Lecture:

3D Reconstruction from visual data

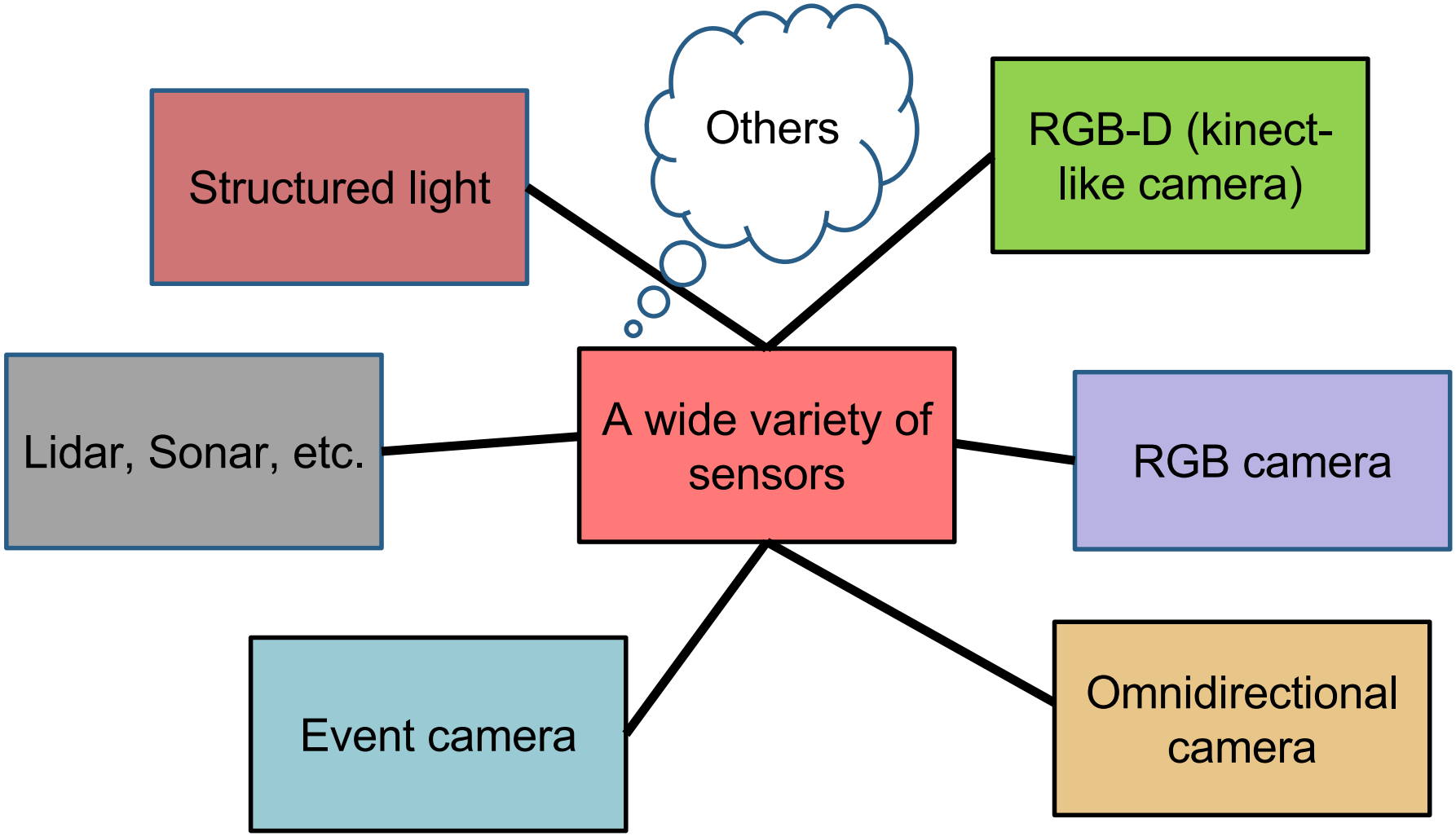
3D Vision
Universitat Pompeu Fabra

Discussion

3D Reconstruction from images?

- Can we obtain 3D information from images?
- Can you see in 3D by using one single eye?

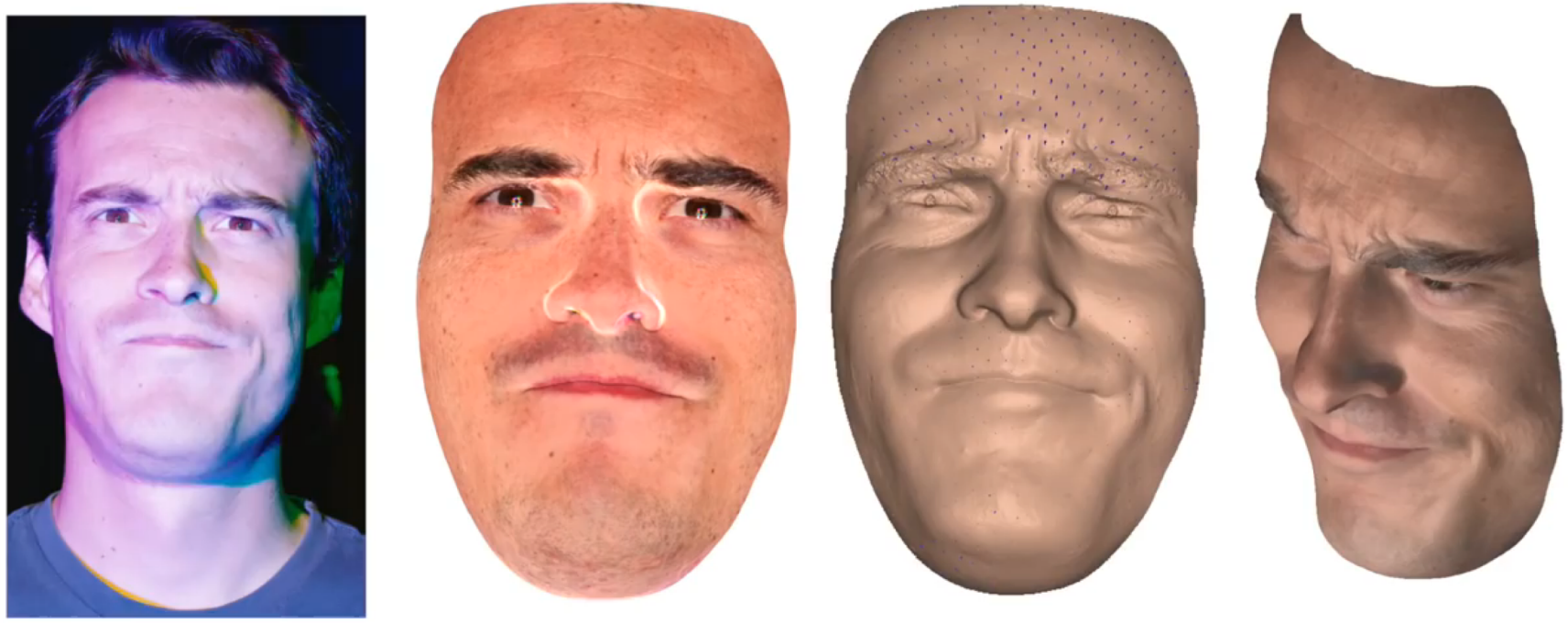
How can I obtained a 3D model?



Light and Vision

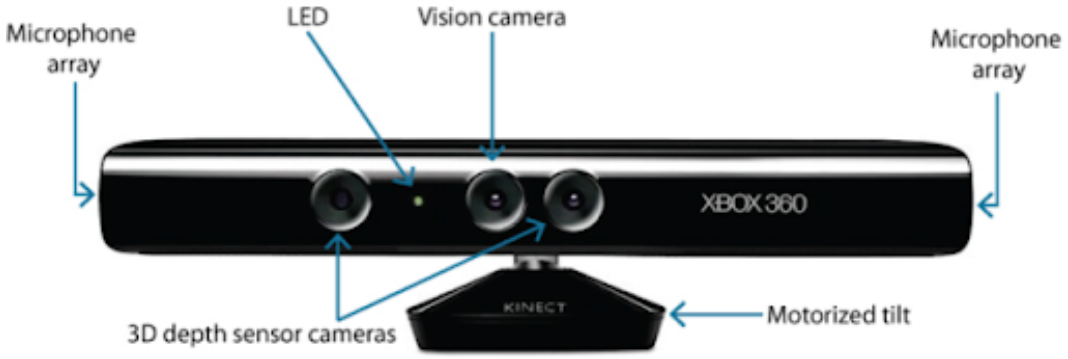
Photogeometric Scene Flow

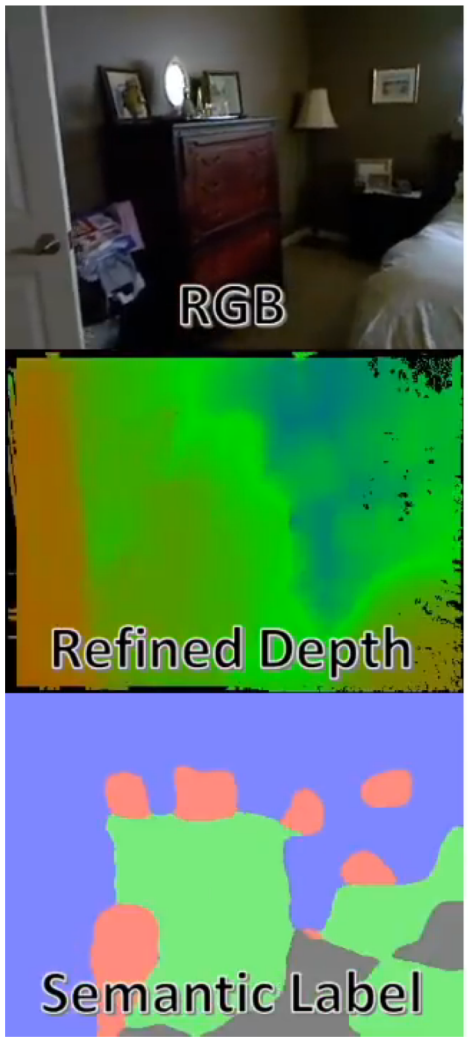
Simultaneous multiview photometric stereo and 3D flow



Input | RGB albedo | 3D surface and motion

RGB-D Camera

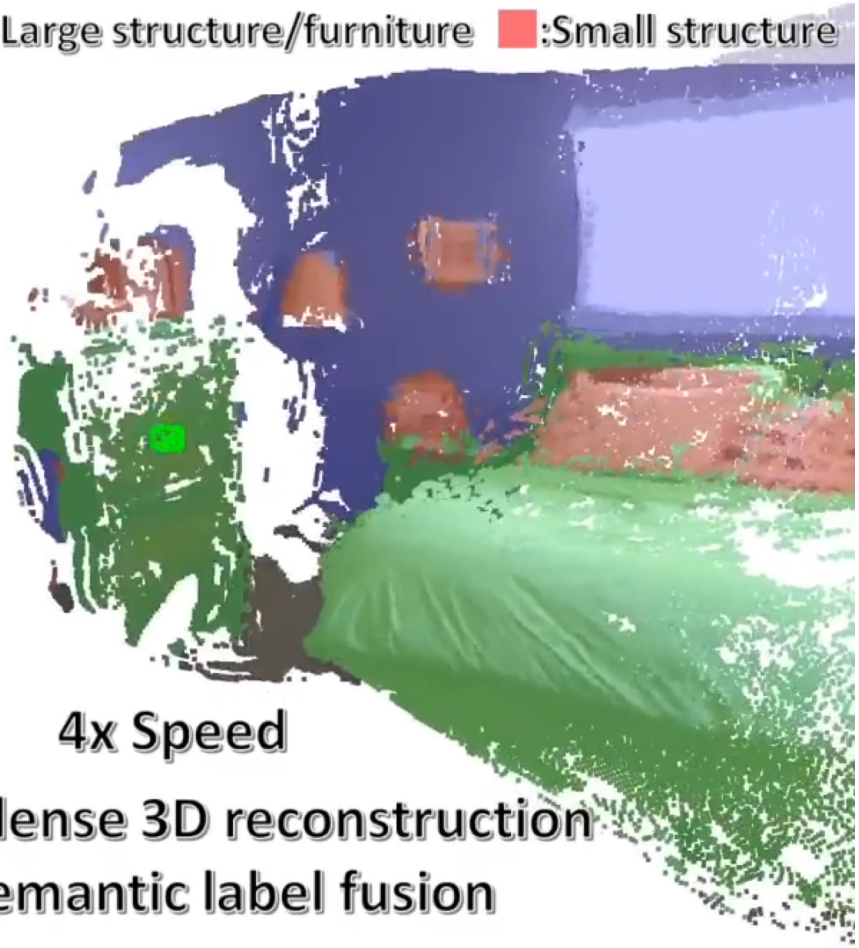




FPS : 32.868420

:Floor :Vertical structure/Wall

:Large structure/furniture :Small structure

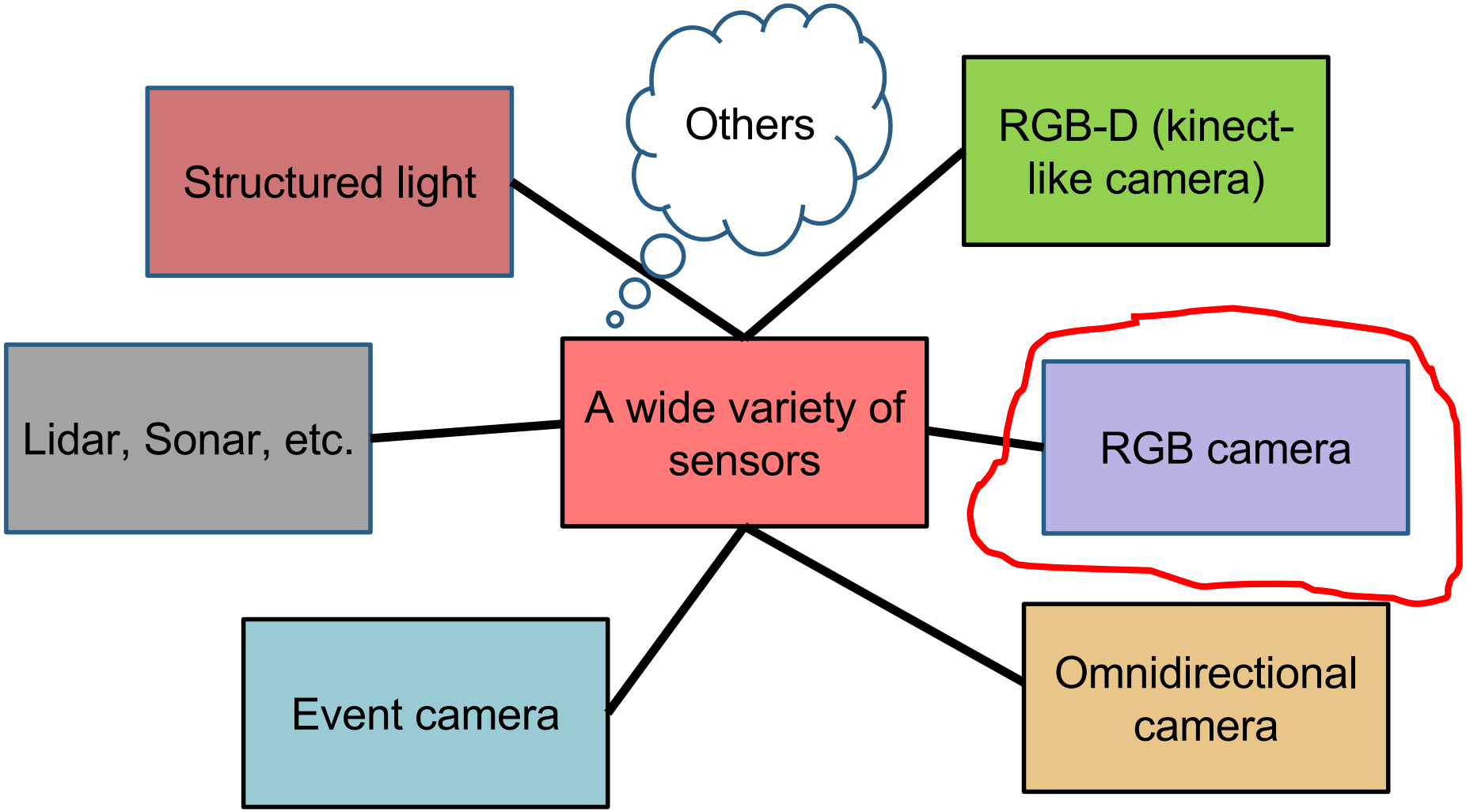


4x Speed

Result of dense 3D reconstruction and semantic label fusion

Camera Tracking with Events

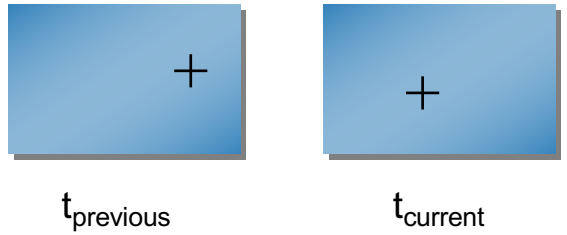
How can I obtained a 3D model?



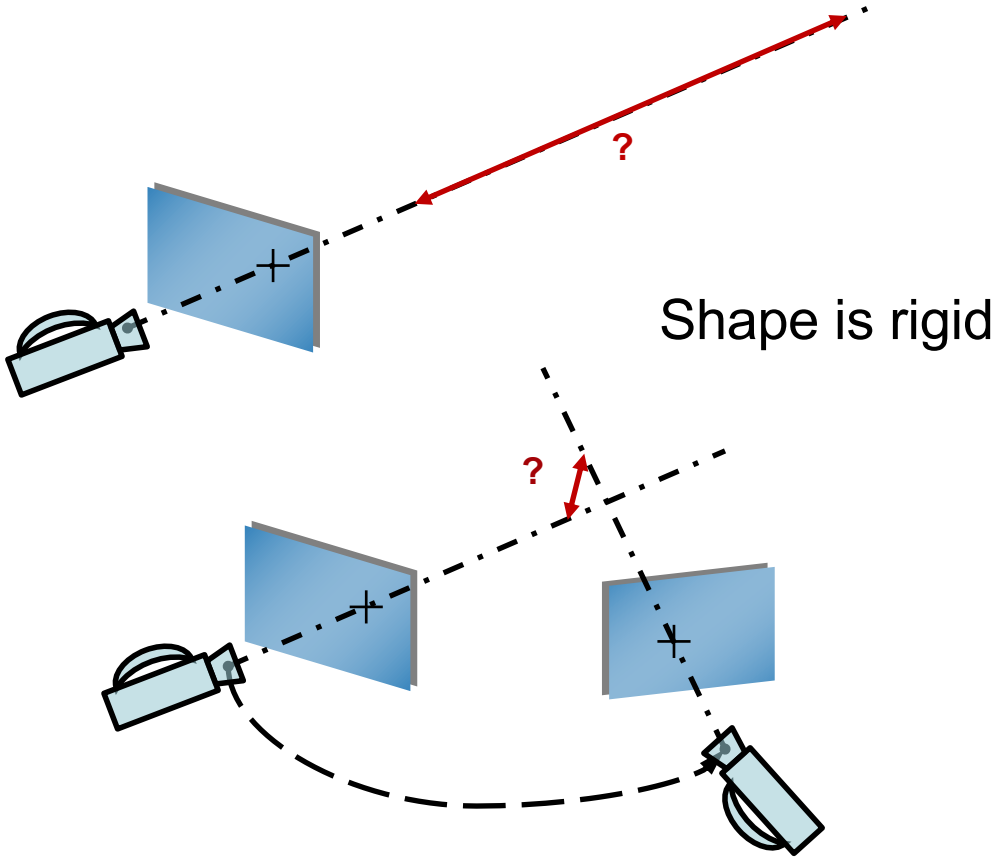
For RGB data
(at the moment, the world is rigid)

A Geometrical Constraint

For every feature, we know a ray but not its depth



$$\mathbf{x} = \mathbf{P}\boldsymbol{X}$$

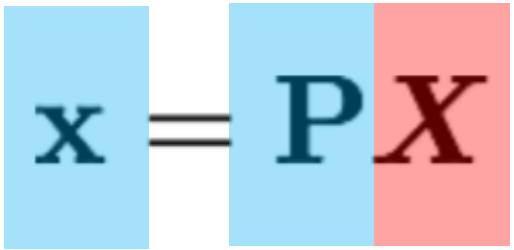


Triangulation vs. Camera Pose

Supposing we know the camera parameters (pose), observing 2D points in the image

How can we compute the 3D location of these points?

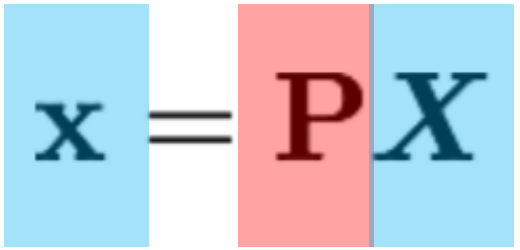
Triangulation



Supposing we know the 3D points, and we also have the matches between these points and the 2D annotations

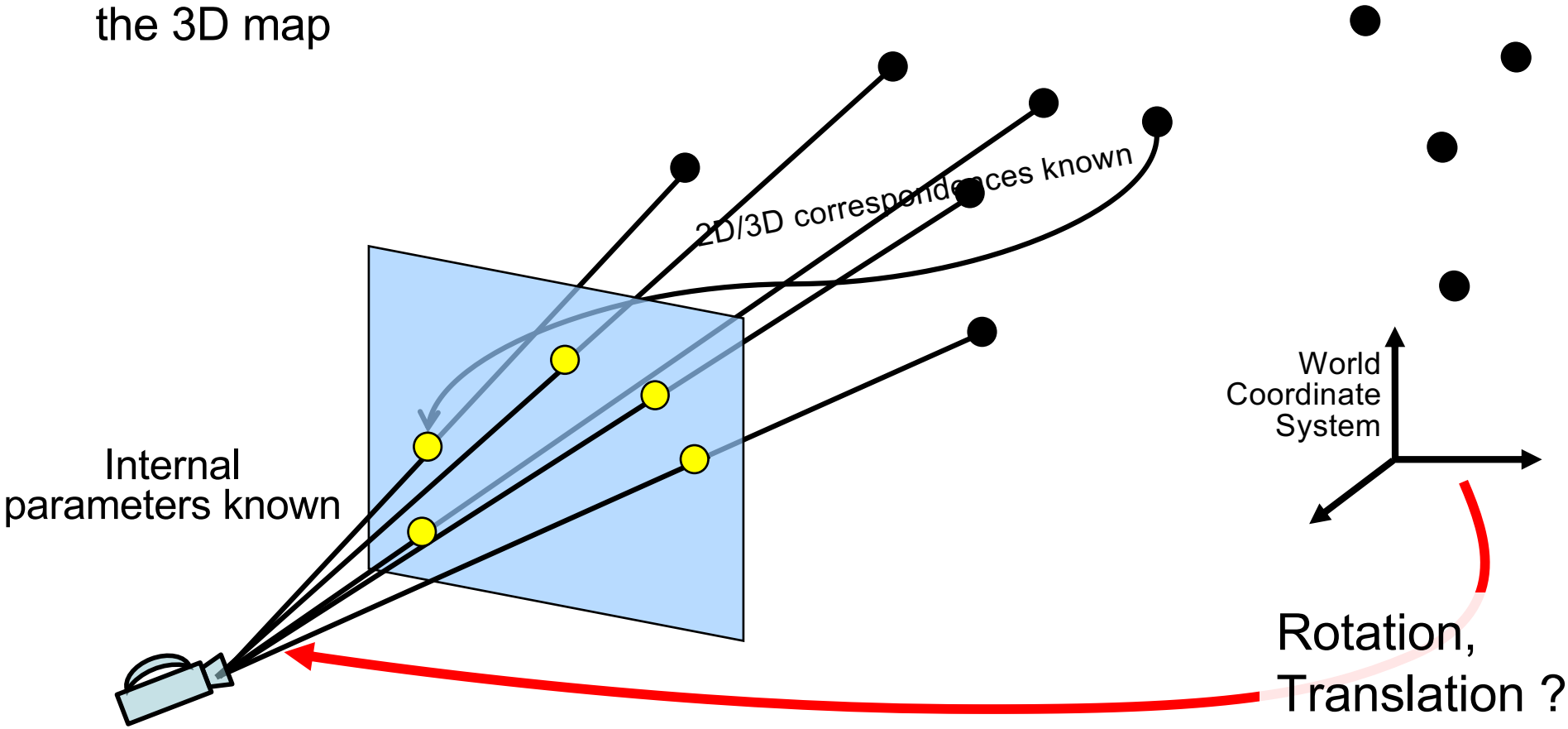
How can we compute the camera parameters?

PnP



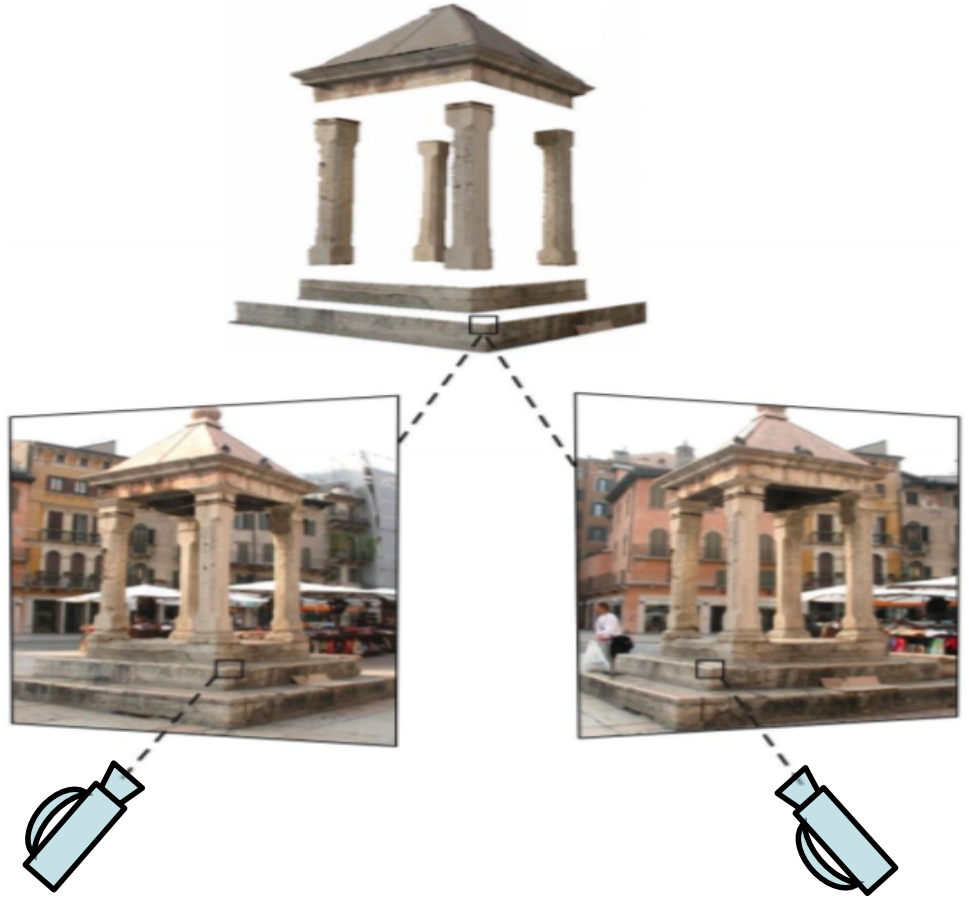
The Perspective-n-Point (PnP) Problem

Given: 3D to 2D correspondences + calibration
We want: Compute the camera pose (rotation and translation) w.r.t. the 3D map

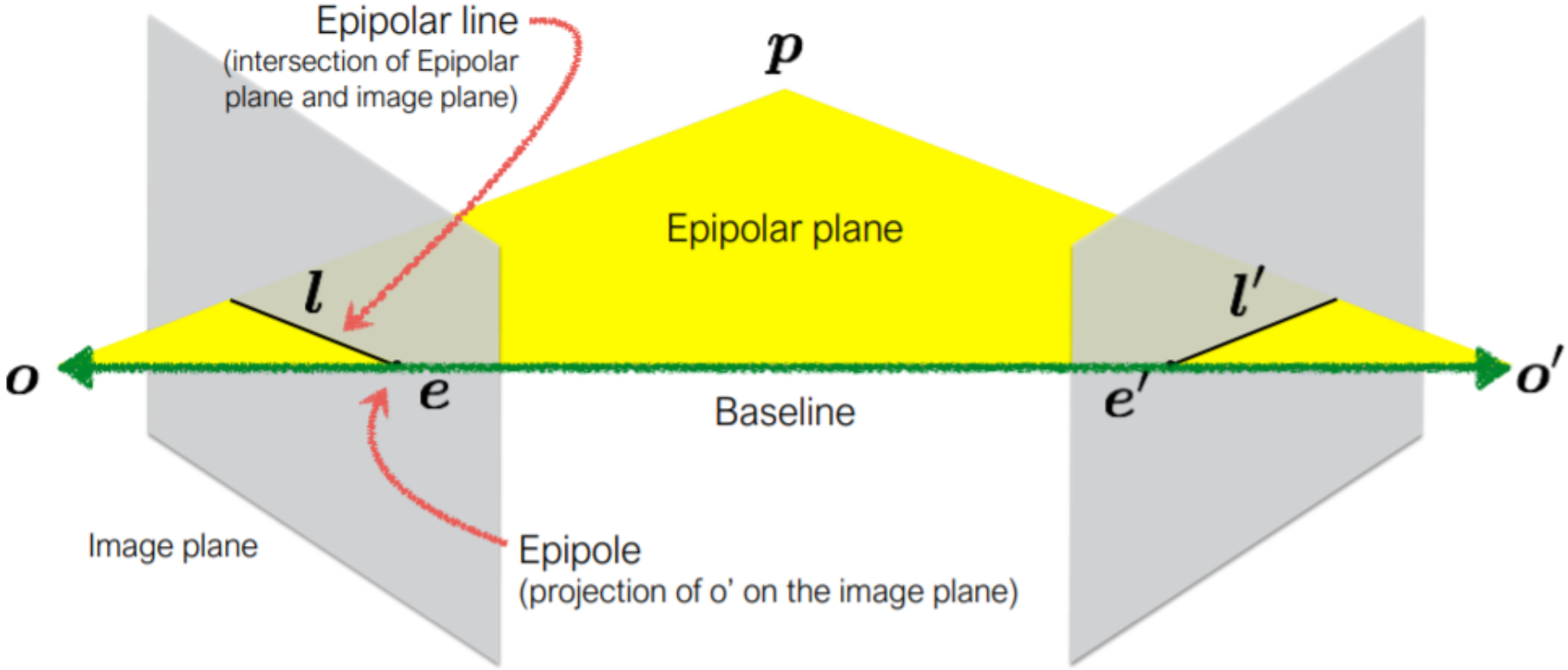


Triangulation

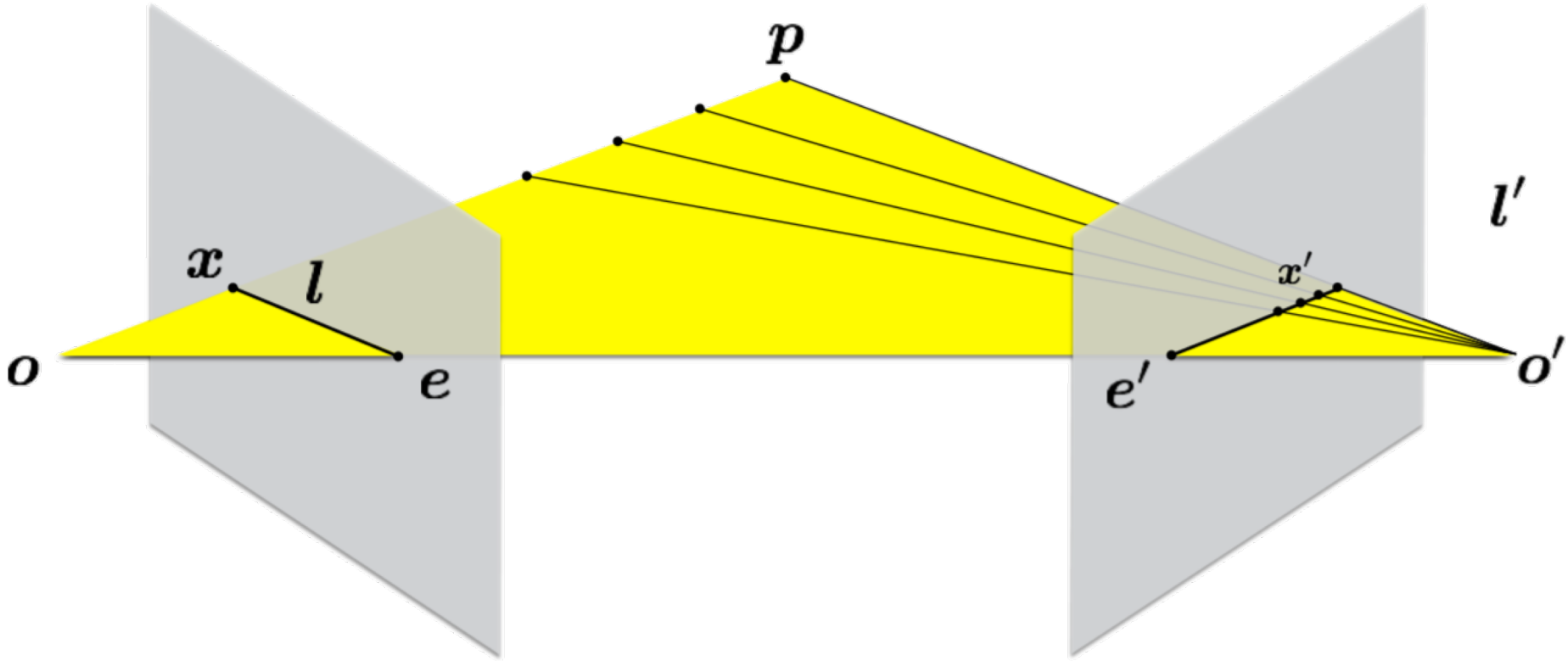
Given: 2D matches and the camera pose
We want: Compute the 3D location of every observed point



Epipolar Constraint

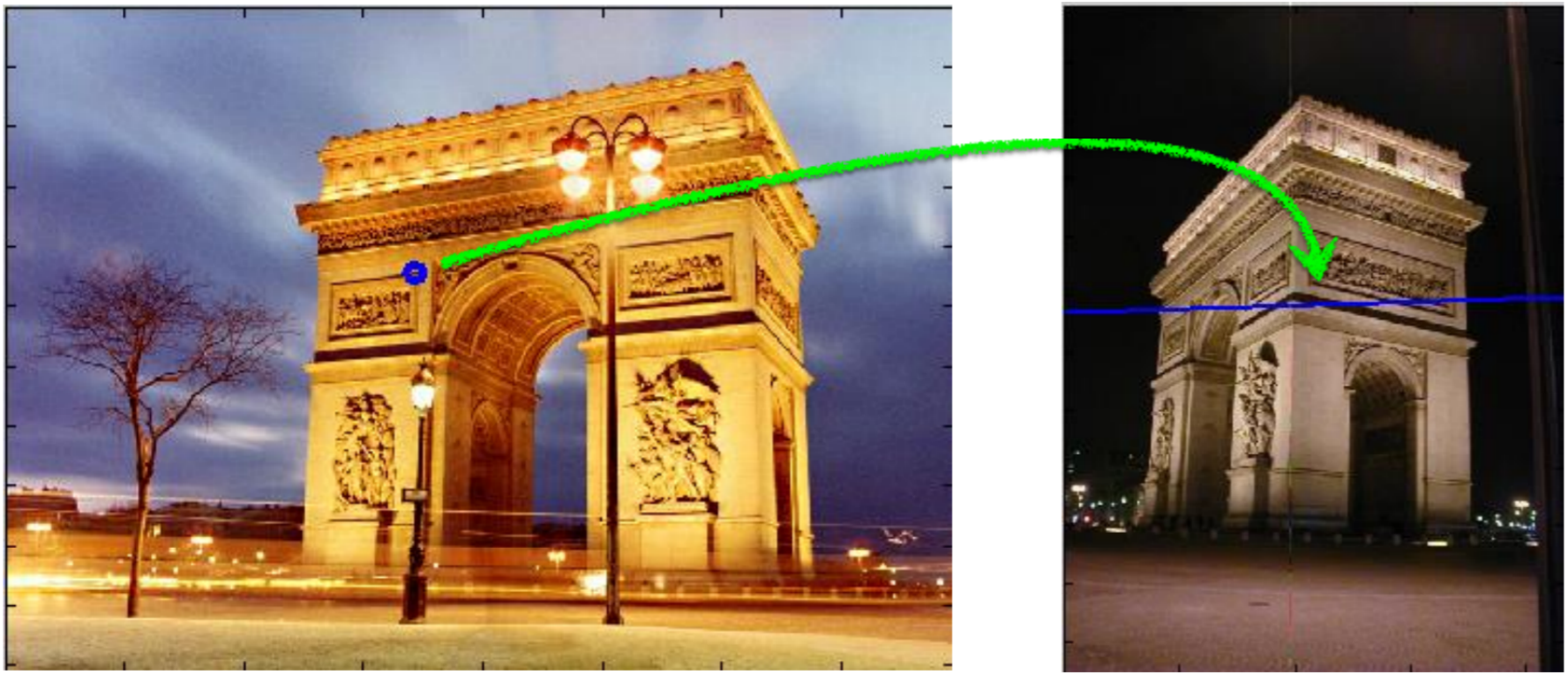


Epipolar Constraint



Potential matches for x lie on the epipolar line l'

Epipolar Constraint



Epipolar Constraint

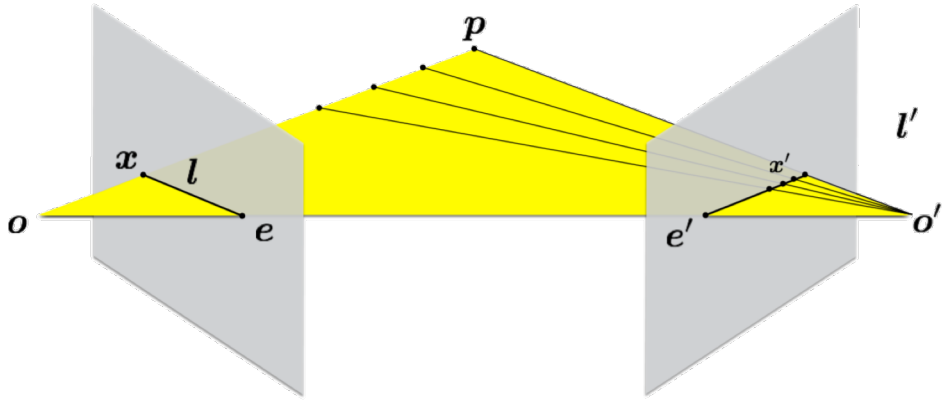
Given a point in one image, multiplying by the essential matrix will tell us the epipolar line in the second view

$$\mathbf{E}\boldsymbol{x} = \boldsymbol{l}'$$

The essential matrix **E** is a 3x3 matrix that encodes the epipolar geometry

To solve, we apply:

$$\boldsymbol{x}'^{\top}\mathbf{E}\boldsymbol{x} = 0$$



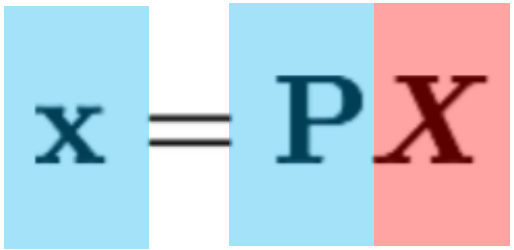
Potential matches for $\boldsymbol{x}$ lie on the epipolar line $\boldsymbol{l}'$

Triangulation vs. Camera Pose

Supposing we know the camera parameters (pose), observing 2D points in the image

How can we compute the 3D location of these points?

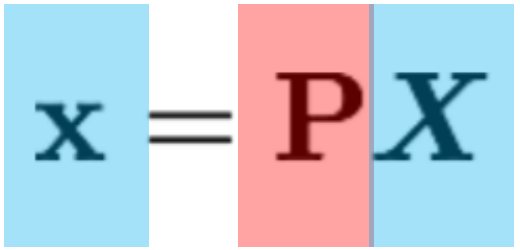
Triangulation



Supposing we know the 3D points, and we also have the matches between these points and the 2D annotations

How can we compute the camera parameters?

PnP



Can we solve both problems at once?

Yes, this problem is denoted as ***Structure from Motion*** (in robotics, the problem is known as ***SLAM***: both are roughly the same)

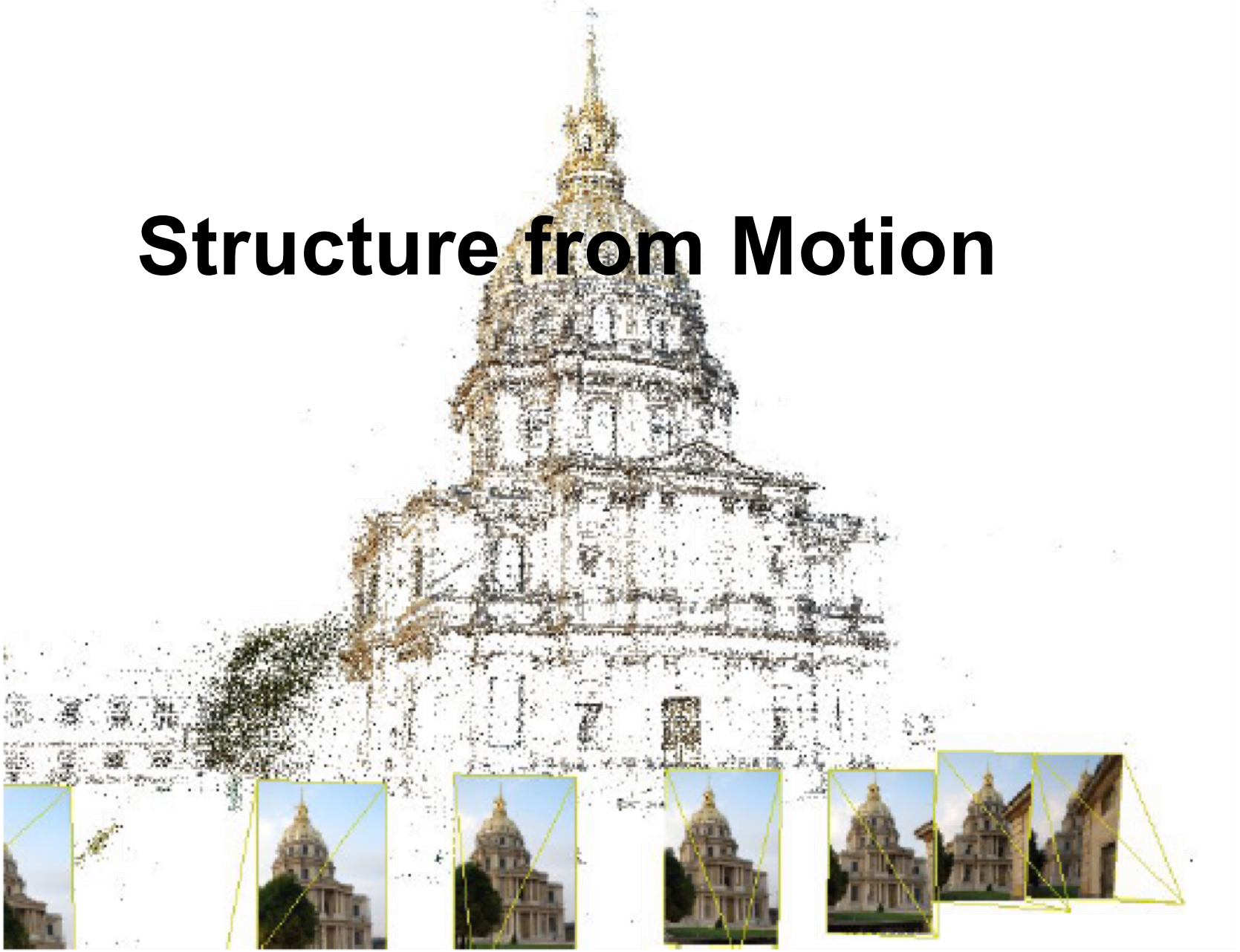
Given a monocular video (or a collection of pictures) acquired with a camera with trajectory unknown, the problem is simultaneously to recover the 3D geometry that the sensor is observing and its motion (6 d.o.f.)

This is a kind of *chicken-and-egg problem*, but fortunately solvable

Only 2D information!

$$\mathbf{x} = \mathbf{P}\boldsymbol{X}$$

Structure from Motion



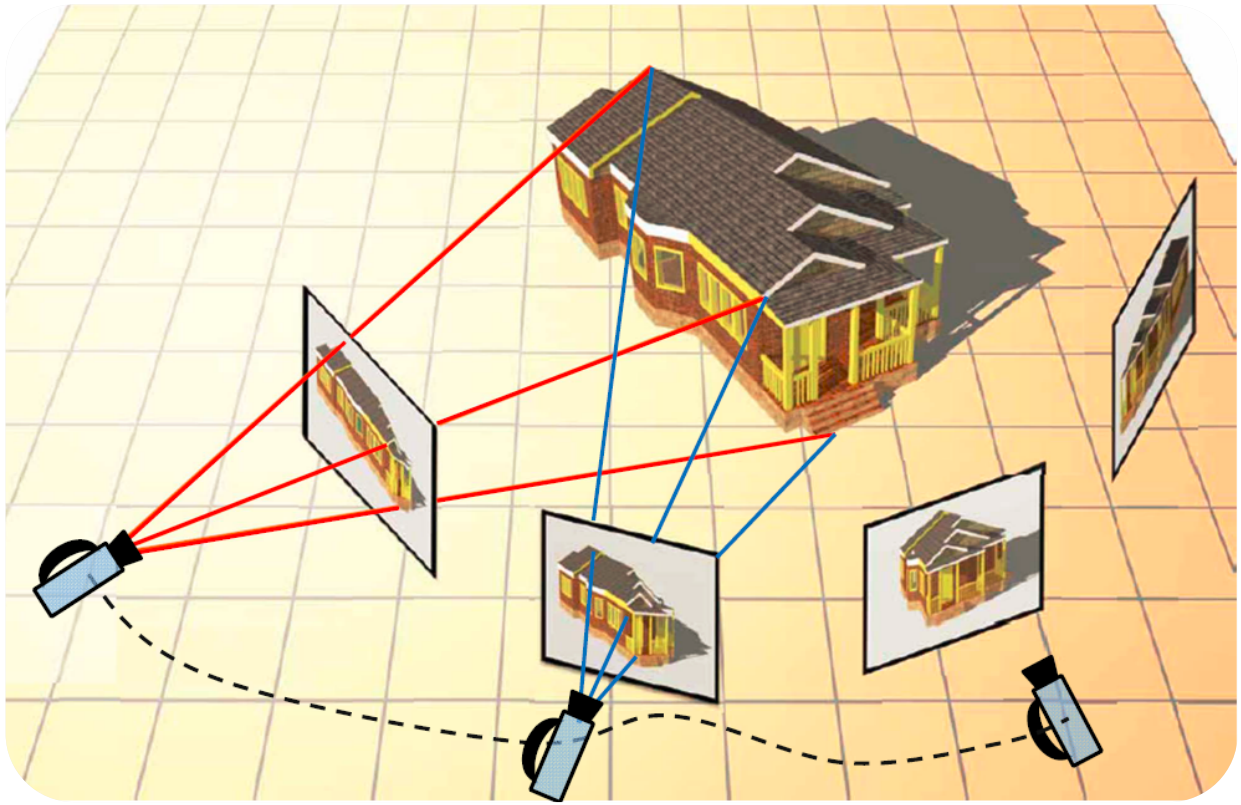
(Rigid) Structure from Motion

Rigid shape
=
same 3D per image

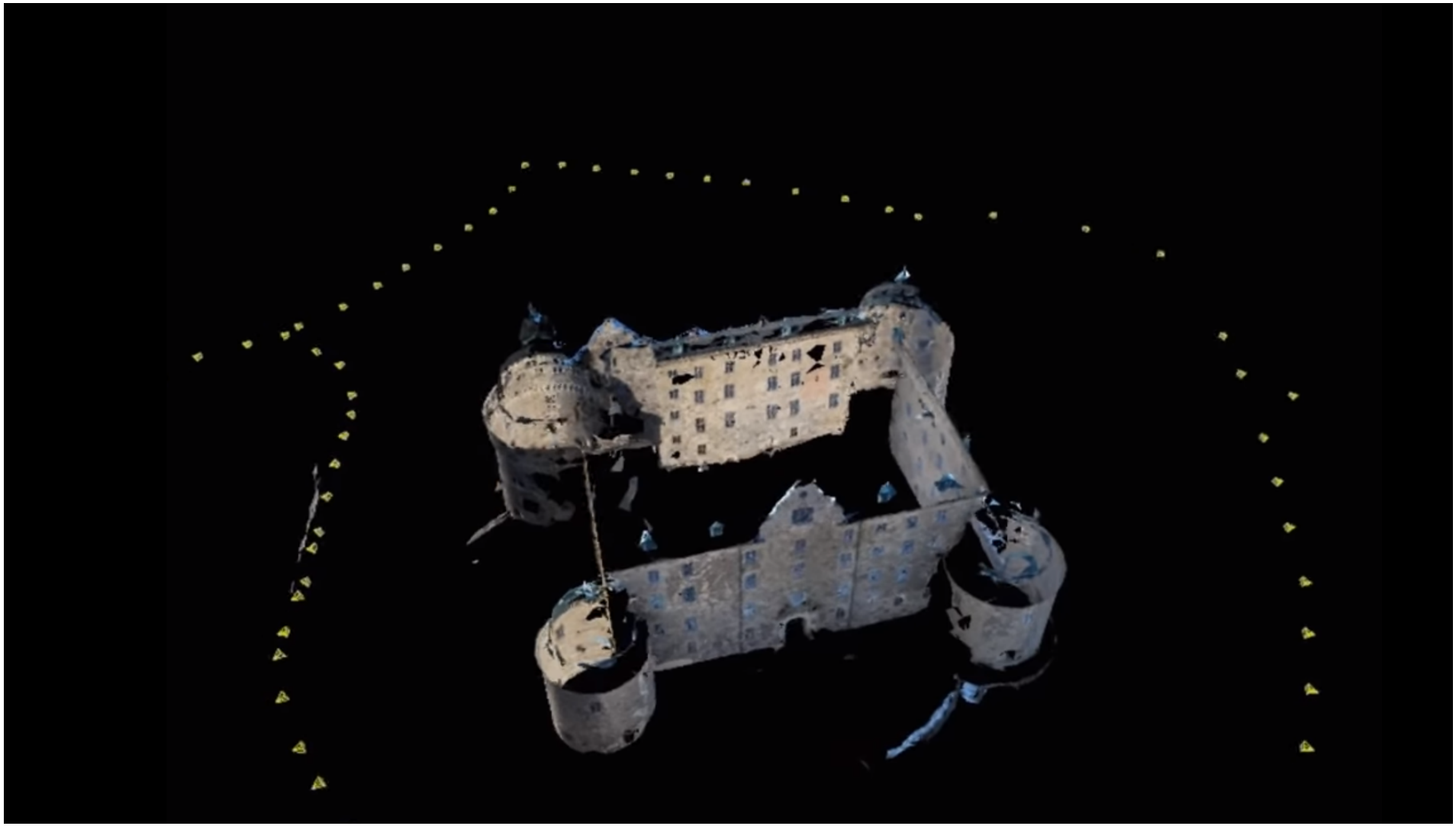
Given: a monocular video (or a collection of pictures)
We want: simultaneously recovering the 3D shape and the camera motion

Epipolar geometry can be used

The assumption of rigidity is enough to make the problem well-posed

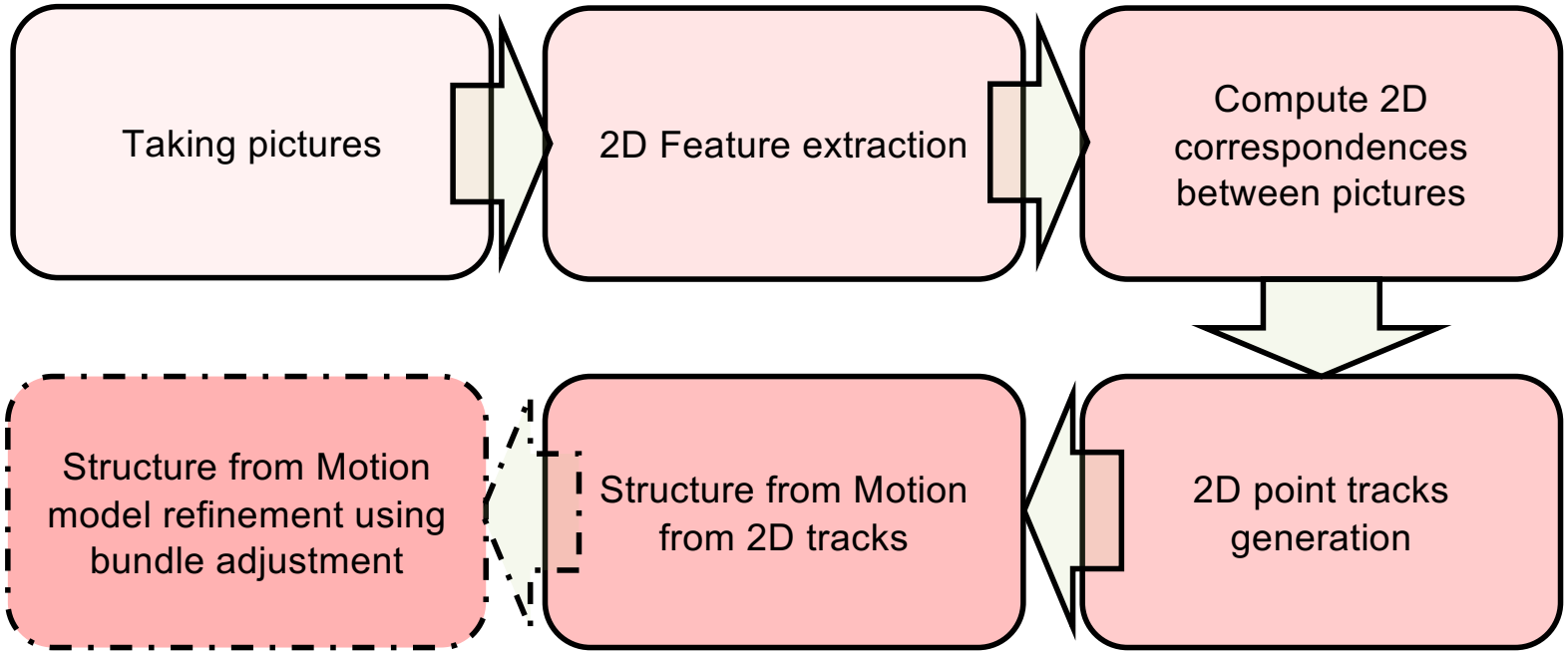


The structure from motion pipeline



https://www.youtube.com/watch?v=i7ierVkXYa8

Structure from Motion Pipeline



(Fine solution)
Non-linear
Optimization

(Coarse solution)
Closed-form solution
by rigid factorization

How do we get correspondences?

- Feature detection and matching
- Graph of Matches
- Using RANSAC to estimate the fundamental matrices between each pair
- *Some patterns are repetitive*. Establishing correspondences can become a hard task

Solving the problem

The problem can be solved by:

- **Factorization**: a closed-form solution can be achieved by using SVD factorization, enforcing a specific rank (this can change as a function of the type of camera model, or the type of scene). In theory, it is hard to accurately enforce constraints
- **Non-linear Optimization**: the solution is achieved iteratively, the computational cost can be bigger but additional priors can be enforced accurately

In terms of processing, the problem can be solved:

- **Offline:** all the frames are processed at once, after video capture
- **Online:** the frames are processed as the data arrive, frame by frame. More real applications, but can become less accurate

Batch vs. Sequential

NO FREE LUNCH

BATCH CASE

✗ Estimation:
After the video capture

✗ Applicability:
Off-line

✓ Accuracy:
Depending on priors

SEQUENTIAL CASE

✓ Estimation:
As the data arrives

✓ Applicability:
On-line and real-time

✗ Accuracy:
Less than batch

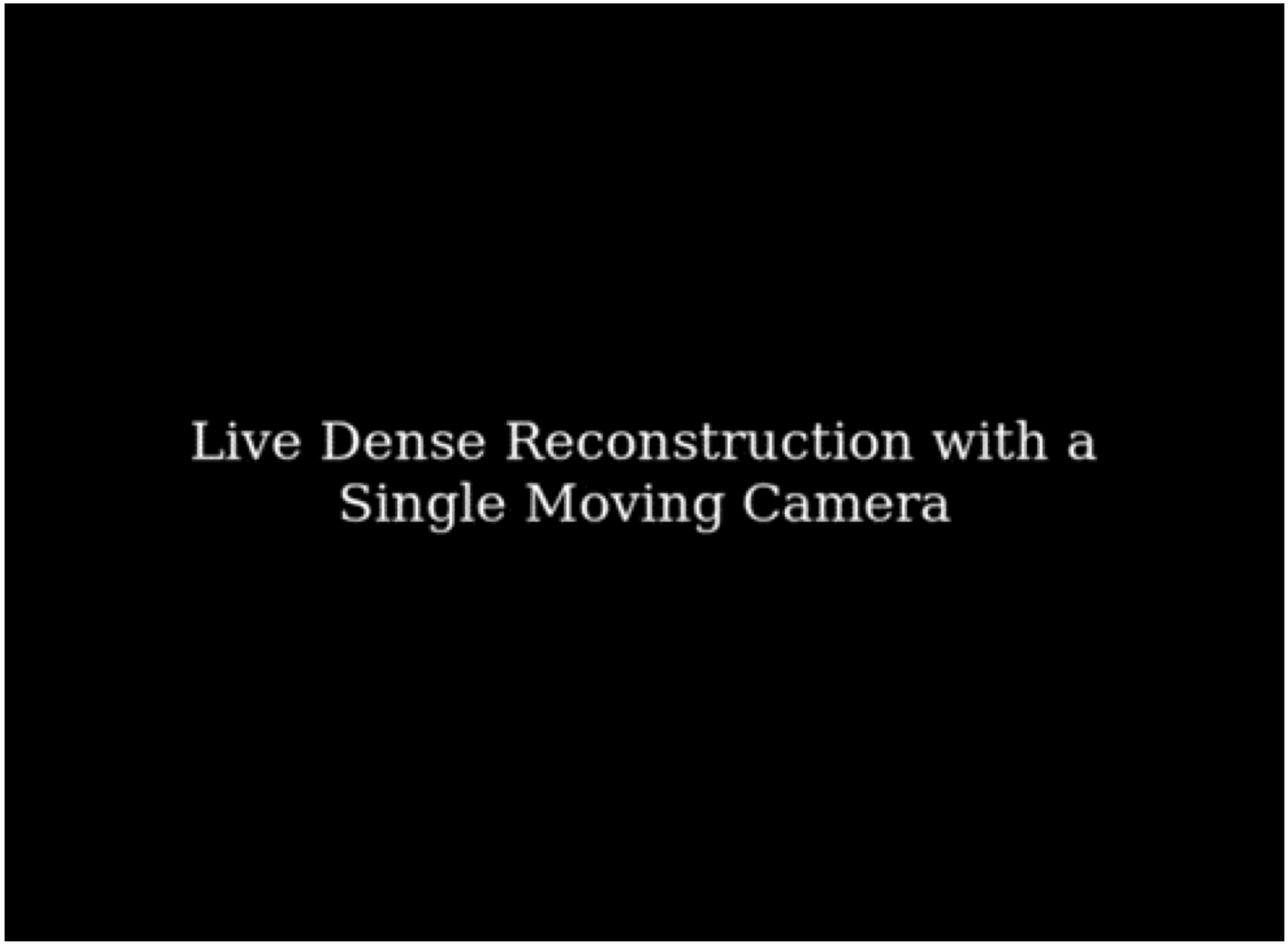
Large Scale Reconstruction

Large Scale Reconstruction

COLMAP Results

Dense Structure from Motion

Dense Structure from Motion

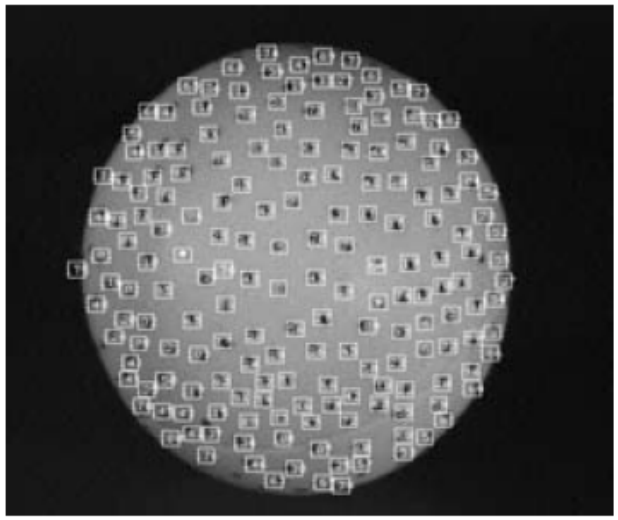


Structure from Motion by Factorization

Factorization from Video Data

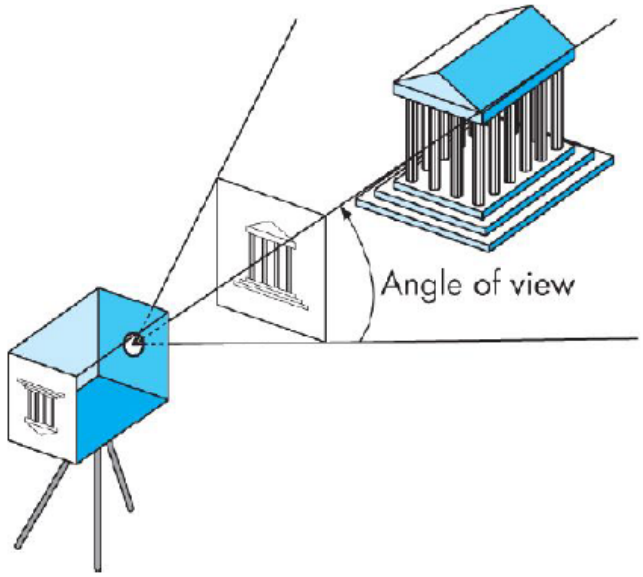
When we have video sequences, we can get feature tracks

Normally, we can reconstruct structure and motion from those tracks using factorization

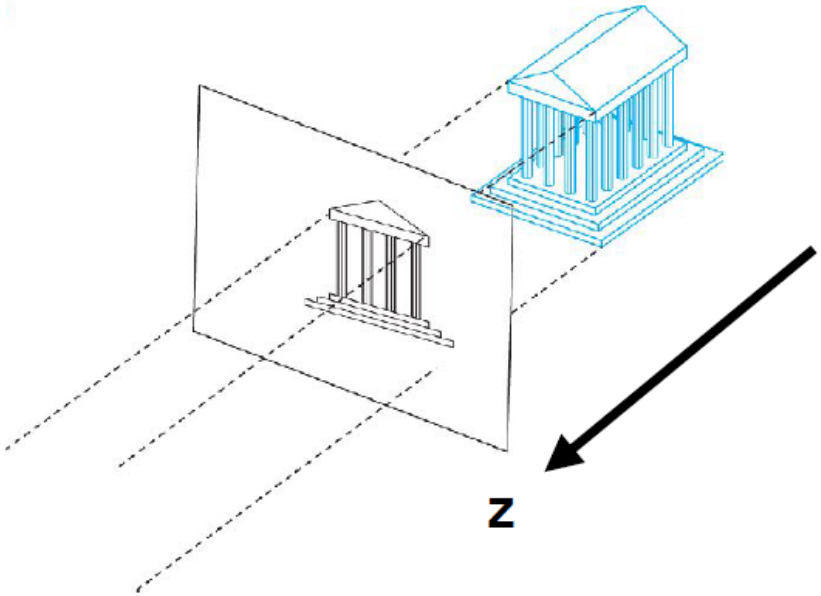


A Reminder of Camera Models

- Perspective camera: All rays converge to the optical center
- Orthographic camera: All rays are parallel. Z-coordinate is irrelevant in the projection



Perspective camera



Orthographic camera

3D-to-2D: Perspective Model

A *p*-th 3D point $\mathbf{X}_p=[X_p, Y_p, Z_p]^T$ in homogeneous coordinates can be related with its 2D projection $\mathbf{x}_p=[x_p, y_p]^T$ by means of a matrix $\mathbf{P}^i$ for the *i*-th image, such as:

$$\begin{bmatrix} \mathbf{x}_p^i \\ 1 \end{bmatrix} = [\mathbf{P}^i] \begin{bmatrix} \mathbf{X}_p \\ 1 \end{bmatrix}$$

where $\mathbf{P}^i$ is 3x4 matrix as:

$$\mathbf{P}^i = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

3D-to-2D: Orthographic Model

A *p*-th 3D point $\mathbf{X}_p=[X_p, Y_p, Z_p]^T$ can be related with its 2D projection $\mathbf{x}_p=[x_p, y_p]^T$ by means of a matrix $\mathbf{R}^i$ for the *i*-th image, such as:

$$\left[\mathbf{x}_p^i\right] = \left[\mathbf{R}^i\right]\left[\mathbf{X}_p\right] + \mathbf{t}_p^i$$

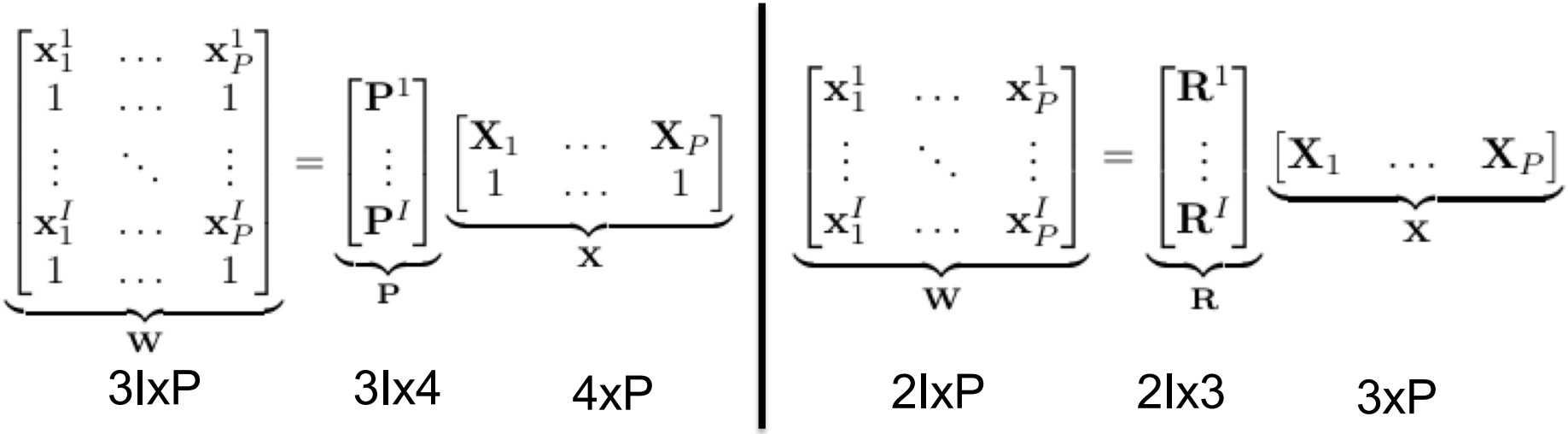
where $\mathbf{R}^i$ is 2x3 matrix and $\mathbf{t}^i$ is a 2x1 translation vector as:

$$\mathbf{R}^i = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{bmatrix} \qquad \mathbf{t}^i = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

In practice, we subtract the translations by assuming centered observations (i.e., they are equivalent to the mean values of $\mathbf{x}_p$). For later computations, we will approximate $\mathbf{x}_p = \mathbf{x}_p - \mathbf{t}^i$

Measurement Matrix

Considering P 3D points observed in I RGB images, we can collect all observations to obtain a linear system such as:



Perspective camera

Orthographic camera

where **W** is a 3IxP matrix, **P** is 3Ix4, and **X** is 4xP for the perspective case (relation on the left); and **W** is a 2IxP matrix, **R** is 2Ix3, and **X** is 3xP for the orthographic case (relation on the right).

A toy example

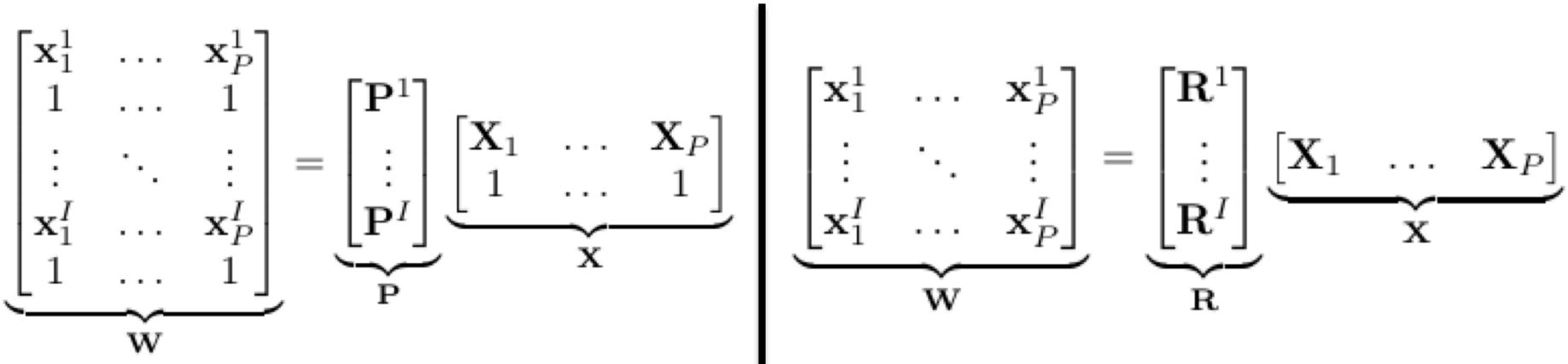
Orthographic camera

$$\begin{pmatrix} \mathbf{x}_1^1 & \mathbf{x}_2^1 & \mathbf{x}_3^1 & \mathbf{x}_4^1 \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \mathbf{x}_3^2 & \mathbf{x}_4^2 \\ \mathbf{x}_1^3 & \mathbf{x}_2^3 & \mathbf{x}_3^3 & \mathbf{x}_4^3 \\ \mathbf{x}_1^4 & \mathbf{x}_2^4 & \mathbf{x}_3^4 & \mathbf{x}_4^4 \end{pmatrix} = \begin{pmatrix} \mathbf{R}^1 \\ \mathbf{R}^2 \\ \mathbf{R}^3 \\ \mathbf{R}^4 \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \mathbf{X}_3 & \mathbf{X}_4 \end{pmatrix}$$

8x4 8x3 3x4

Factorization

In both cases, the goal is to infer the motion factor (**P** or **R**) and the 3D coordinates **X** of the observed object from 2D point tracks in a monocular video **W**



Perspective camera

Orthographic camera

The full linear system

$$W=RX$$

Two factors: motion factor **R** and shape one **X**

More on factorization

Orthographic camera

Because **R** is a 2Ix3 matrix and **X** is a 3xP matrix, the rank of **W** is 3. If we apply SVD to **W**, we will have only ***3 non-zero singular values***

However, measurements are normally noisy, and in practice the rank will not be 3. We have to impose it

Applying SVD factorization, we have:

$$\mathbf{W} = \mathbf{U}\mathbf{A}\mathbf{V}^{T} = [\mathbf{U}\sqrt{\mathbf{A}}][\sqrt{\mathbf{A}}\mathbf{V}^{T}] = [\mathbf{U}\sqrt{\mathbf{A}}\mathbf{Q}][\mathbf{Q}^{-1}\sqrt{\mathbf{A}}\mathbf{V}^{T}]$$

i.e., $\mathbf{R} = \mathbf{U}\sqrt{\mathbf{A}}\mathbf{Q}$ and $\mathbf{X} = \mathbf{Q}^{-1}\sqrt{\mathbf{A}}\mathbf{V}^{T}$ (the two factors we look for)

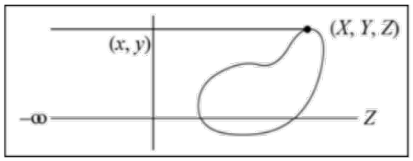
Many solutions can be achieved by modifying **Q**. Of course, for all invertible 3x3 **Q** matrices

Metric Upgrade

How is **Q** computed?

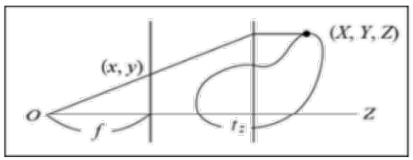
Enforcing orthogonality constraints on the camera rotation. A rotation matrix always has some properties (it is not a random matrix), since lies in the SO(3) manifold. These have to satisfied:

1.Orthographic:



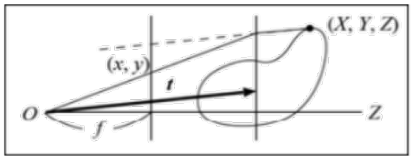
$\mathbf{R}_i$ (2 x 3) $\mathbf{R}_i^T$ (3 x 2) = $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 2x2 identity matrix

2.Weak perspective:



$\mathbf{R}_i$ (2 x 3) $\mathbf{R}_i^T$ (3 x 2) = $\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$ 2x2 diagonal matrix

3.Para perspective:



$\mathbf{R}_i$ (2 x 3) $\mathbf{R}_i^T$ (3 x 2) = $\begin{bmatrix} a & c \\ c & b \end{bmatrix}$ 2 x2 full matrix

What about the perspective case?

You have two possibilities:

- Assuming an orthographic camera, solve for it and then iterate to be perspective
- Applying a SVD factorization enforcing a rank 4 over **W**

But in many cases, we cannot observe all the points in all the images

==

Missing tracks

A toy example with missing tracks

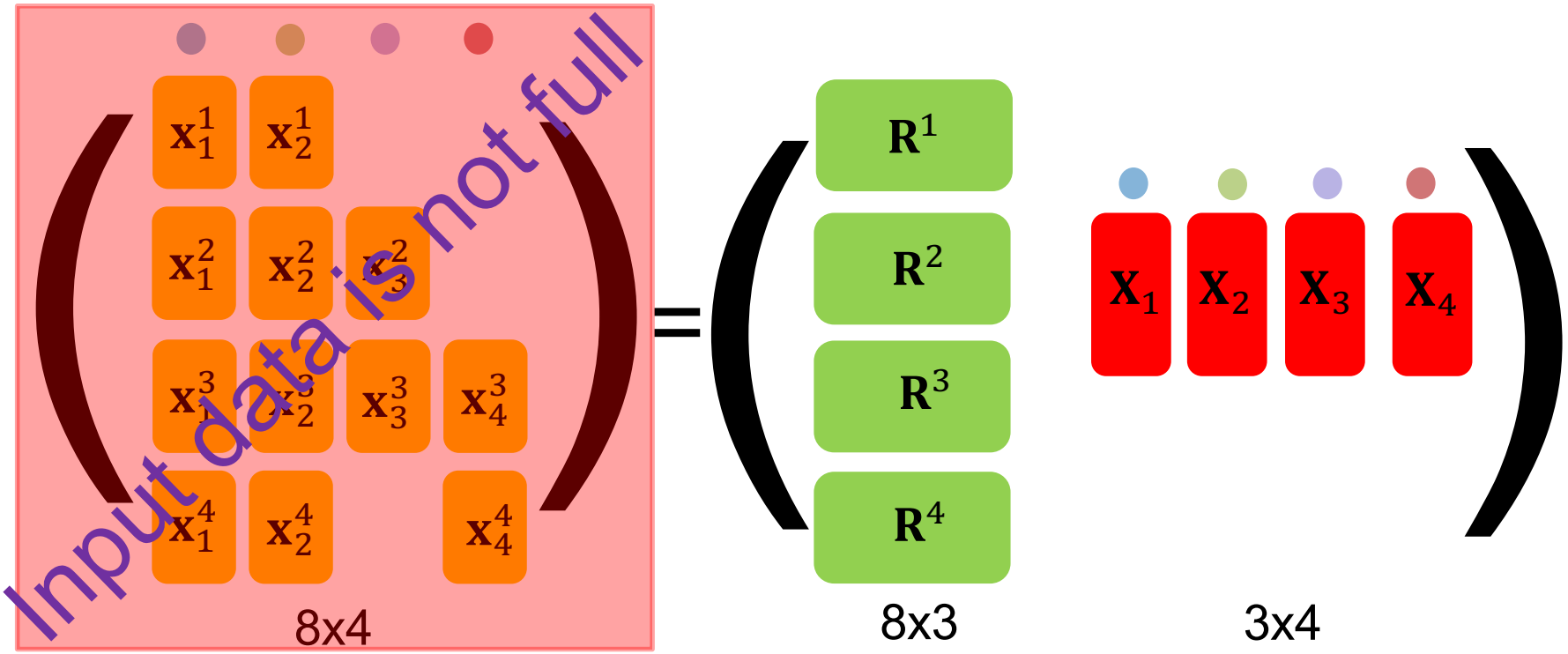
Orthographic camera

$$\begin{pmatrix} \mathbf{x}_1^1 & \mathbf{x}_2^1 & & \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \mathbf{x}_3^2 & \\ \mathbf{x}_1^3 & \mathbf{x}_2^3 & \mathbf{x}_3^3 & \mathbf{x}_4^3 \\ \mathbf{x}_1^4 & \mathbf{x}_2^4 & & \mathbf{x}_4^4 \end{pmatrix} = \begin{pmatrix} \mathbf{R}^1 \\ \mathbf{R}^2 \\ \mathbf{R}^3 \\ \mathbf{R}^4 \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \mathbf{X}_3 & \mathbf{X}_4 \end{pmatrix}$$

8x4 8x3 3x4

A toy example with missing tracks

Orthographic camera



Handling missing tracks

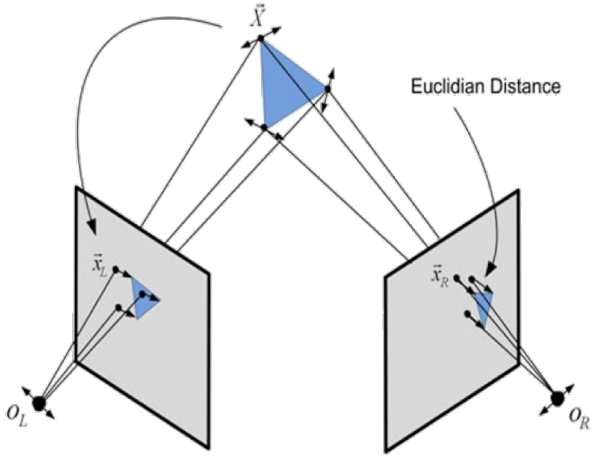
Two alternatives are possible:

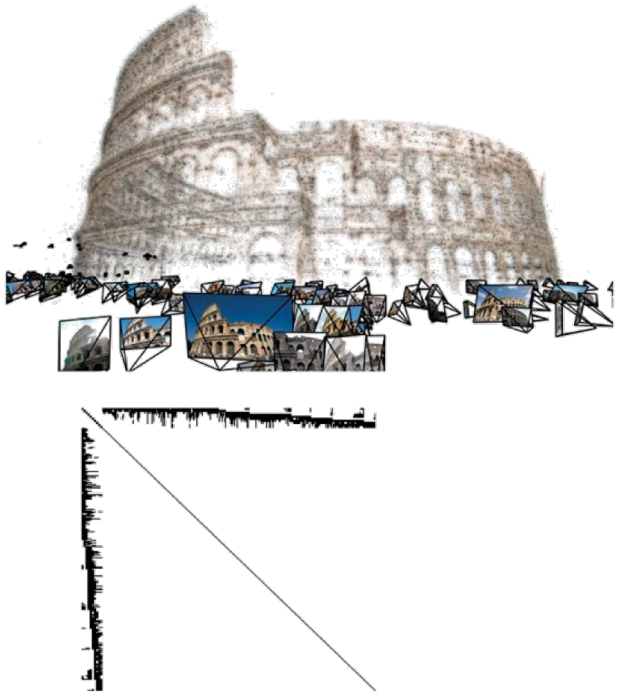
- Applying a *matrix completion* algorithm to infer the missing entries, and then run factorization over the full measurement matrix
- No consider missing entries in the formulation by applying non-linear optimization. Once the 3D model and camera pose are computed, the 2D missing tracks can be inferred too

Structure from Motion
by
Non-linear Optimization

Bundle Adjustment

As the image measurements are noisy, then the projection equation for an arbitrary point is not satisfied exactly





We seek for factor **R** and **X** such that the projection after applying the model is as much close as possible to the real observation

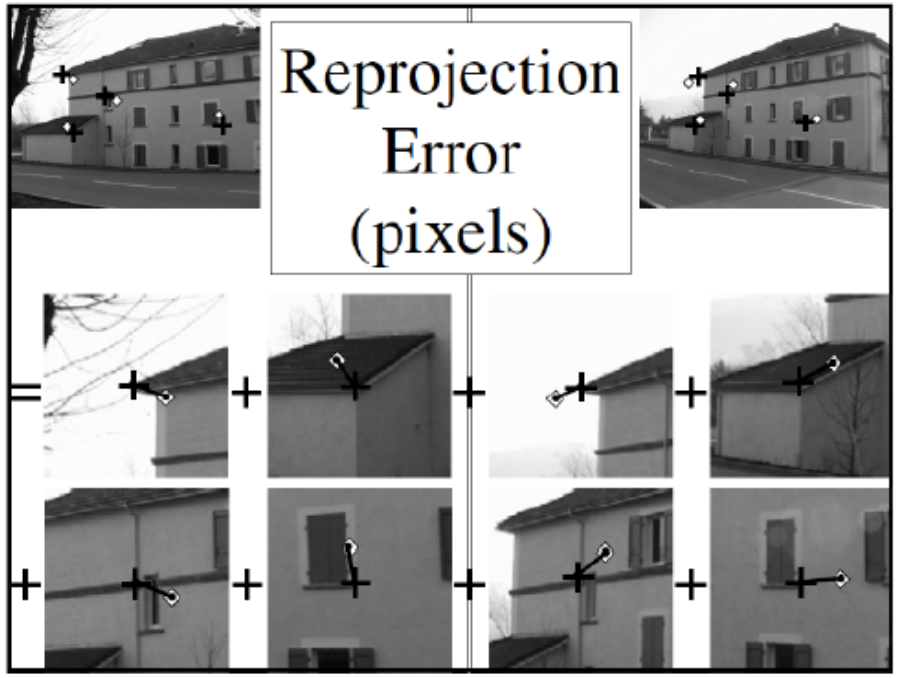
Bundle Adjustment

The strategy consist in minimizing the re-projection error as:

$$\min_{\mathbf{R}^i, \mathbf{X}_p} \sum_{i=1}^{i=I} \sum_{p=1}^{p=P} d(\mathbf{R}^i \mathbf{X}_p, \hat{\mathbf{x}}_p^i)^2$$

where we recover $\mathbf{R}^i$ for the I images and $\mathbf{X}_p$ for the P points.

When the noise follows a Gaussian distribution, minimizing this Expression we are looking for the Maximum likelihood estimator



Bundle Adjustment

This estimation is known as *bundle adjustment* in literature, since it involves adjusting the bundle of rays between each camera and the set of 3D points. It is generally used as a final step for any reconstruction algorithm

The good points are:

- Missing data can be handled
- Robust to noisy observations
- Additional priors can be easily included and imposed

The bad points are:

- Non-convex problem, it requires a good initialization
- It can become a large minimization problem because of the number of parameters involved (large scale problems)

Bundle Adjustment

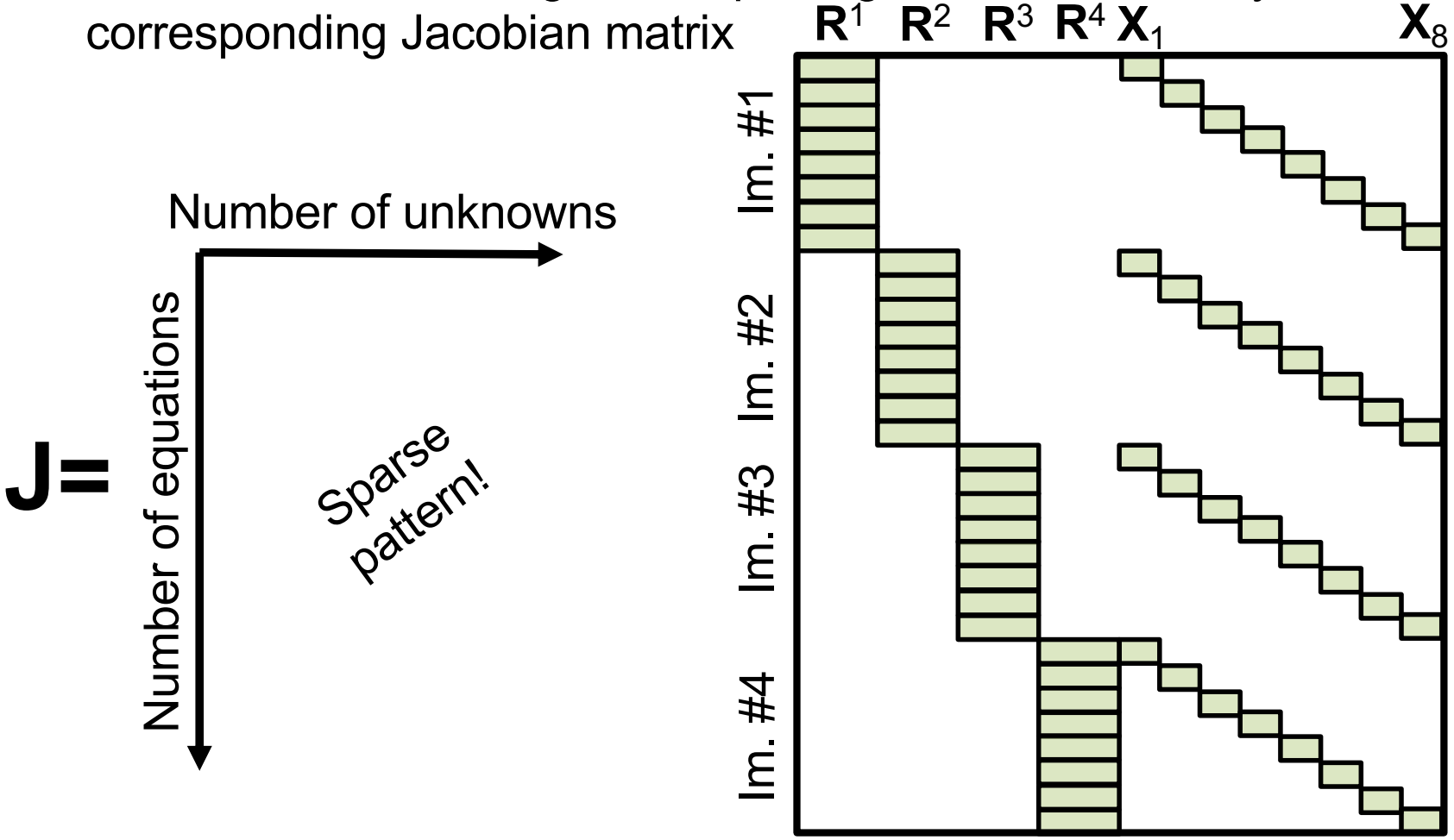
Normally, the Levenberg-Marquardt method is used to minimize the problem. We need a Jacobian matrix **J** as the derivative of the function with respect to the unknowns (**R** and **X**)

There are many variants on how to proceed to reduce the computational complexity of the problem:

- No including all the views or the points, and fill these in later by re-sectioning or triangulation, respectively
- Partition the data into sets, bundle adjust every set separately and then merge
- Alternate minimization of motion and shape parameters
- *Sparse methods*. The computation of **J** is complex, but it can be approximated by considering a *binary pattern*

Jacobian matrix sparseness

Let us assume a collection of 4 RGB images, where 8 points are observed. Considering the map is rigid and full visibility, define the corresponding Jacobian matrix



Including priors

The most direct prior we can use in rigid structure from motion is a *temporal smoothness* one. As the shape is rigid, we only apply this over the camera motion (of course, for a collection of images the prior can fail). In this case, we have to solve:

$$\min_{\mathbf{R}^i,\mathbf{X}_p} \sum_{i=1}^{i=I} \sum_{p=1}^{p=P} d(\mathbf{R}^i\mathbf{X}_p, \hat{\mathbf{x}}_p^i)^2 + \phi \sum_{i=1}^{I-1} \|\mathbf{R}^i - \mathbf{R}^{i+1}\|_{\mathcal{F}}^2$$

In this particular example, we consider a *first-order filter* (2 frames are considered). Other *high-order filters* can be applied, e.g., a second-order filter by considering three frames and so on

$$+ \sum_{i=2}^{I-1} \|2\mathbf{R}^i - \mathbf{R}^{i+1} - \mathbf{R}^{i-1}\|_{\mathcal{F}}^2 \; ?$$

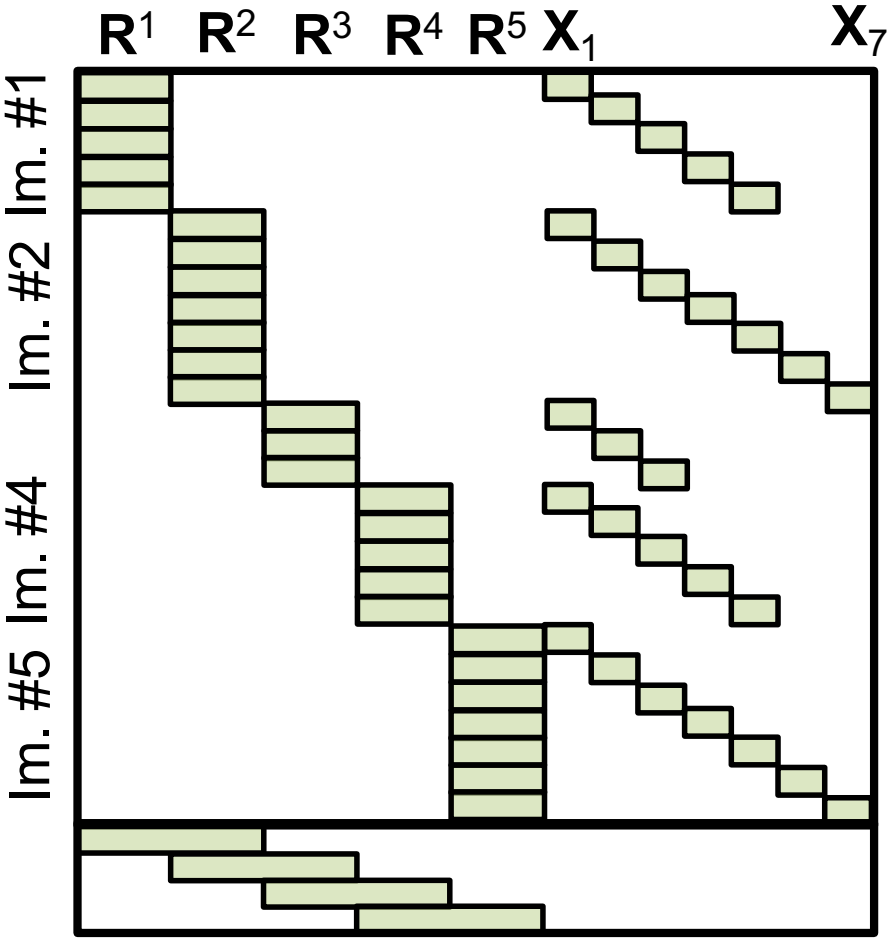
Exercise

Let us assume a monocular video of 5 RGB images, where 7 points are observed. Considering the map is rigid and visibility is {5, 7, 3, 5, 7}, define the corresponding Jacobian matrix. Could we include any smoothness prior? How could that be done?

Note: in all cases, the visible points are considered to have the smaller indices in the map.

Exercise

Let us assume a monocular video of 5 RGB images, where 7 points are observed. Considering the map is rigid and visibility is {5, 7, 3, 5, 7}, define the corresponding Jacobian matrix. Could we include any smoothness prior? How could that be done?



More equations!

How can we obtain a sequential solution?

We solve the optimization in a sequential manner, considering the information as the data arrives. Future frames are not available. Two options:

- Pure sequential (frame by frame)
- Mini-batch:
  - Sliding window (from 3 to 5 consecutive frames)
  - Selecting key-frames (some relevant frames)

Initialization is performed by factorization (assuming just the initial frames), or including some parametrizations (inverse depth in filtering approaches)

More extensions: Self-Calibration

Self-Calibration

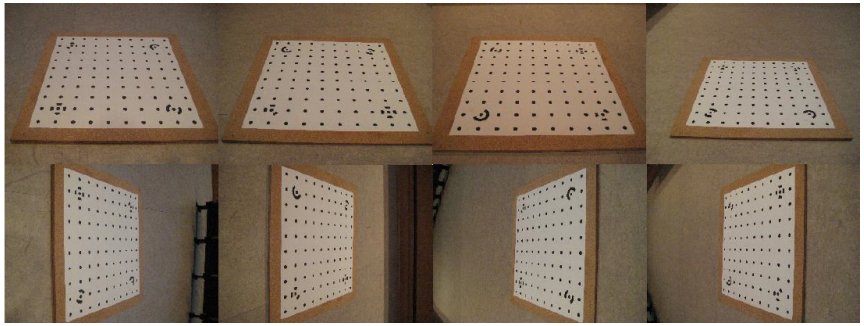
Given: a monocular video (or a collection of pictures)
We want: simultaneously recovering the 3D shape, the camera motion, and the calibration of the camera

Self-Calibration

Normally, the parameters of the camera are estimated off-line, before acquiring a video using a calibration pattern

However, these parameters can change due to:

- Vibrations
- Thermal shocks
- Mechanical shocks
- Zooming
- …



Self-calibration allows the computation from scratch of some projective parameters: focal length, principal point, skew… and sometimes, distortion parameters are also included

Be careful. For some motions, the problem cannot be solved (pure translation over optical axis, etc.)

3D-to-2D: Perspective Model

A *p*-th 3D point $\mathbf{X}_p=[X_p, Y_p, Z_p]^T$ in homogeneous coordinates can be related with its 2D projection $\mathbf{x}_p=[x_p, y_p]^T$ by means of a matrix $\mathbf{P}^i$ for the *i*-th image, such as:

$$\begin{bmatrix} \mathbf{x}_p^i \\ 1 \end{bmatrix} = [\mathbf{P}^i] \begin{bmatrix} \mathbf{X}_p \\ 1 \end{bmatrix}$$

Intrinsic parameters are estimated on the fly

where $\mathbf{P}^i$ is 3x4 matrix as:

$$\mathbf{P}^i = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

What about learning?

Learning-based approaches can be used to solve the problem. Mainly, two possibilities:

- Deep supervised learning
- Deep self-supervised learning

In the first case, we need 3D ground truth to learn a model. After learning, no guaranty in accuracy can be assumed (this depends on the samples in the dataset we use in training)

In the second case, we can exploit the theory in this lecture to define new losses. For example, the re-projection is employed for self-supervision without assuming any 3D data

Shape and motion are estimated in different networks

Things to remember

3D information can be obtained from a sequence of RGB images without assuming any training data

For rigid objects, the problem is well-posed. A geometric analysis of the problem allows us finding a rank constraint

Coarse solutions can be obtained by factorization. After that, we can refine the solution by applying optimization

More sophisticated priors can be included in optimization, providing even more accurate and robust solutions

The final goal is, with a hand-held camera, to estimate every scenario, preferably, in real time at frame rate

Acknowledgments

Thanks to Kris Kitani, Davide Scaramuzza, Adrien Bartoli, and Raquel Urtasun