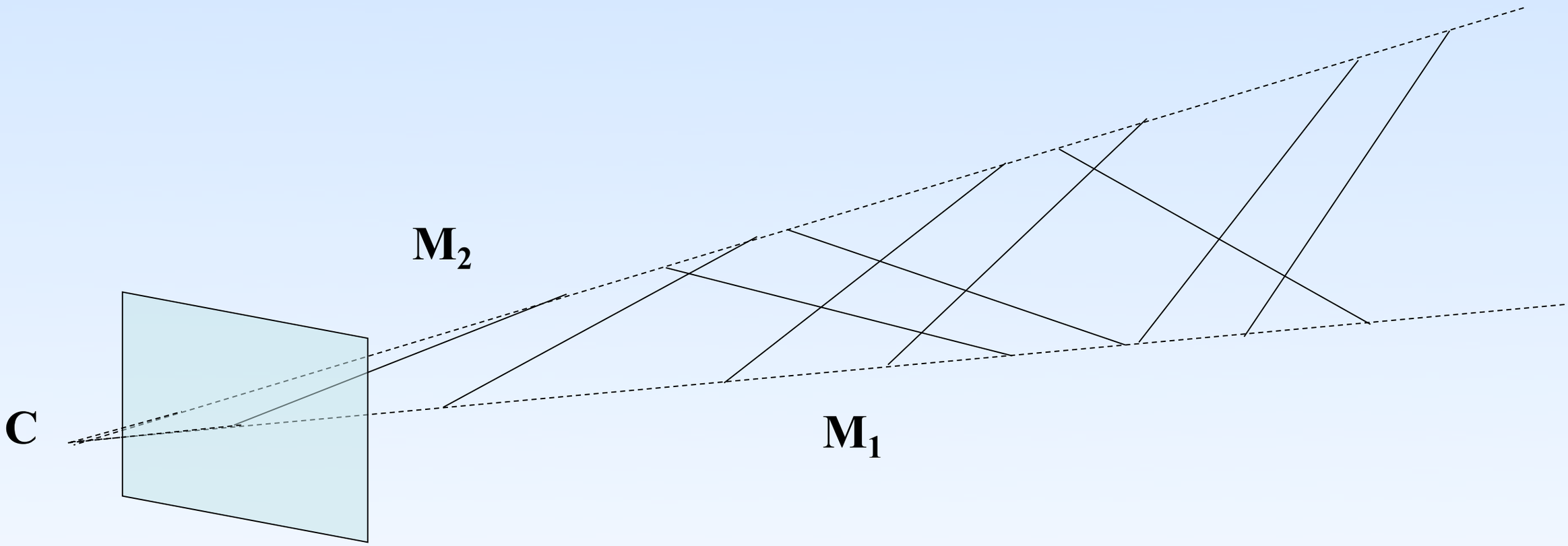
What are the possible solutions?

General case:



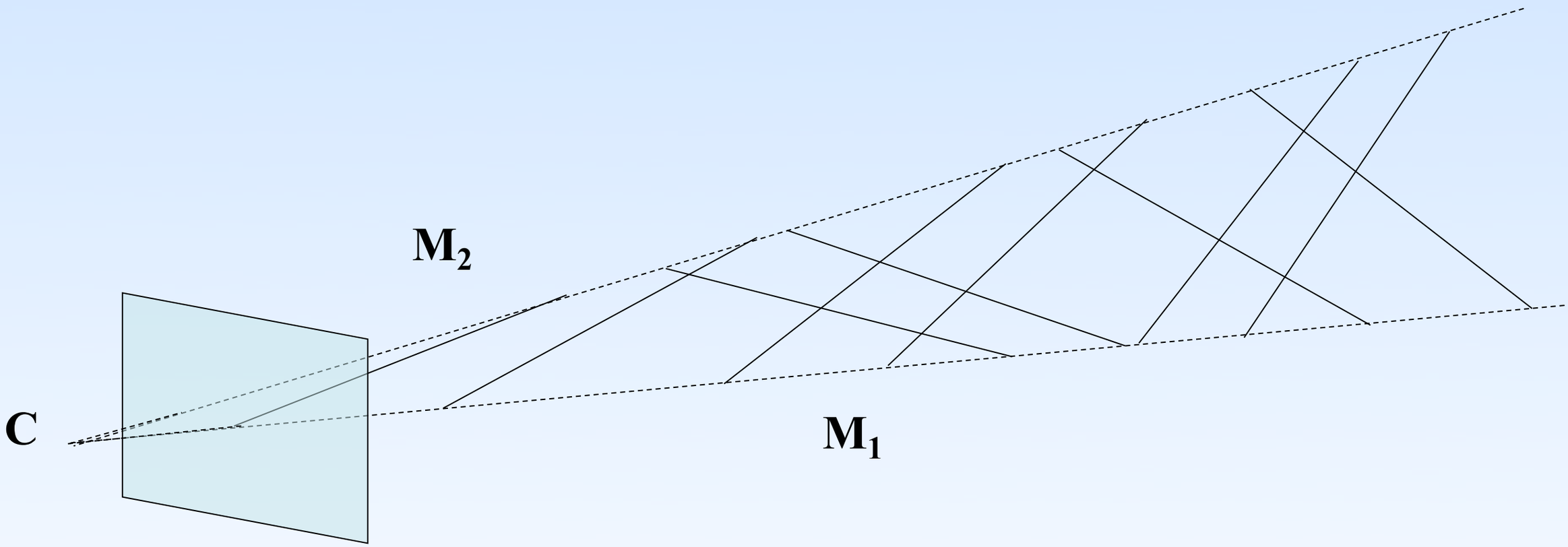
What are the possible solutions?

General case:



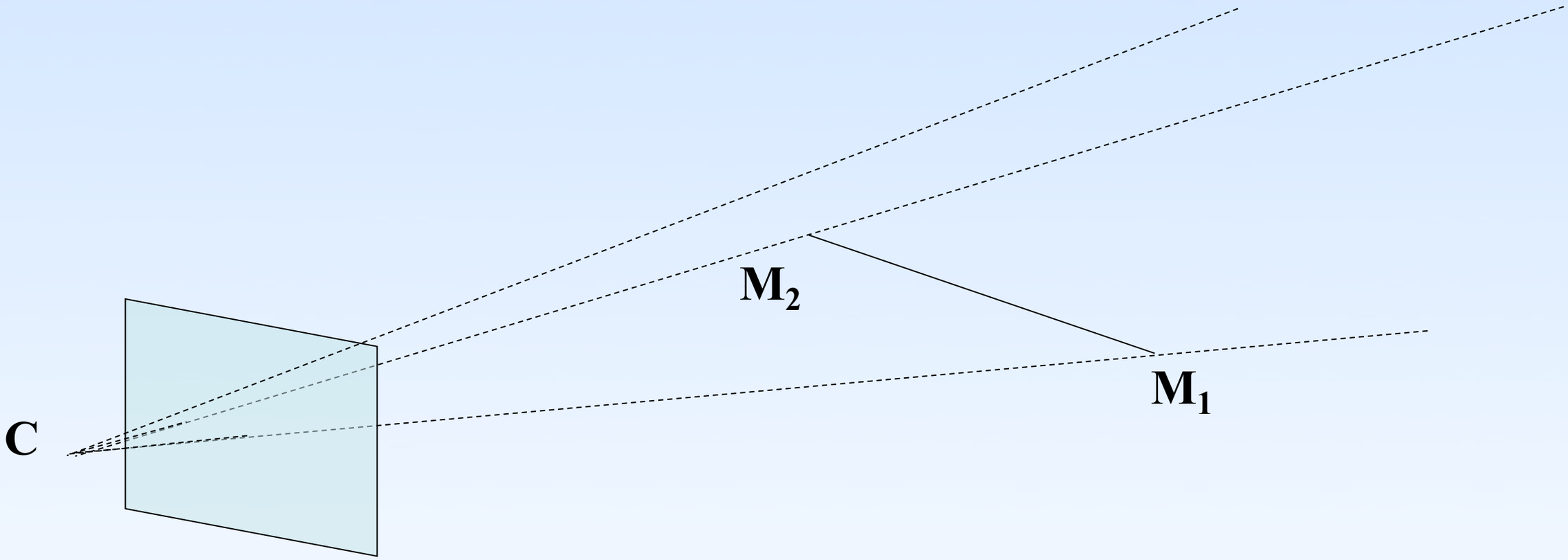
What are the possible solutions?

General case:



What are the possible solutions?

General case:



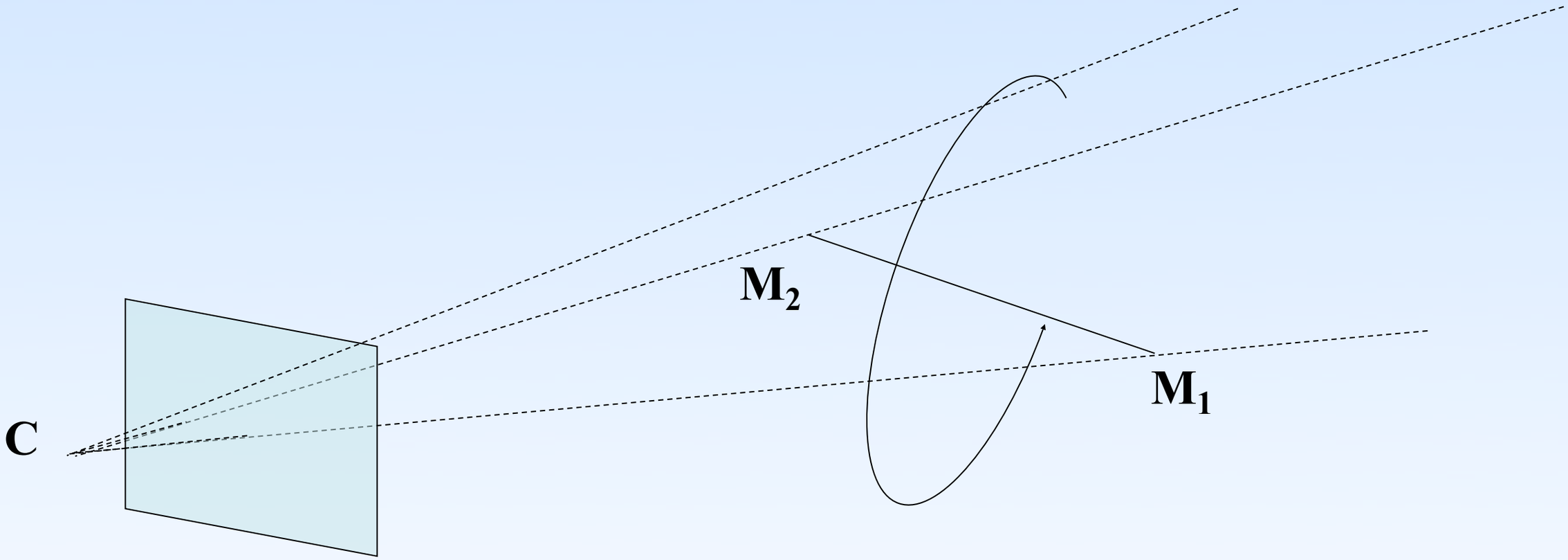
The set of circles generates a surface.

Each intersection between the line of sight $[CM_3)$ and this surface corresponds to one solution to the P3P problem.

[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

What are the possible solutions?

General case:



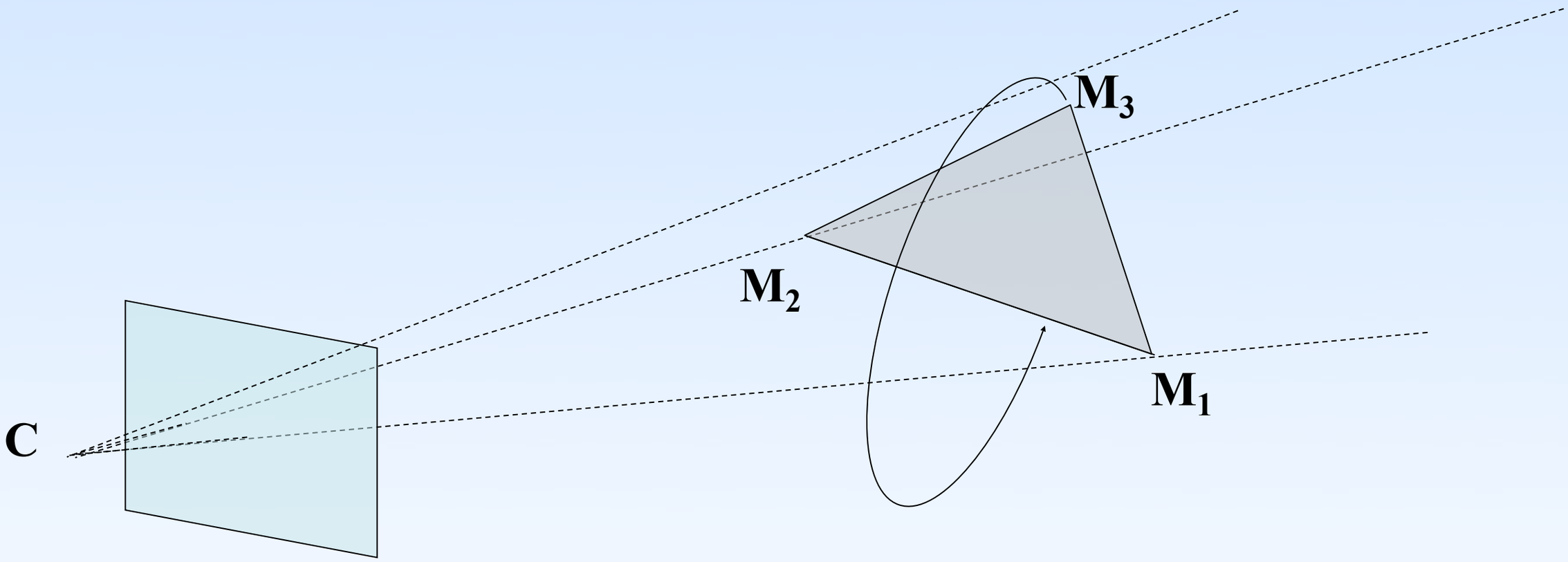
The set of circles generates a surface.

Each intersection between the line of sight $[CM_3)$ and this surface corresponds to one solution to the P3P problem.

[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

What are the possible solutions?

General case:



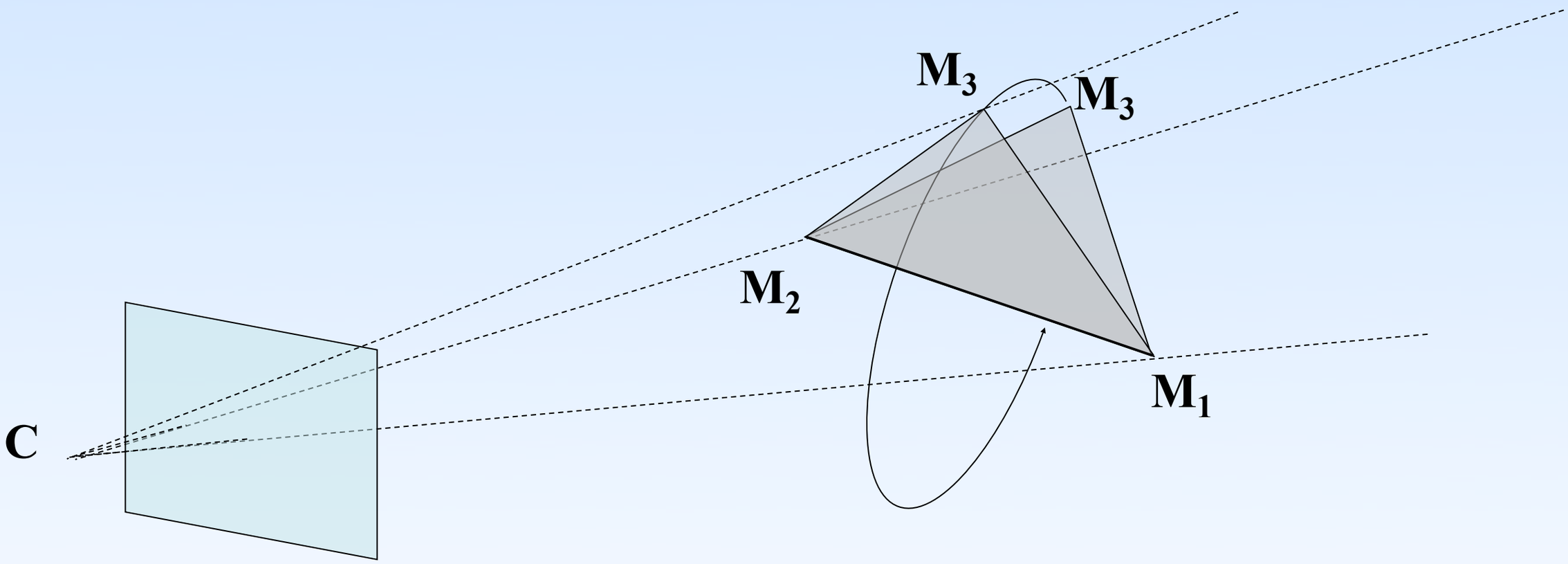
The set of circles generates a surface.

Each intersection between the line of sight $[CM_3)$ and this surface corresponds to one solution to the P3P problem.

[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

What are the possible solutions?

General case:



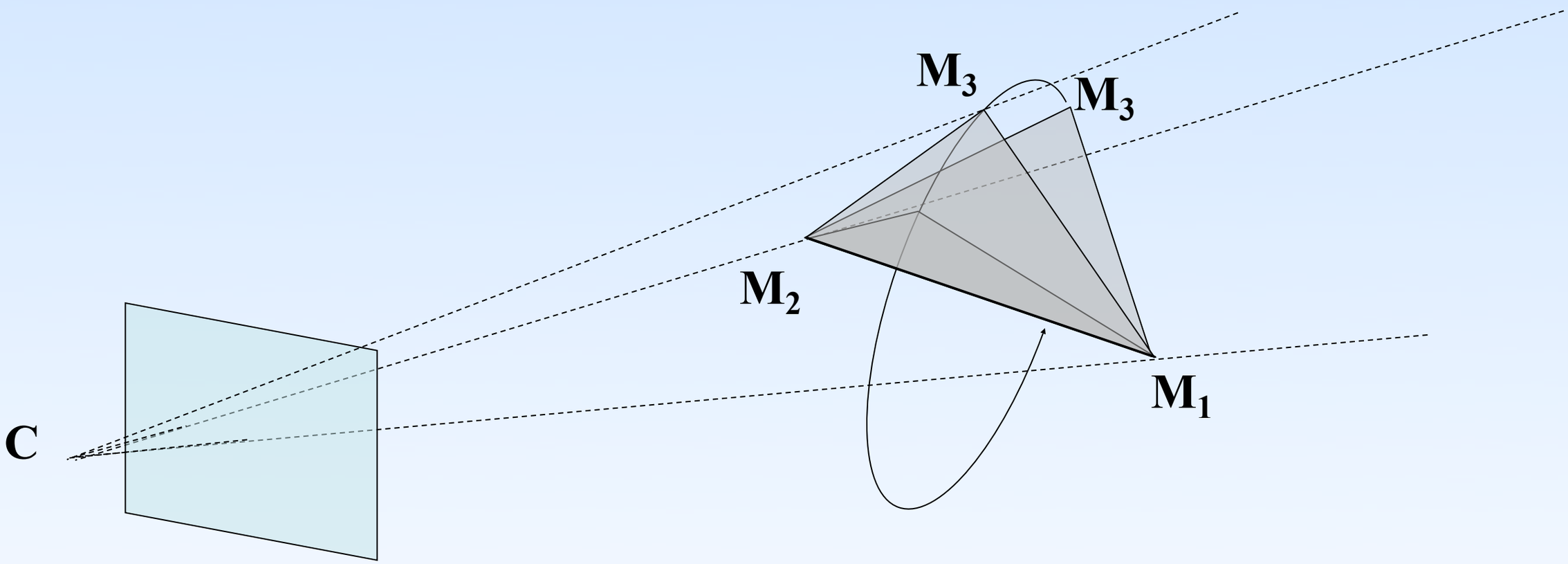
The set of circles generates a surface.

Each intersection between the line of sight $[CM_3)$ and this surface corresponds to one solution to the P3P problem.

[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

What are the possible solutions?

General case:



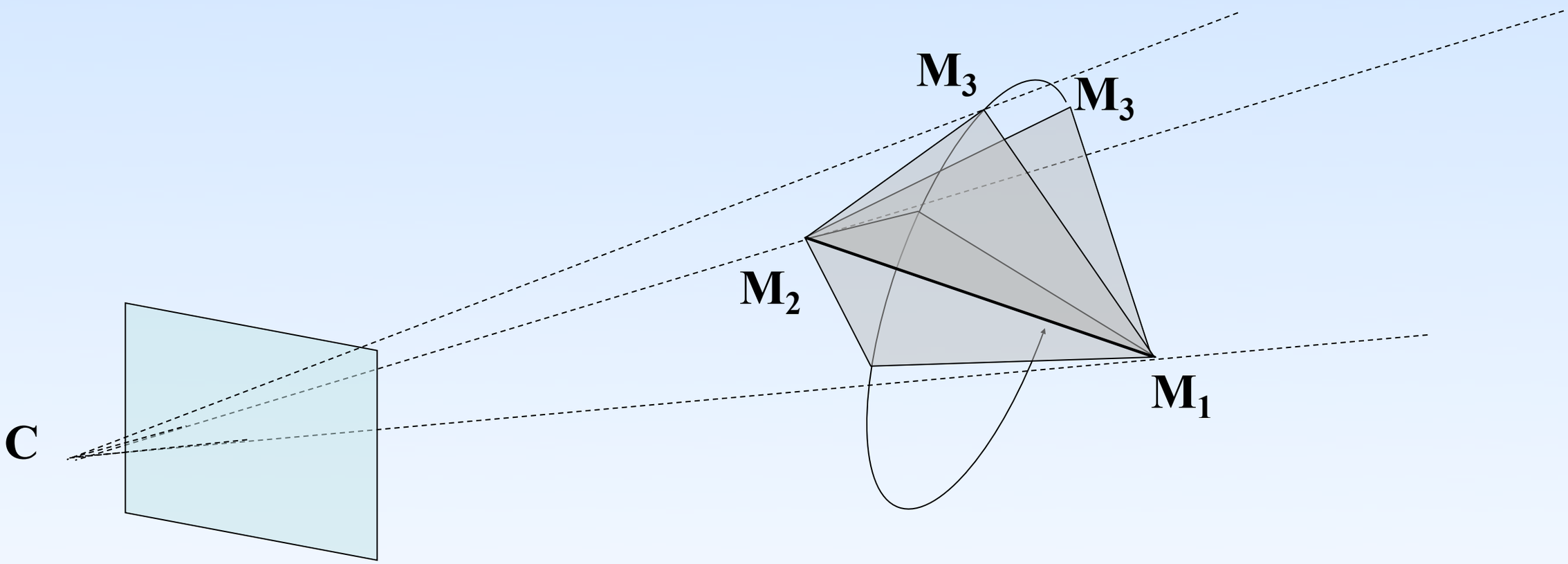
The set of circles generates a surface.

Each intersection between the line of sight $[CM_3)$ and this surface corresponds to one solution to the P3P problem.

[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

What are the possible solutions?

General case:



The set of circles generates a surface.

Each intersection between the line of sight $[CM_3)$ and this surface corresponds to one solution to the P3P problem.

[From "The Perspective View of Three Points", Wolfe et al. PAMI91]

P3P: Summary

Well adapted to the problem

Closed-form solution.

Only 3 points + one constraint needed to compute the pose.

Inconvenient:

Pose computed with only 3 points, can be inaccurate.

→ Pose refined by a non-linear estimation with all the points available.

P*n*P

Lu-Hager-Mjolsness: One on the most efficient algorithms in the general case.

Iterative algorithm.

Requires an initialization.

Pick a starting $\mathbf{R}_0$

Repeat

Compute $\mathbf{T}(\mathbf{R}_k)$ [can be computed in closed-form]

Compute $\mathbf{N}_i(\mathbf{R}_k)$

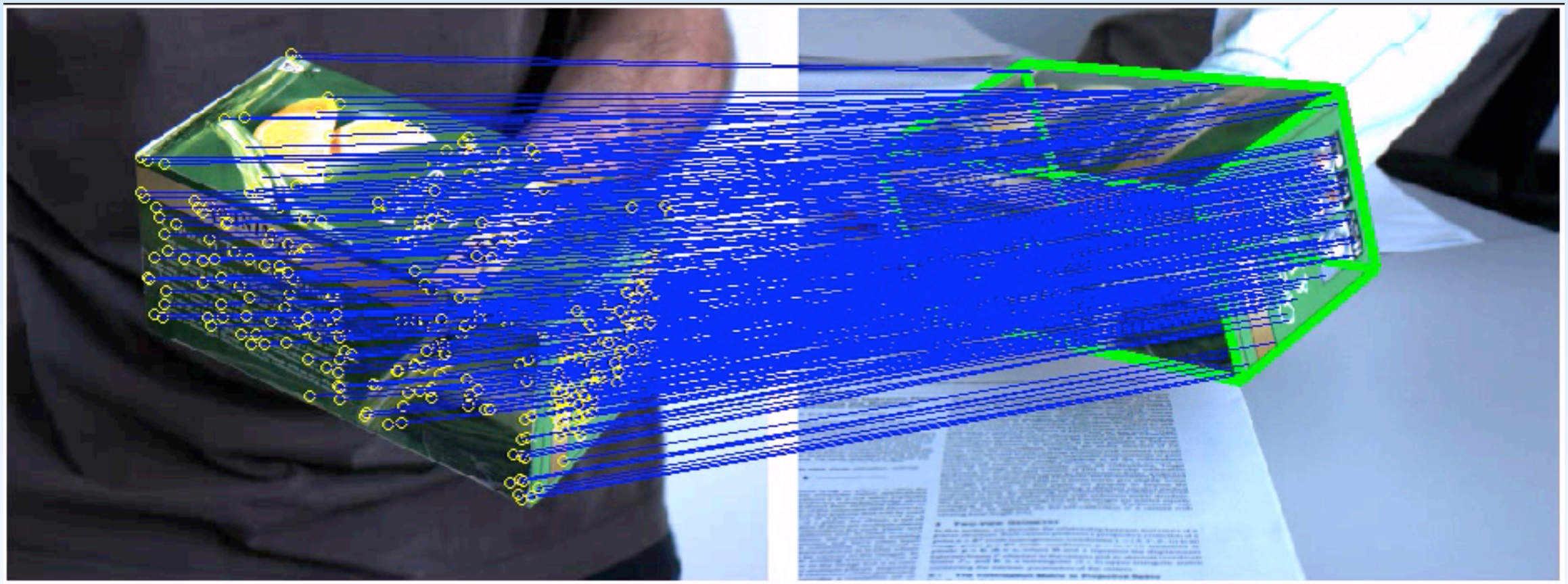
Compute $\mathbf{L}(\mathbf{R}_k)$

Compute $\mathbf{L}(\mathbf{R}_k) = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^t$

Set $\mathbf{R}_{k+1} = \mathbf{V}\,\mathbf{U}^t$

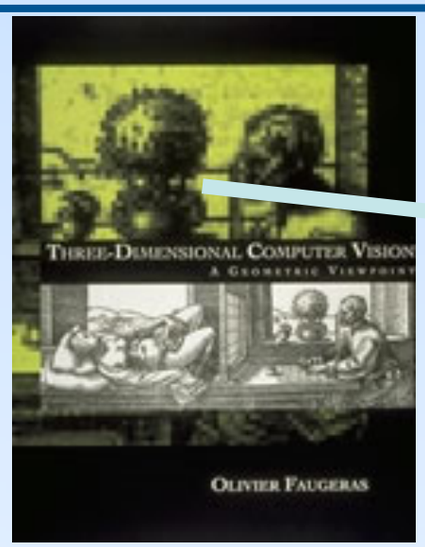
Until convergence. Can converge into a local minimum!

PnP algorithm: Application

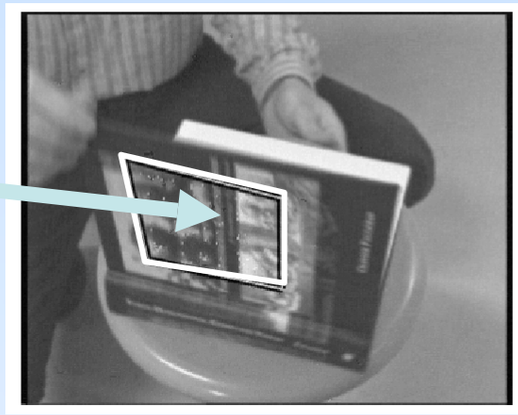


[Moreno-Noguer et al. ICCV07]

Pose Estimation from a Plane



H



Let assume that the homography between the plane and the plane image is known.

$$\mathbf{m}' = \mathbf{H}\mathbf{m}$$

$$\begin{pmatrix} ku' \\ kv' \\ k \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

$\mathbf{m}$ = $\mathbf{PM}$ with $\mathbf{M}$ a point on to the plane

$$= \mathrm{A}\begin{bmatrix}\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\ \mathbf{T}\end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$= A\begin{bmatrix}\mathbf{R}_1\mathbf{R}_2\ \mathbf{T}\end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$= \mathbf{H}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Pose Estimation from a Plane

We have:

$$\mathbf{H} \propto A\left[\mathbf{R}_1 \mathbf{R}_2 \; \mathbf{T}\right]$$

$\mathbf{R}_1, \mathbf{R}_2$ and $\mathbf{T}$ can be retrieved from the product $\mathbf{G} = \mathbf{A}^{-1}\mathbf{H}$

$\mathbf{G}$ is defined up to a scale factor and $\mathbf{R}_1$ and $\mathbf{R}_2$ are not perfectly orthonormal.

Normalization :

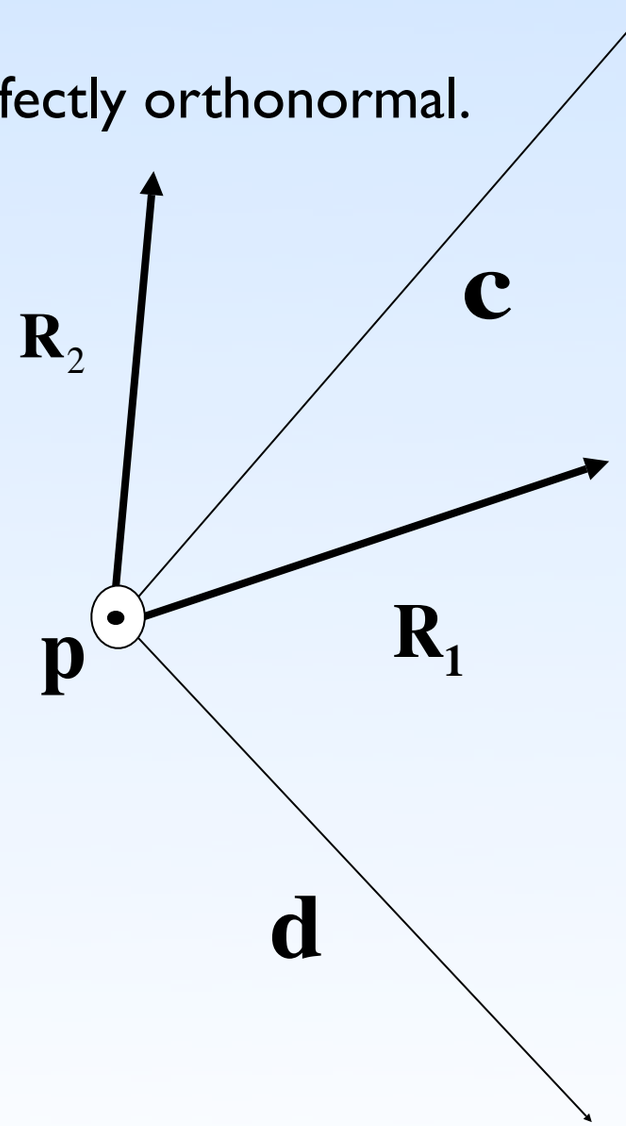
$$l = \sqrt{\|\mathbf{G}_1\| \|\mathbf{G}_2\|} \quad \mathbf{R}_1 = \frac{\mathbf{G}_1}{l}, \mathbf{R}_2 = \frac{\mathbf{G}_2}{l}, \mathbf{T} = \frac{\mathbf{G}_3}{l}$$

$$\mathbf{c} = \mathbf{R}_1 + \mathbf{R}_2, \mathbf{p} = \mathbf{R}_1 \times \mathbf{R}_2, \mathbf{d} = \mathbf{c} \times \mathbf{p}$$

$$\mathbf{R}_1' = \frac{1}{\sqrt{2}}\left(\frac{\mathbf{c}}{\|\mathbf{c}\|} + \frac{\mathbf{d}}{\|\mathbf{d}\|}\right), \mathbf{R}_2' = \frac{1}{\sqrt{2}}\left(\frac{\mathbf{c}}{\|\mathbf{c}\|} - \frac{\mathbf{d}}{\|\mathbf{d}\|}\right)$$

$$\mathbf{R}_3 = \mathbf{R}_1' \times \mathbf{R}_2'$$

+ Refinement with a non-linear optimization.



Pose Estimation from\ a Plane: Application

[Benhimane et al. CVPR06]

Pose Estimation from\ a Plane: Application

[Benhimane et al. CVPR06]

Non-Linear Optimization

- Non-linear least-squares minimization:

$$\min_{\mathbf{R},\mathbf{T}} \sum_i \left\| \mathbf{A}[\mathbf{R} \mid \mathbf{T}]\mathbf{M}_i - \mathbf{m}_i \right\|^2$$

- Minimization of a physical, meaningful error (re-projection error, in pixels)
- Handle an arbitrary number of points.
- Gauss-Newton, Levenberg-Marquardt... minimization (next lecture).