Hands on AI based 3D Vision Summer Semester 25

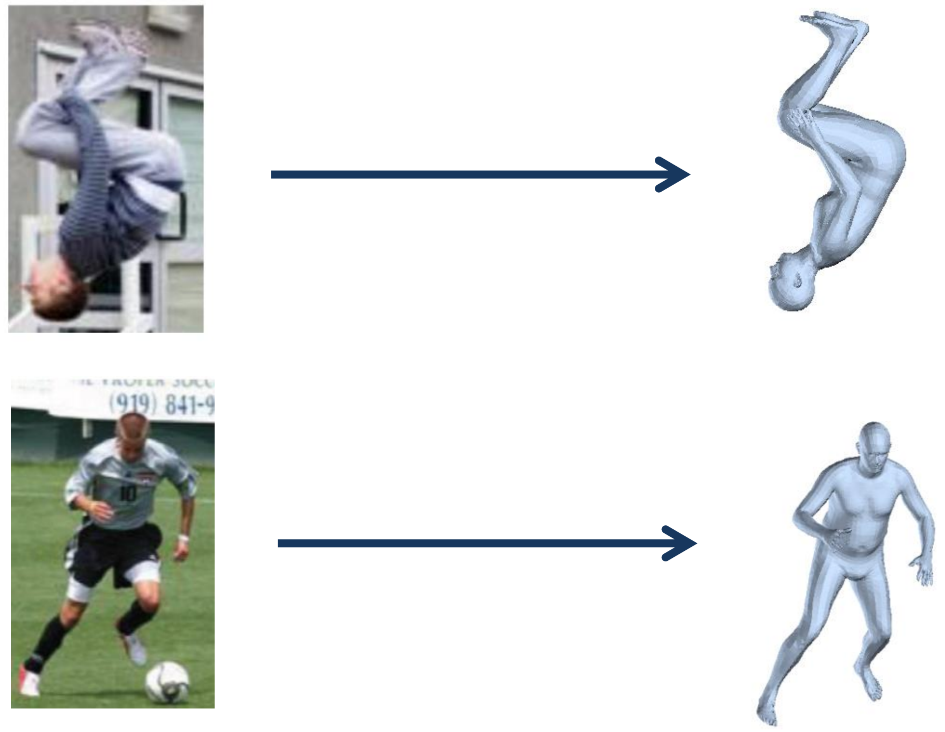
Lecture 3 – Classical Reconstruction Methods

Prof. Dr.-Ing. Gerard Pons-Moll

University of Tübingen / MPI-Informatics

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

"Inverting" the Image Formation Process



Input (2D) Output (3D)

Overview

Introduction

The Structure from Motion Problem

Given large amounts of images of a scene, we want to simultaneously

- Find out where they were taken from
- Build a (sparse) 3D model of the scene

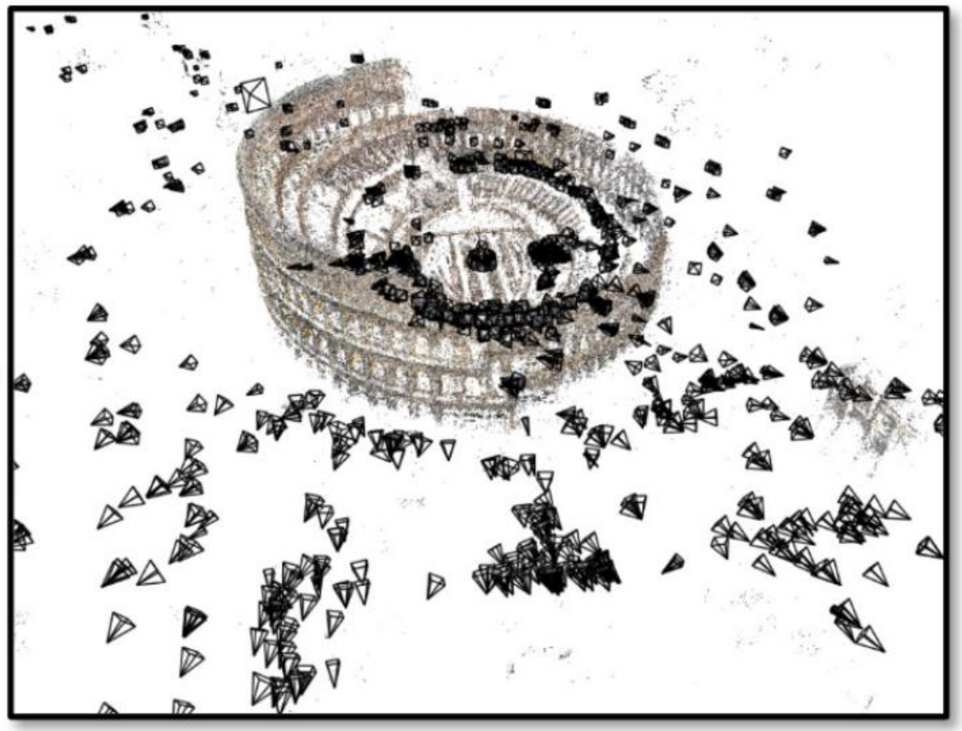


Photo Tourism

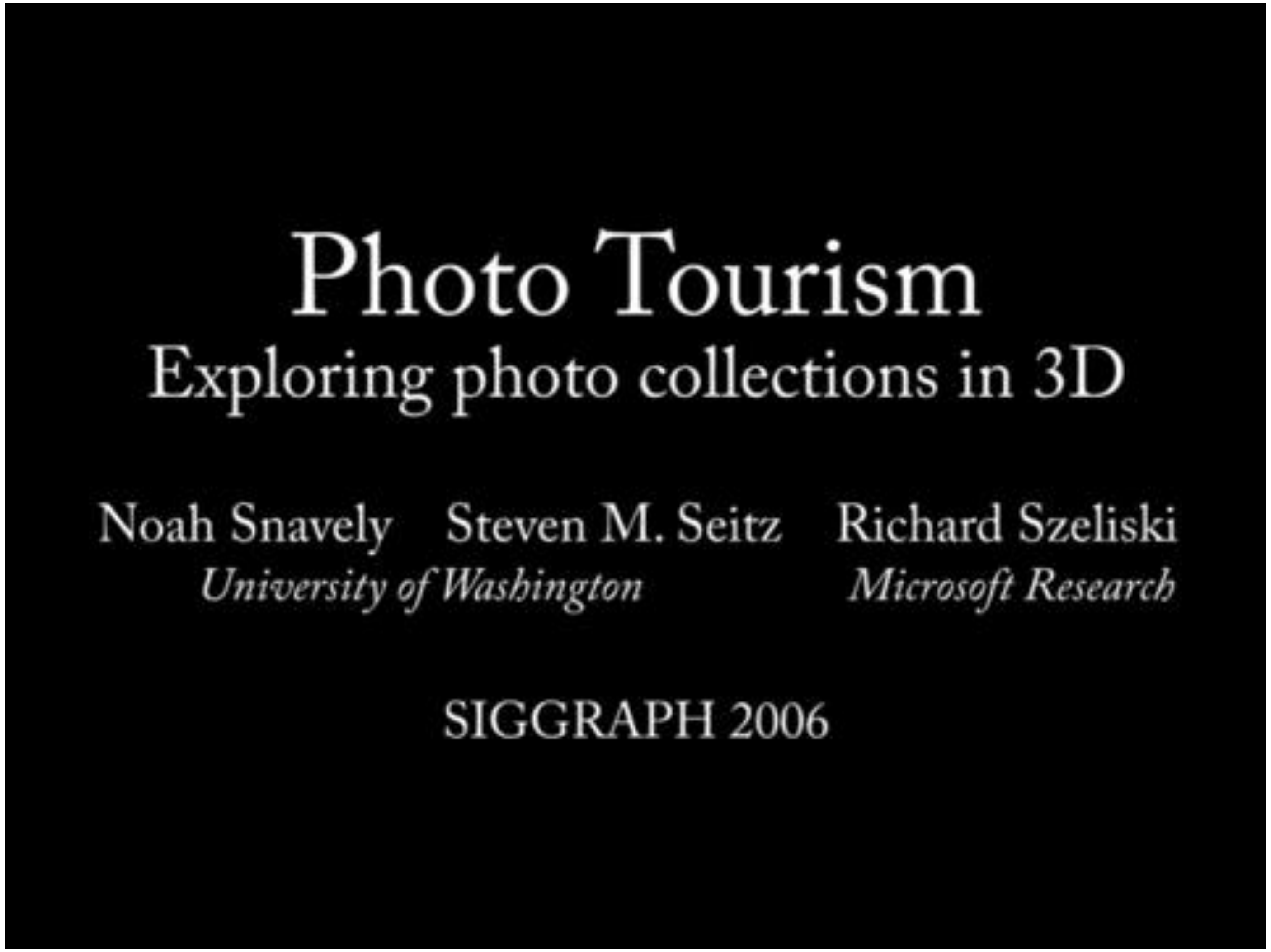


Photo Tourism

Exploring photo collections in 3D

Noah Snavely Steven M. Seitz Richard Szeliski

University of Washington *Microsoft Research*

SIGGRAPH 2006

Snavely et al. SIGGRAPH '06

Building Rome in a Day

Agarval et al. ICCV ’09

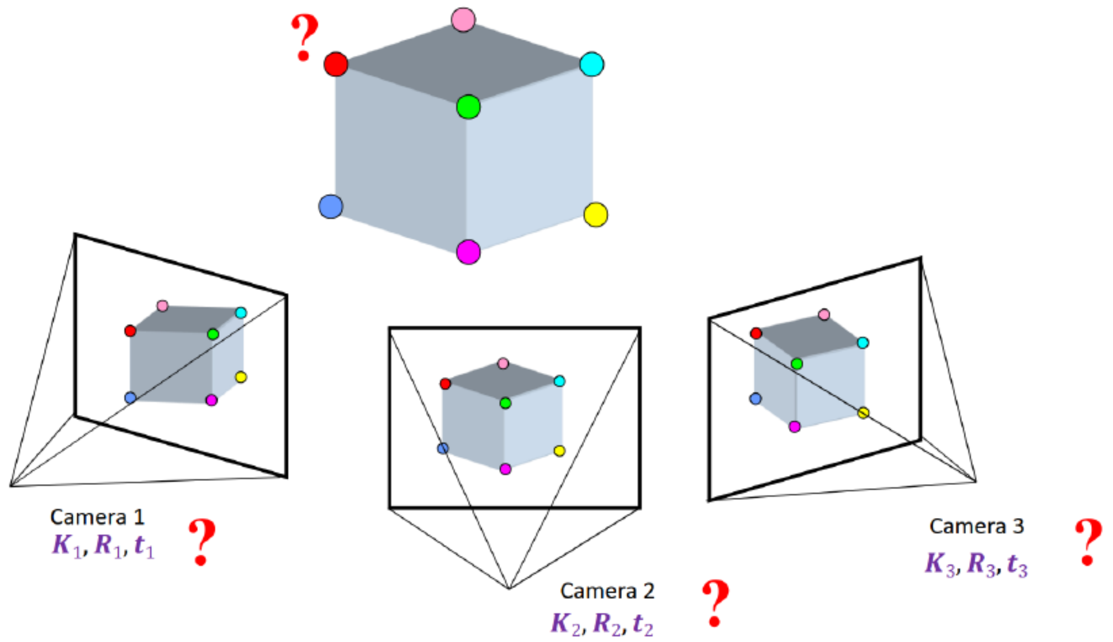
SfM Problem

Given: m images of n fixed 3D points such that

$$\mathbf{u}_{ij} = \mathbf{P}_i \mathbf{X}_j$$

Task: Jointly Estimate

- camera matrices $\mathbf{P}_i$
- 3D points $\mathbf{X}_j$



This lecture:

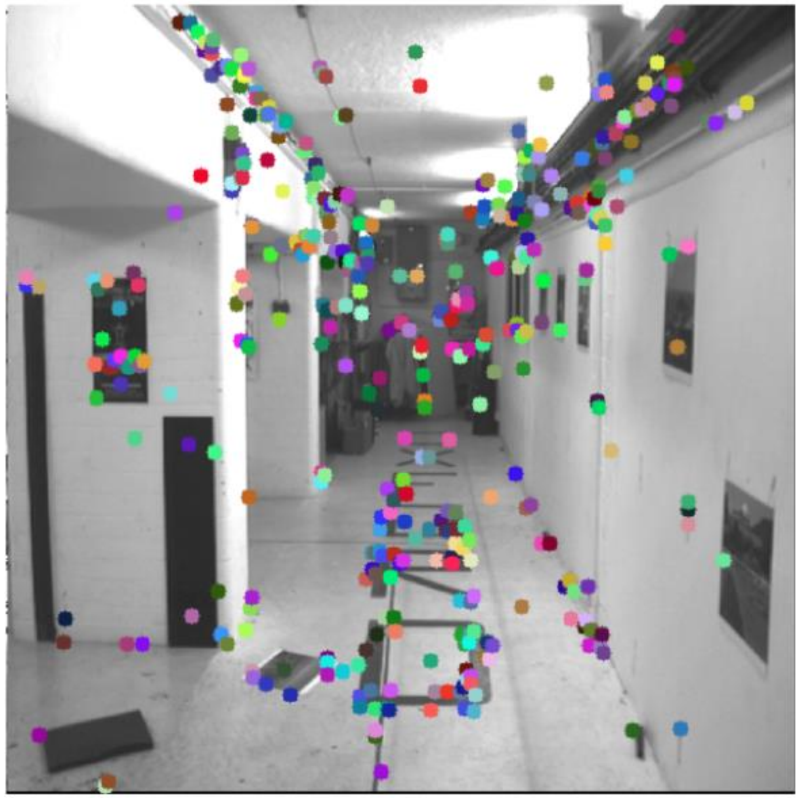
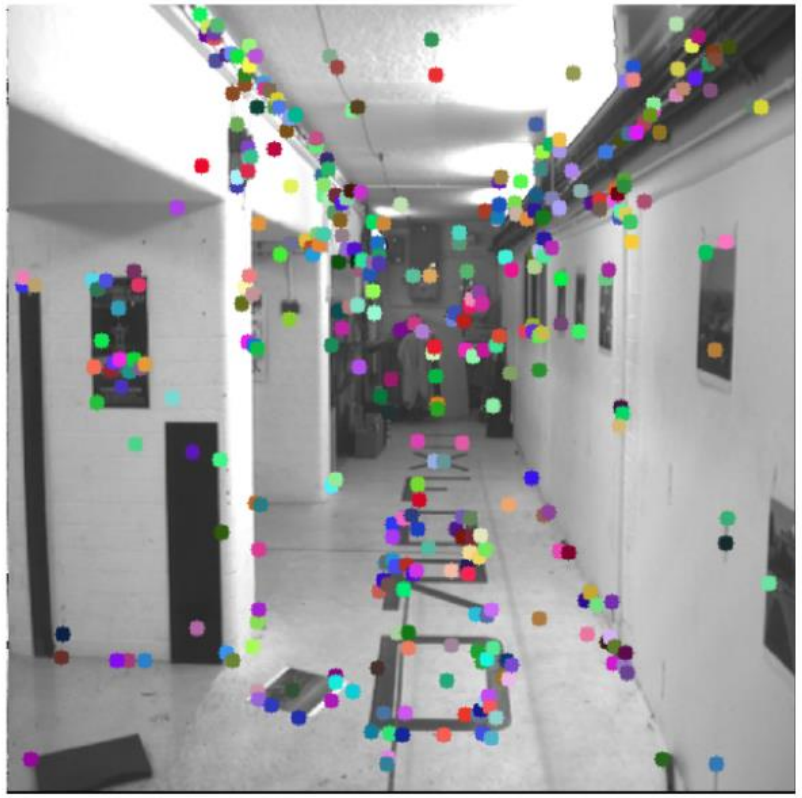
- Look at case where $m = 2$ (epipolar geometry)
- Extend to more m (bundle adjustment, SfM)

A Note on Correspondences

Throughout, we will assume to have access to precomputed 2D-2D correspondences:

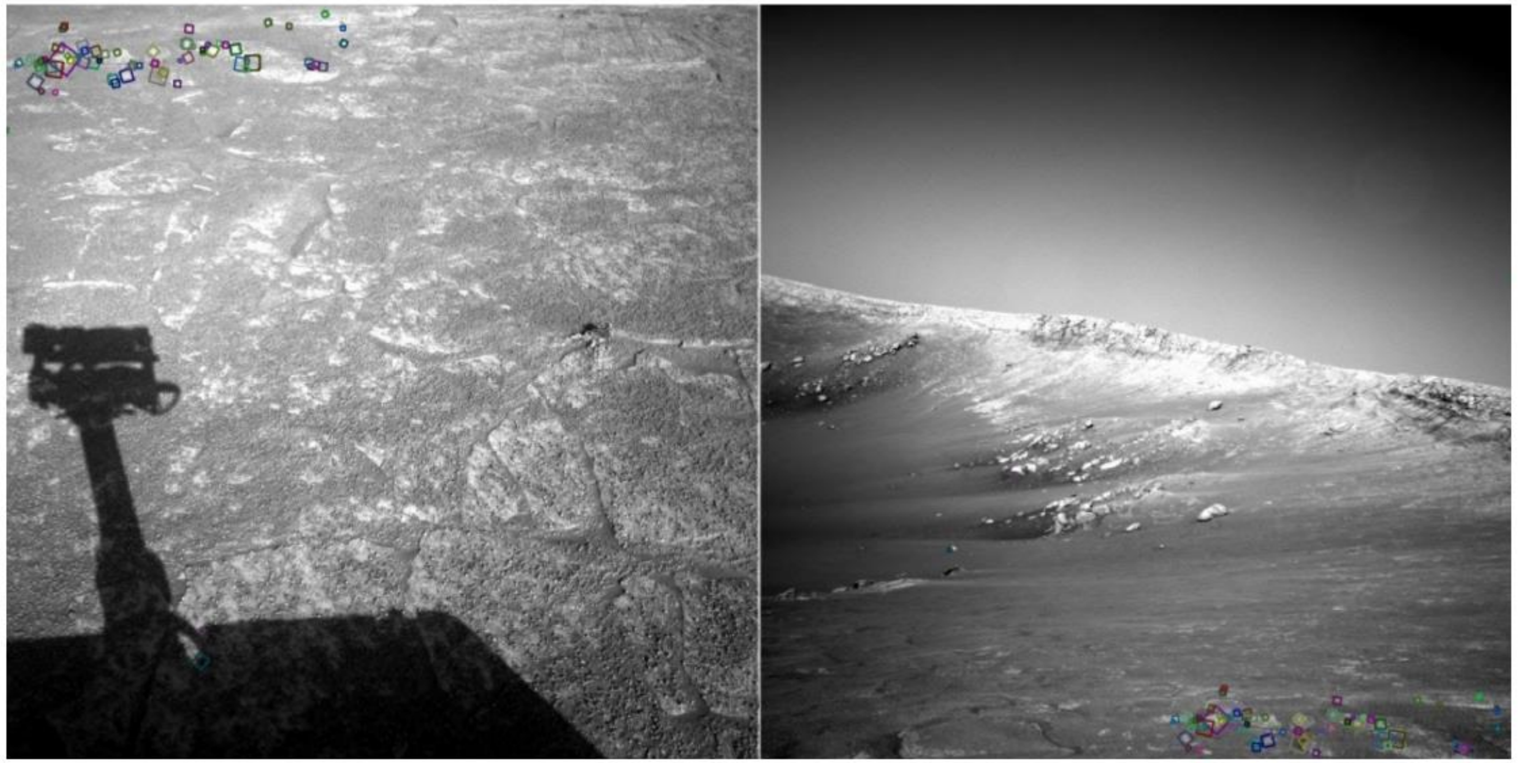
$\mathbf{u}_{ij} \sim \mathbf{u}_{i'j}$ correspond to point $\mathbf{X}_j$ being viewed from cameras i, i'.

We do not assume they are correct, i.e. we allow wrong correspondences and outliers.

A Note on Correspondences

- Rich literature on feature detection algorithms
    - SIFT, SURF, ORB, SuperPoint, ...
- This lecture will not deal with how to detect and match features



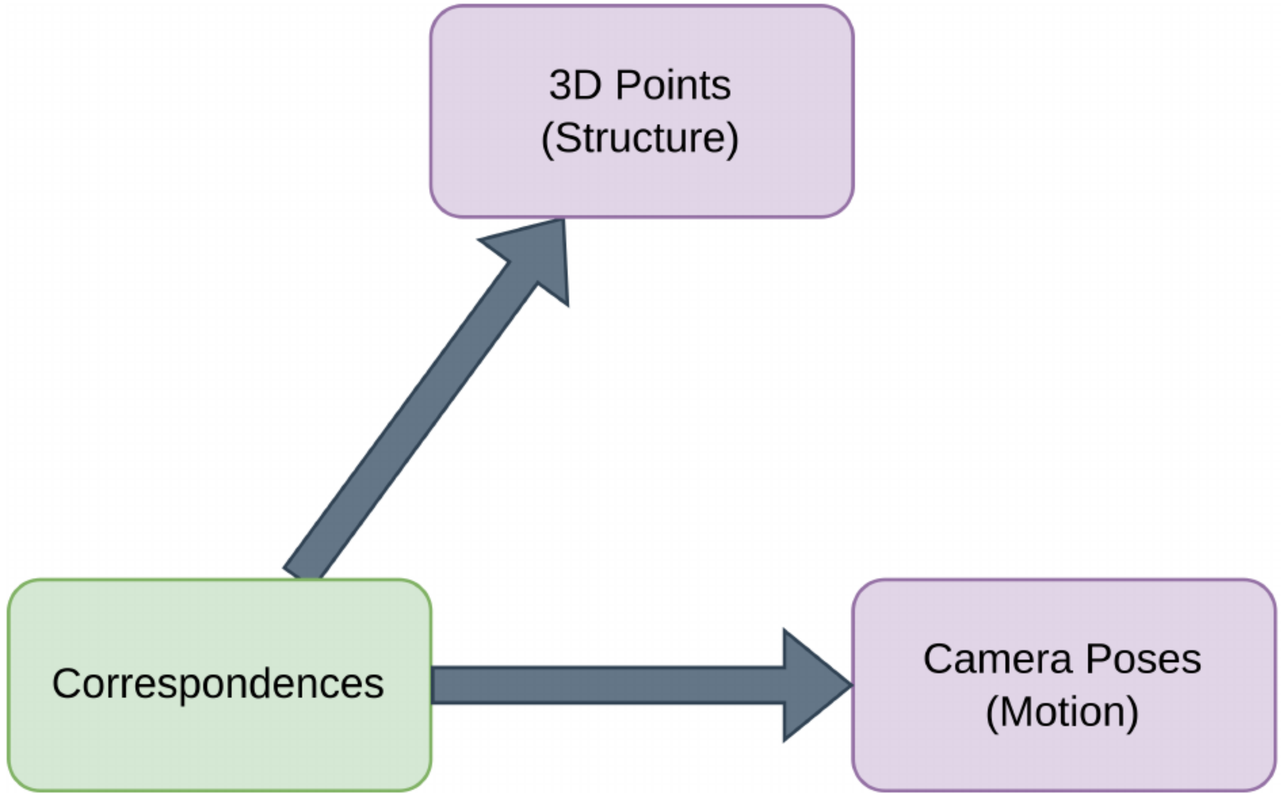
3 Components of SfM

From correspondences, infer

- camera poses (motion),
- 3D points (structure)

Related but simpler:

- Triangulation
- Pose estimation (PnP)



Typically, first solve for motion, then infer 3D Points via triangulation.

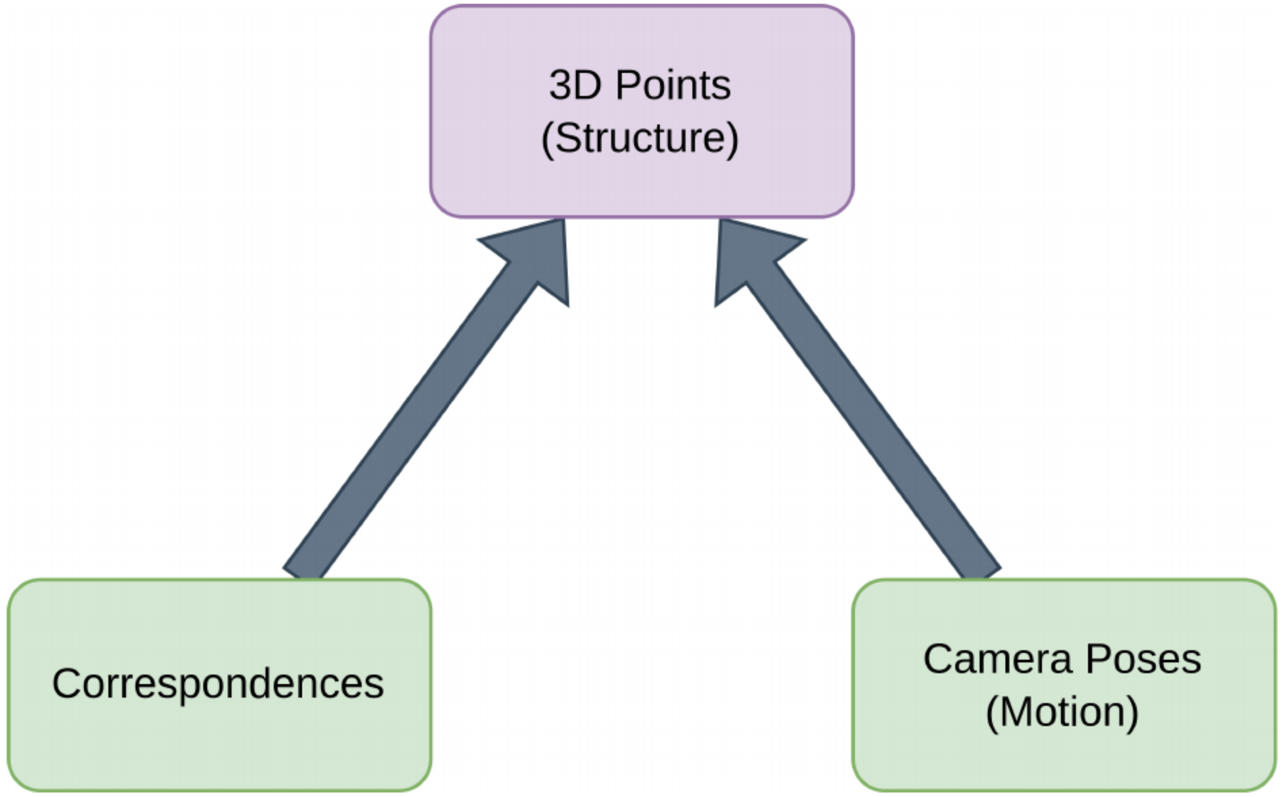
3 Components of SfM

From correspondences, infer

- camera poses (motion),
- 3D points (structure)

Related but simpler:

- Triangulation
- Pose estimation (PnP)



Typically, first solve for motion, then infer 3D Points via triangulation.

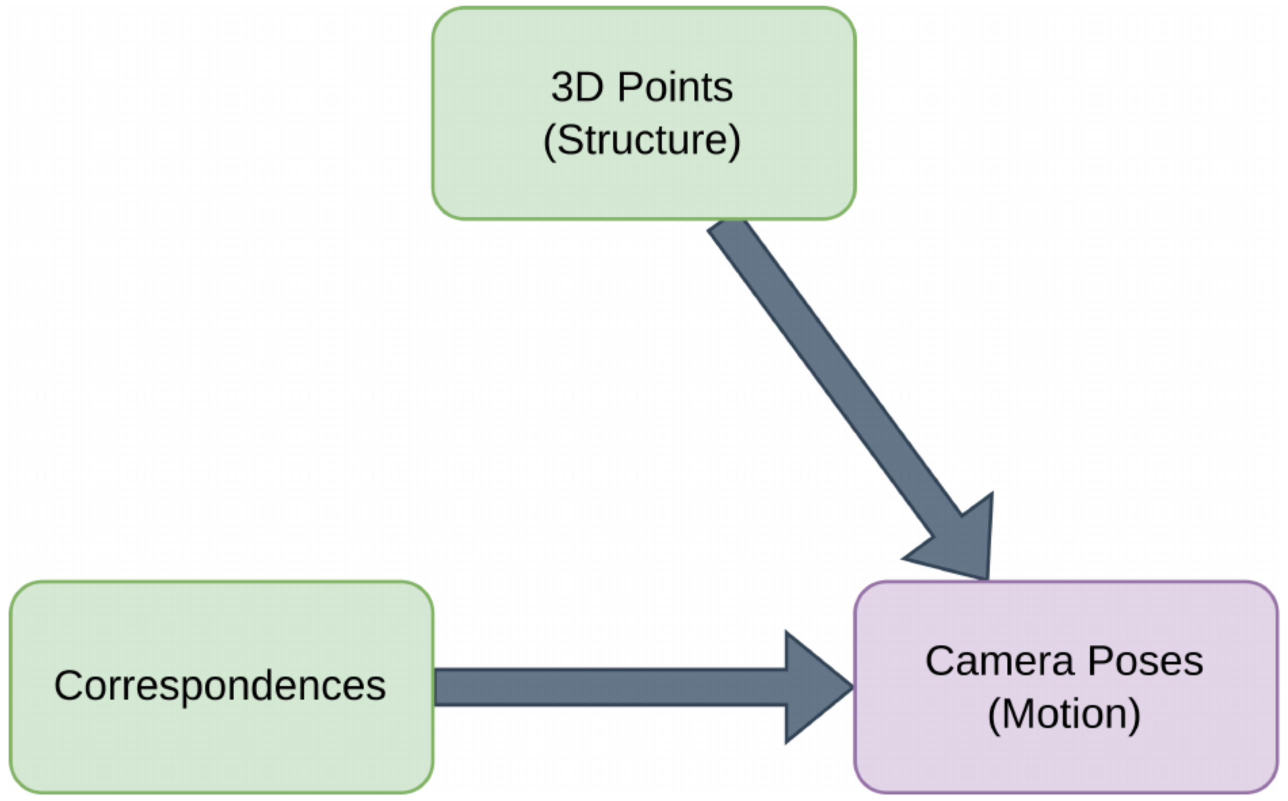
3 Components of SfM

From correspondences, infer

- camera poses (motion),
- 3D points (structure)

Related but simpler:

- Triangulation
- Pose estimation (PnP)



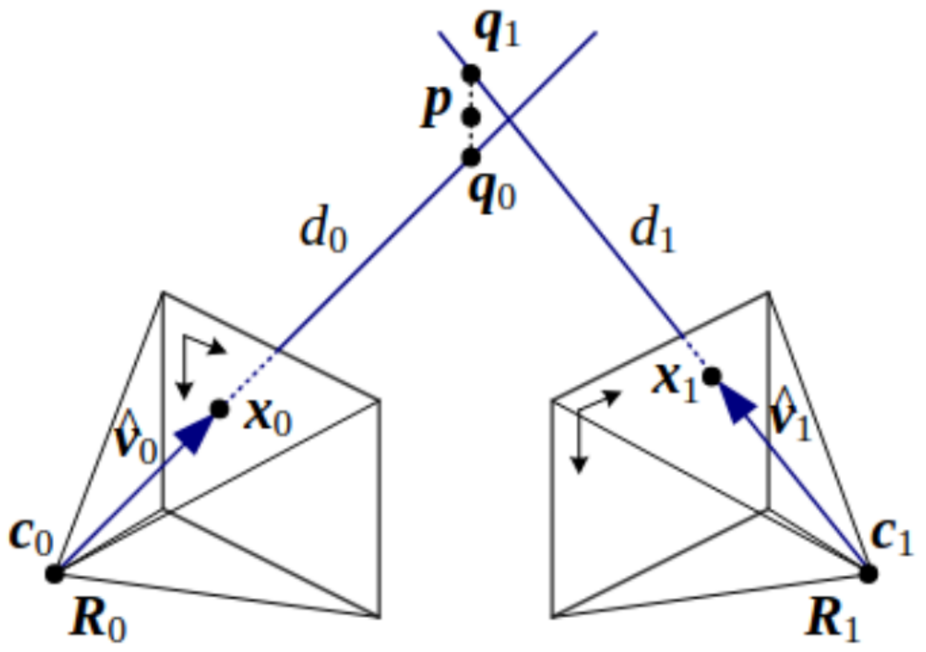
Typically, first solve for motion, then infer 3D Points via triangulation.

Triangulation

Setting: A point $\mathbf{X}$ with unknown 3D position is observed by 2 cameras.

Task: Find the 3D position of $\mathbf{X}$

If observations are perfect, the two vieweing rays should intersect exactly at $\mathbf{X}$.
In noisy settings, we have to find the point of minimum distance to both rays.



Linear Triangulation

- Assumption: $\mathbf{X}$ is projected to $\mathbf{u} \simeq \mathbf{P}\mathbf{X}$ by camera matrix
- We have $\mathbf{u} \times \mathbf{P}\mathbf{X} = 0$
- Can be rewritten as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{P}_1^T\mathbf{X} \\ \mathbf{P}_2^T\mathbf{X} \\ \mathbf{P}_3^T\mathbf{X} \end{bmatrix} = \mathbf{0}.$$

- The cross product evaluates to

$$\begin{bmatrix} y\mathbf{P}_3^T\mathbf{X} - \mathbf{P}_2^T\mathbf{X} \\ \mathbf{P}_1^T\mathbf{X} - x\mathbf{P}_3^T\mathbf{X} \\ x\mathbf{P}_2^T\mathbf{X} - y\mathbf{P}_1^T\mathbf{X} \end{bmatrix} = \mathbf{0}$$

- Last row is linear combination of first two

Linear Triangulation

- We get the linear equations:

$$\begin{bmatrix} y\mathbf{P}_3^T - \mathbf{P}_2^T \\ \mathbf{P}_1^T - x\mathbf{P}_3^T \end{bmatrix} \mathbf{X} = \mathbf{0}$$

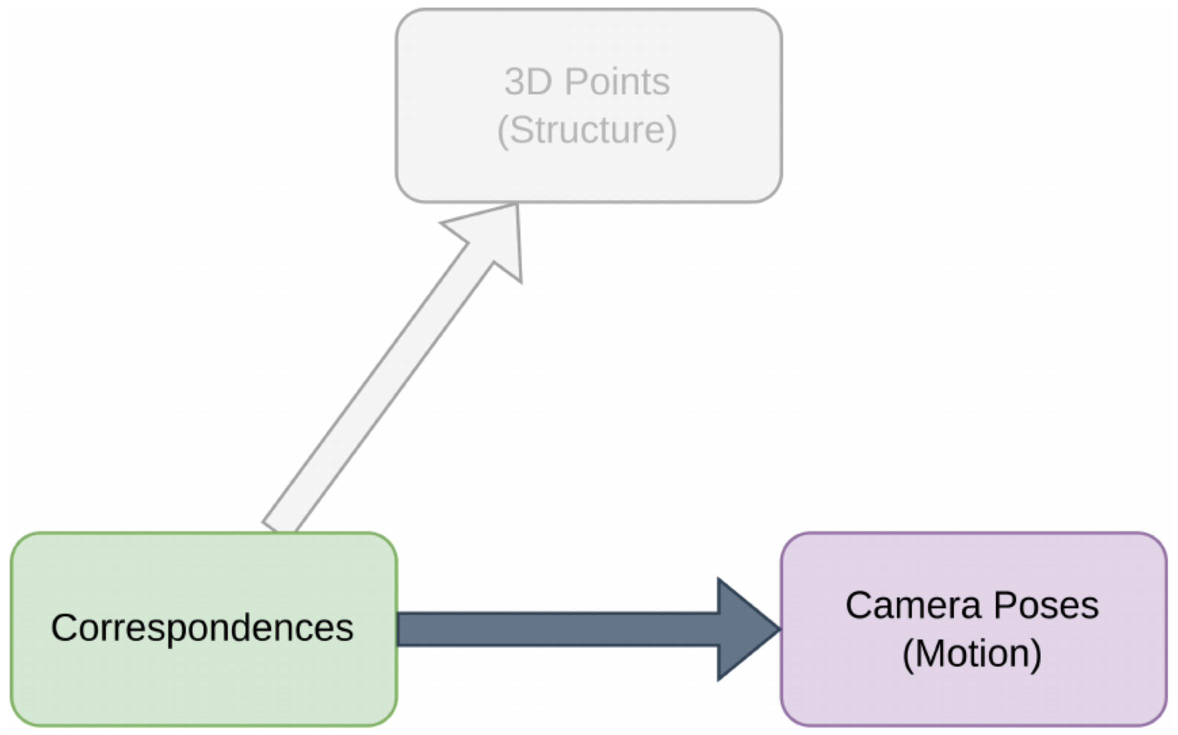
- A single camera is of course not enough to triangulate a point
- Let $\mathbf{P}'$ be projection matrix of another camera and $\mathbf{u}'$ projection of $\mathbf{X}$
- We can setup the linear system

$$\begin{bmatrix} y\mathbf{P}_3^T - \mathbf{P}_2^T \\ \mathbf{P}_1^T - x\mathbf{P}_3^T \\ y'\mathbf{P}_3'^T - \mathbf{P}_2'^T \\ \mathbf{P}_1'^T - x'\mathbf{P}_3'^T \end{bmatrix} \mathbf{X} = \mathbf{0}$$

- Can be solved in the least squares sense using SVD.

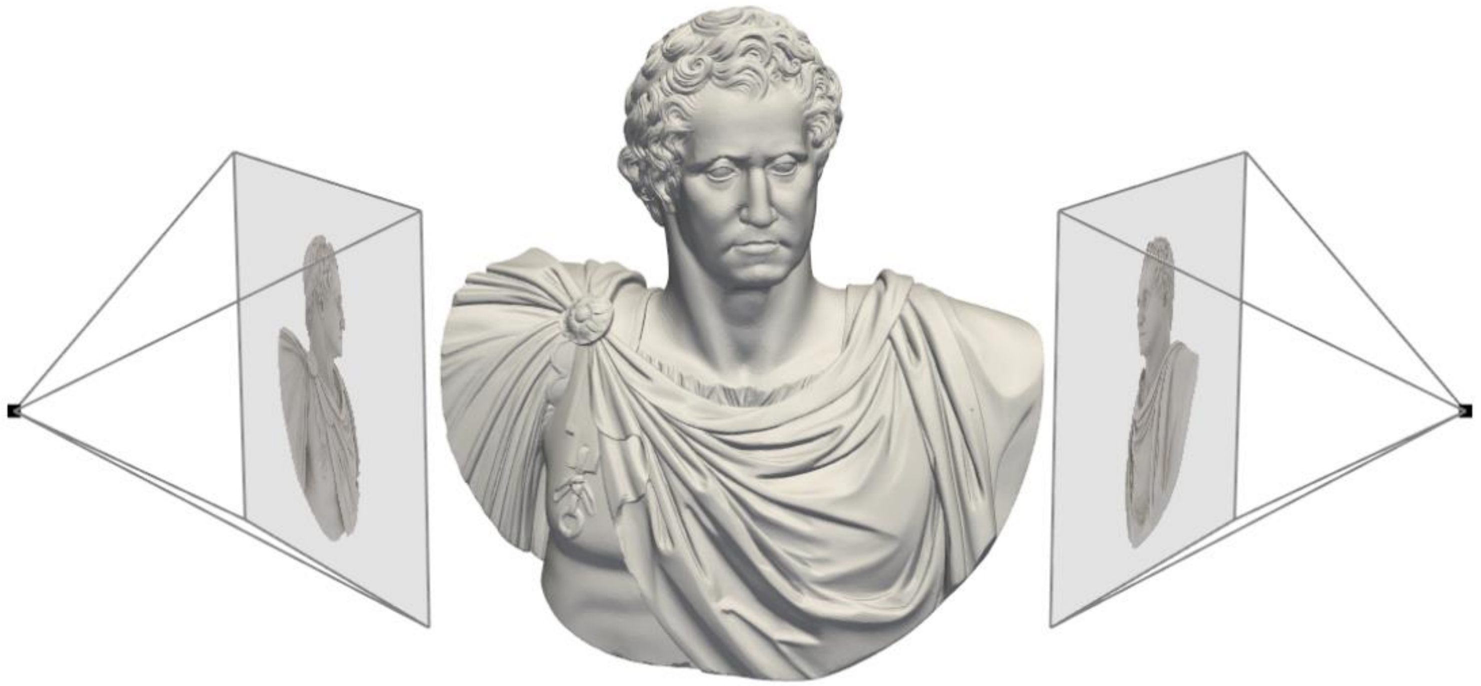
Epipolar Geometry

Estimating Camera Poses from Correspondences



Epipolar Geometry: Setup

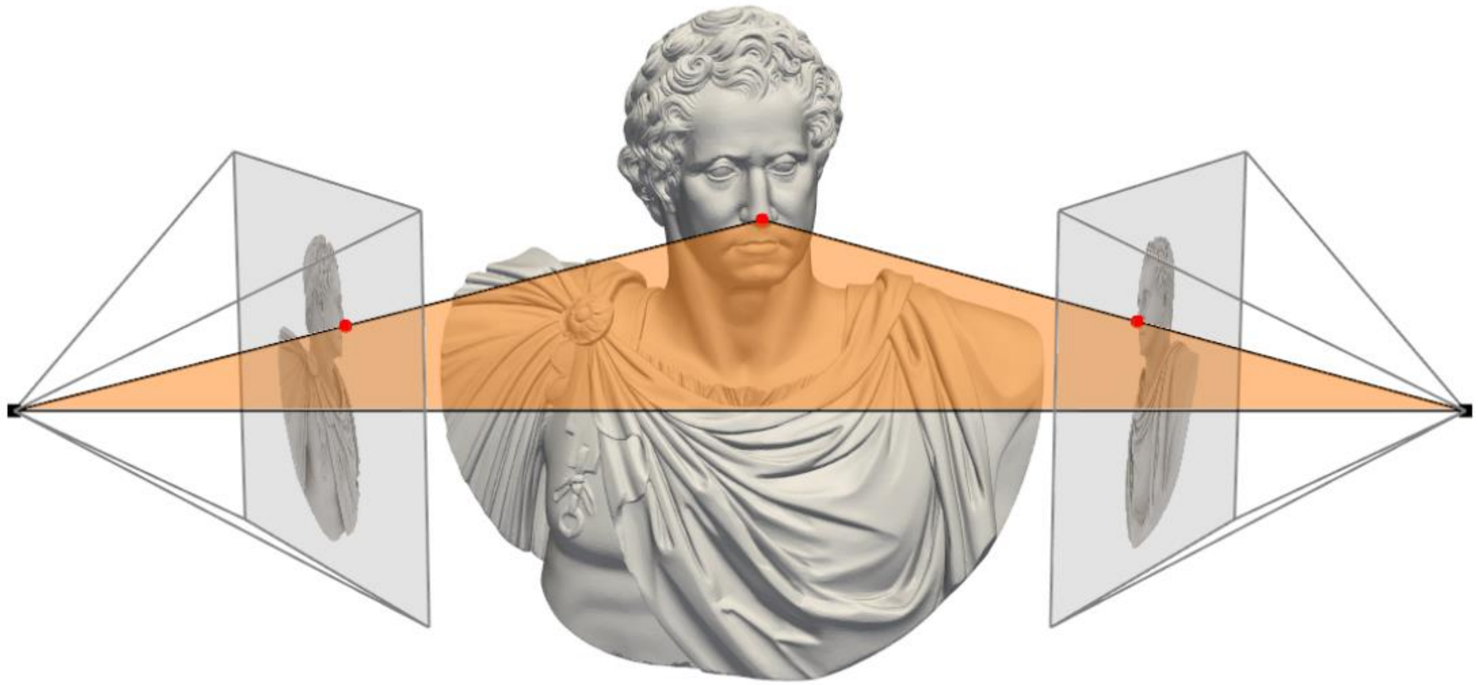
A scene is viewed by two cameras from slightly different angles



Camera views:

Epipolar Geometry: The Epipolar Plane

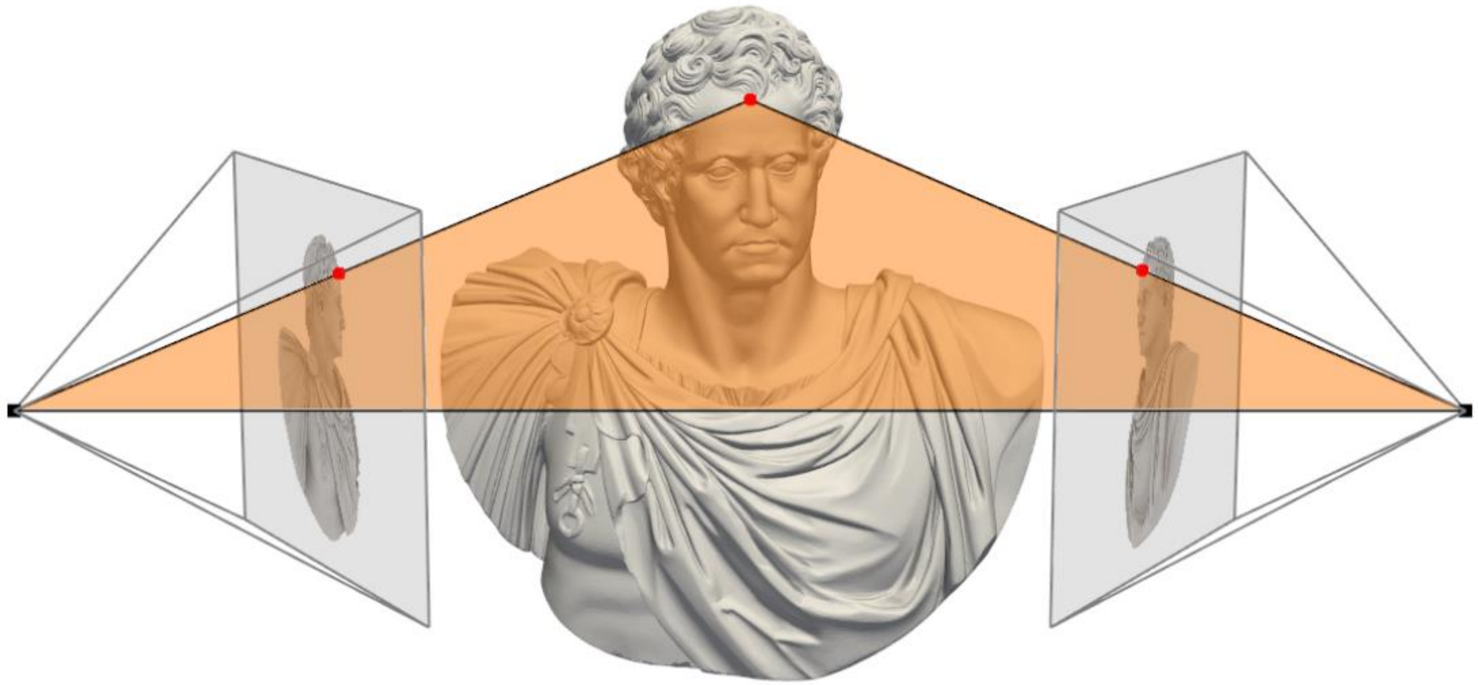
- Each 3D point forms a plane with the two camera centers
- This is called epipolar plane



Camera views:

Epipolar Geometry: The Epipolar Plane

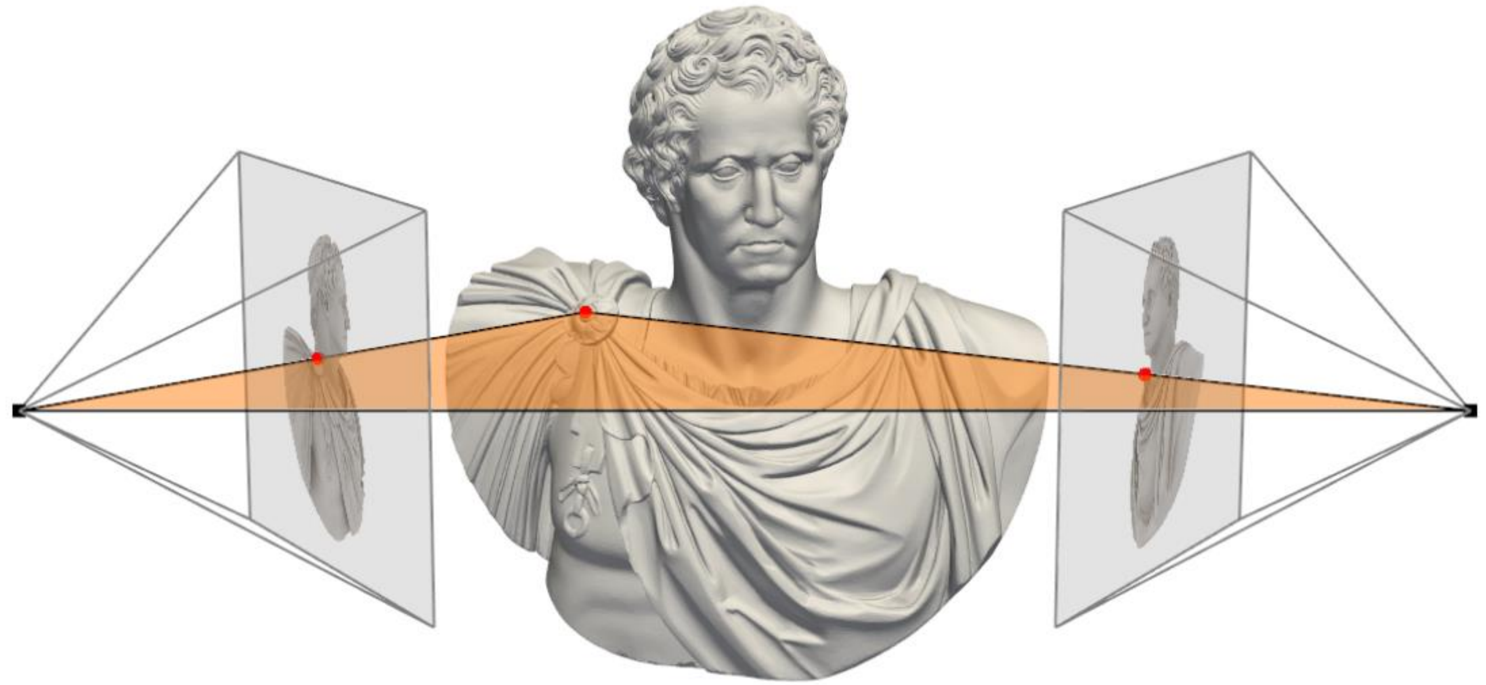
- Each 3D point forms a plane with the two camera centers
- This is called epipolar plane



Camera views:

Epipolar Geometry: The Epipolar Plane

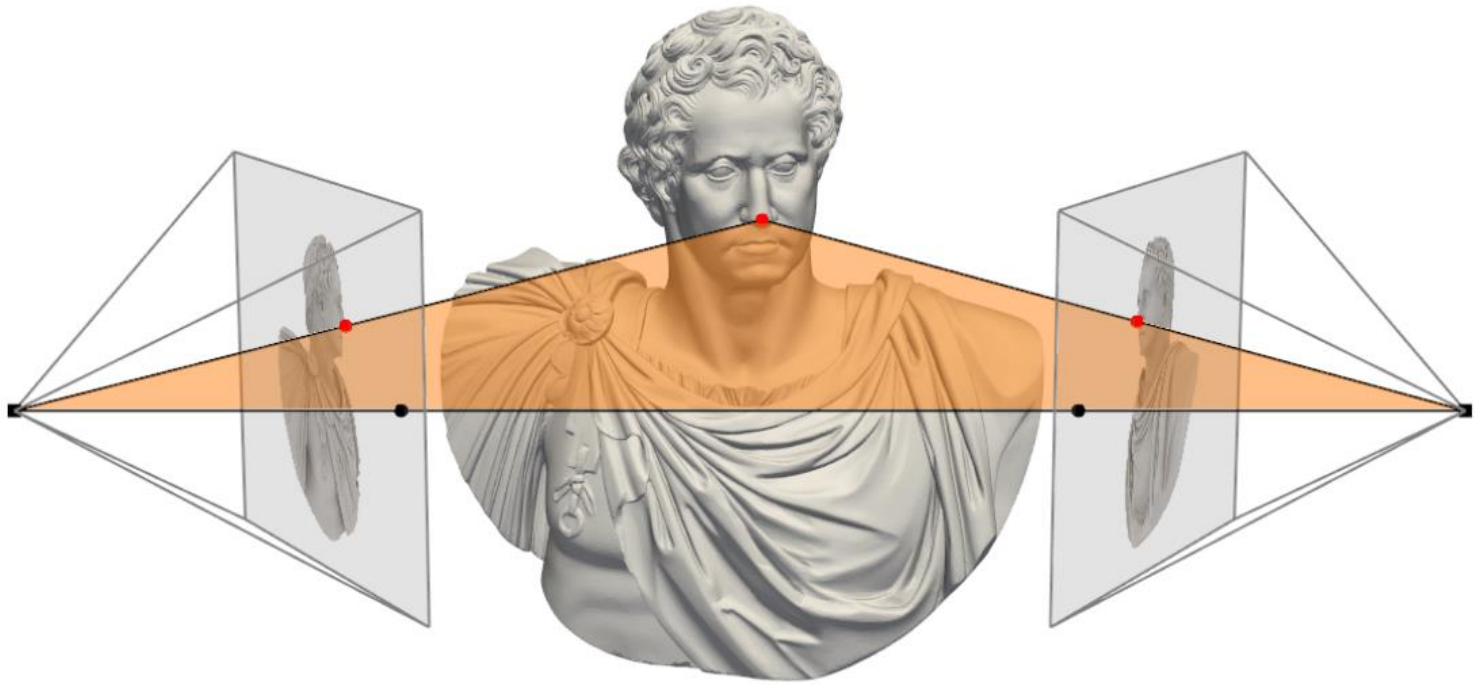
- Each 3D point forms a plane with the two camera centers
- This is called epipolar plane



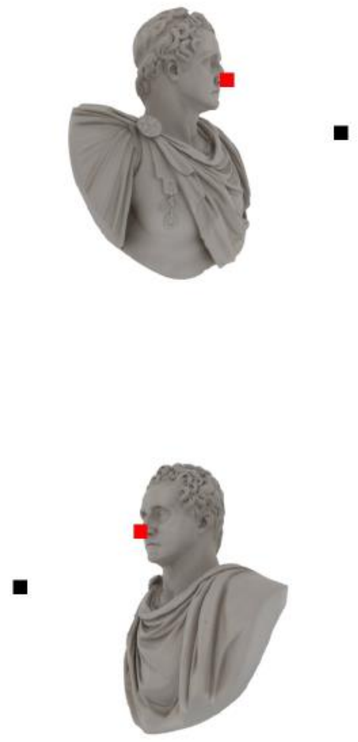
Camera views:

Epipolar Geometry: The Epipoles

- Baseline is always part of epipolar plane
- Epipolar points: Intersections of baseline with image planes

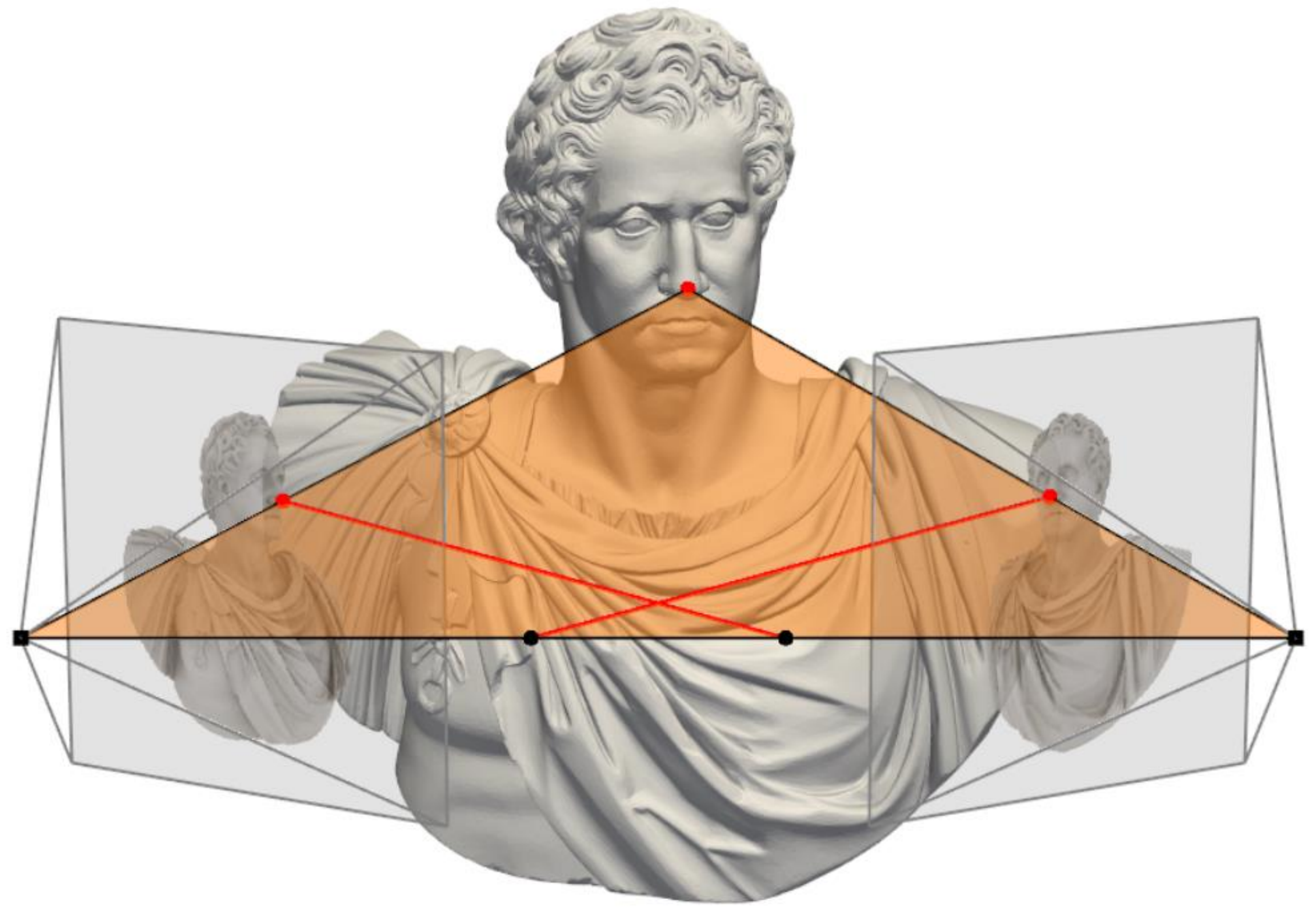


Camera views:



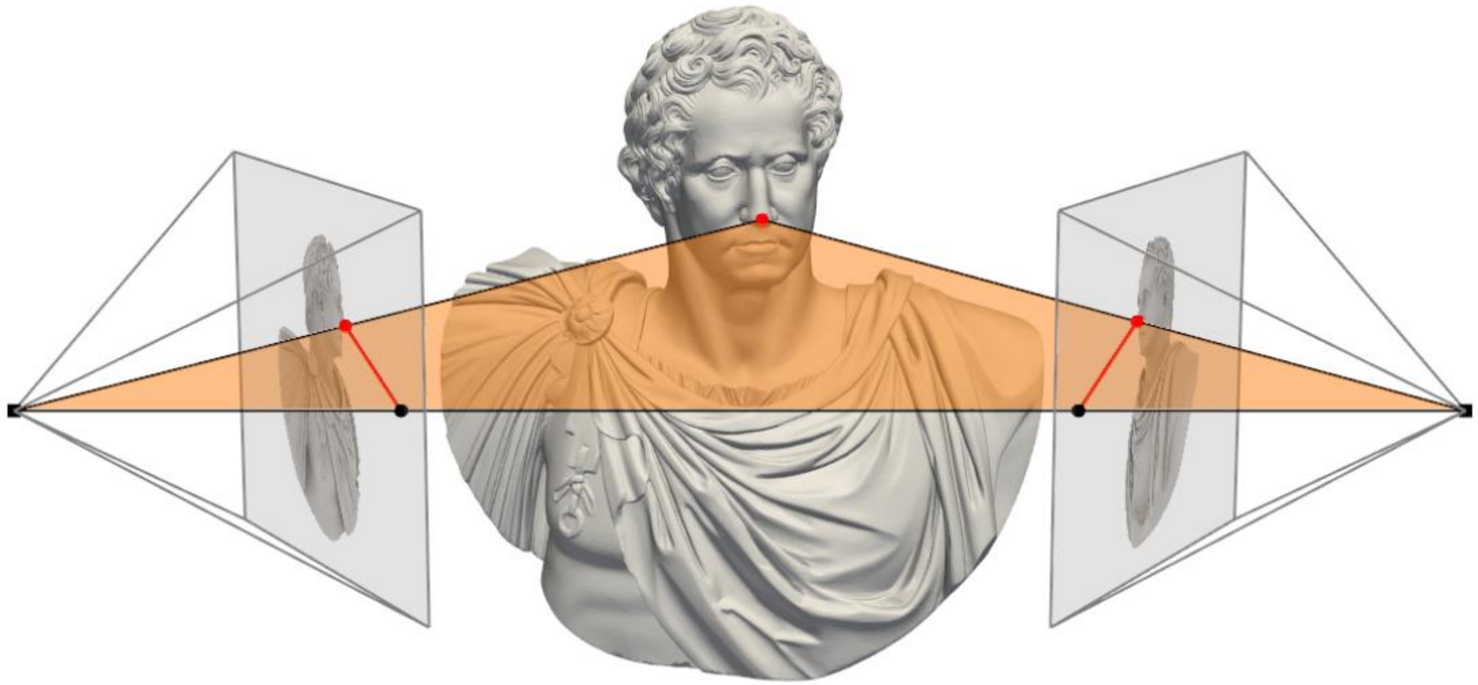
Epipolar Geometry: The Epipolar Points

- Epipolar Points must not always lie within the images
- Can also lie at infinity

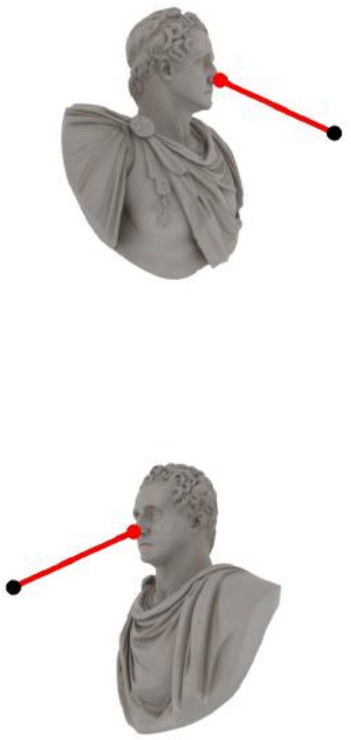


Epipolar Geometry: The Epipolar Lines

- Epipolar lines: Intersection of image planes with epipolar plane
- Projection of other camera's viewing ray onto image plane

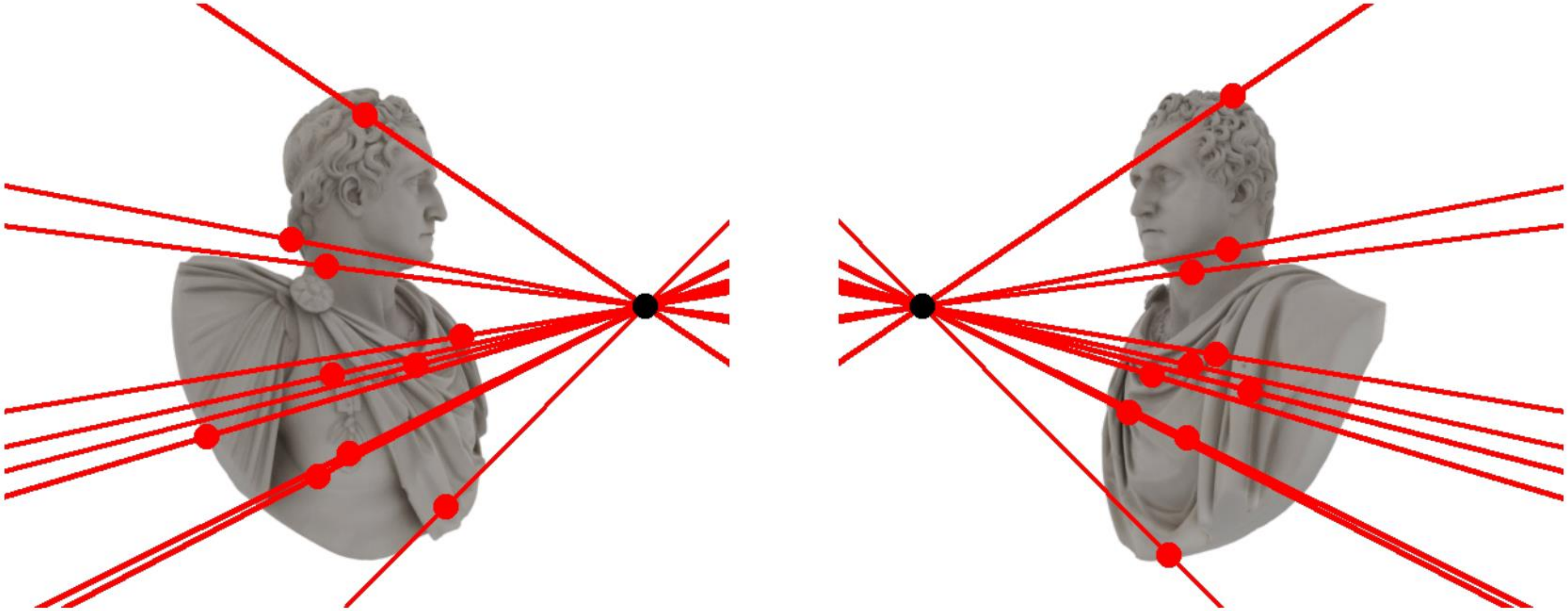


Camera views:



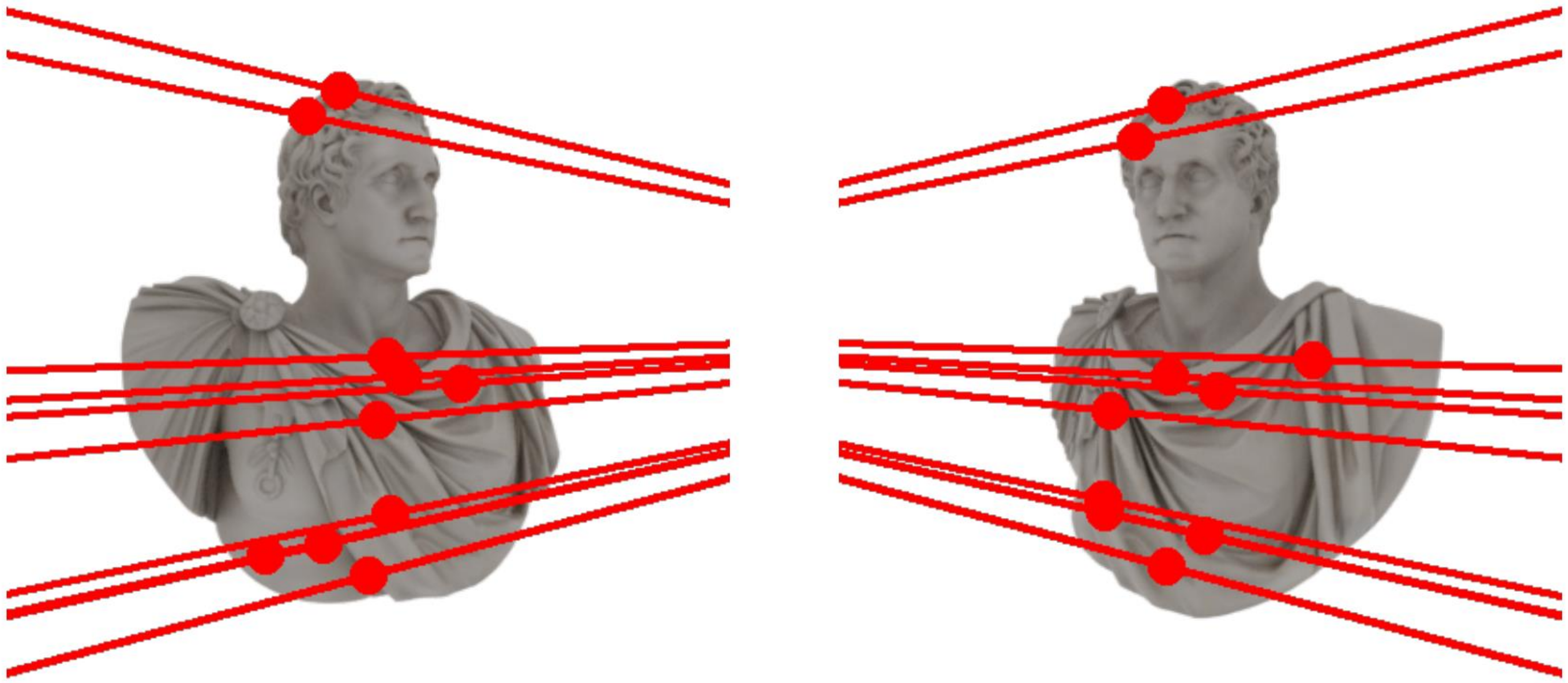
Epipolar Geometry: The Epipolar Lines

All epipolar lines converge in epipolar points (can be inside or outside of image)

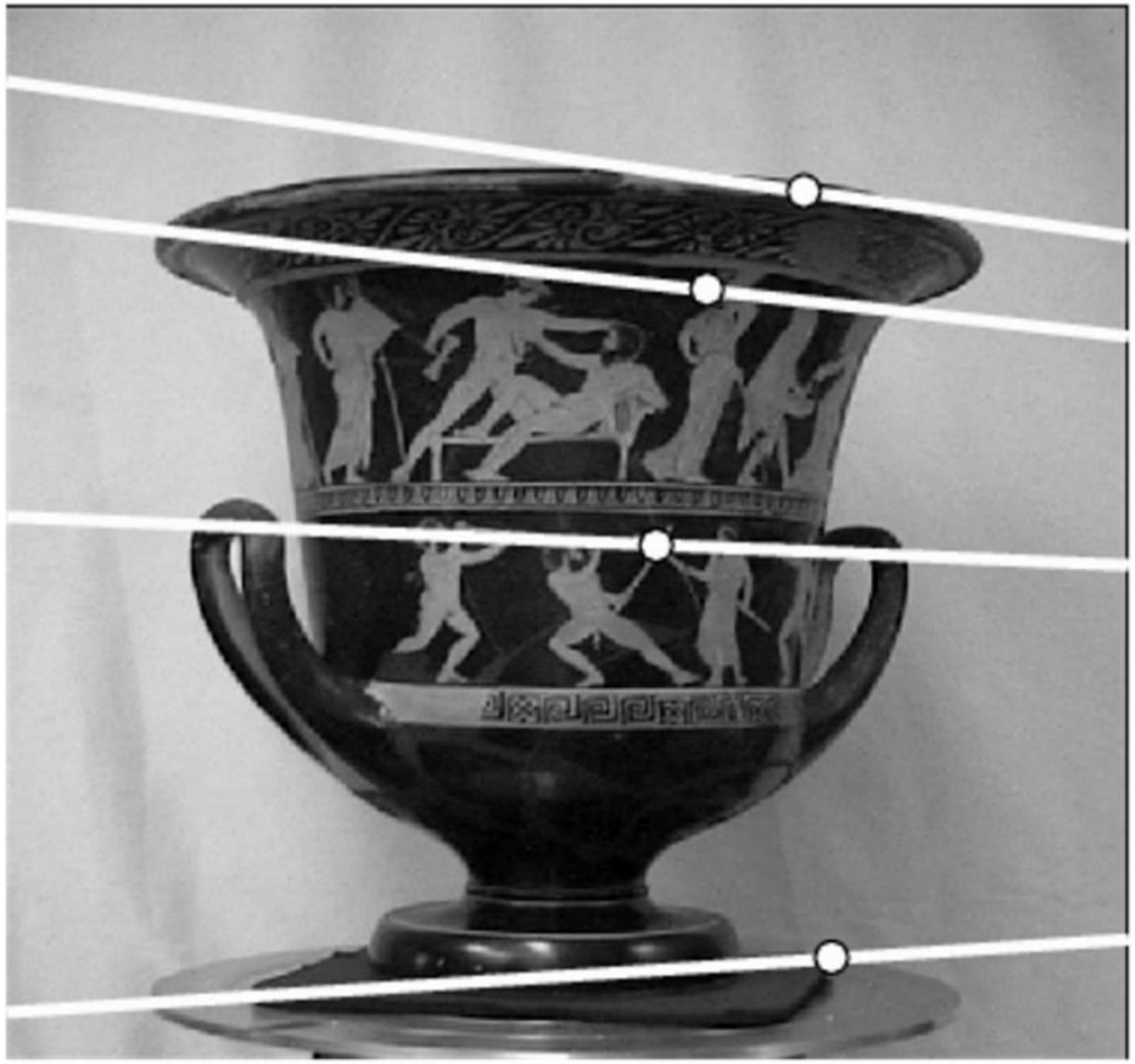


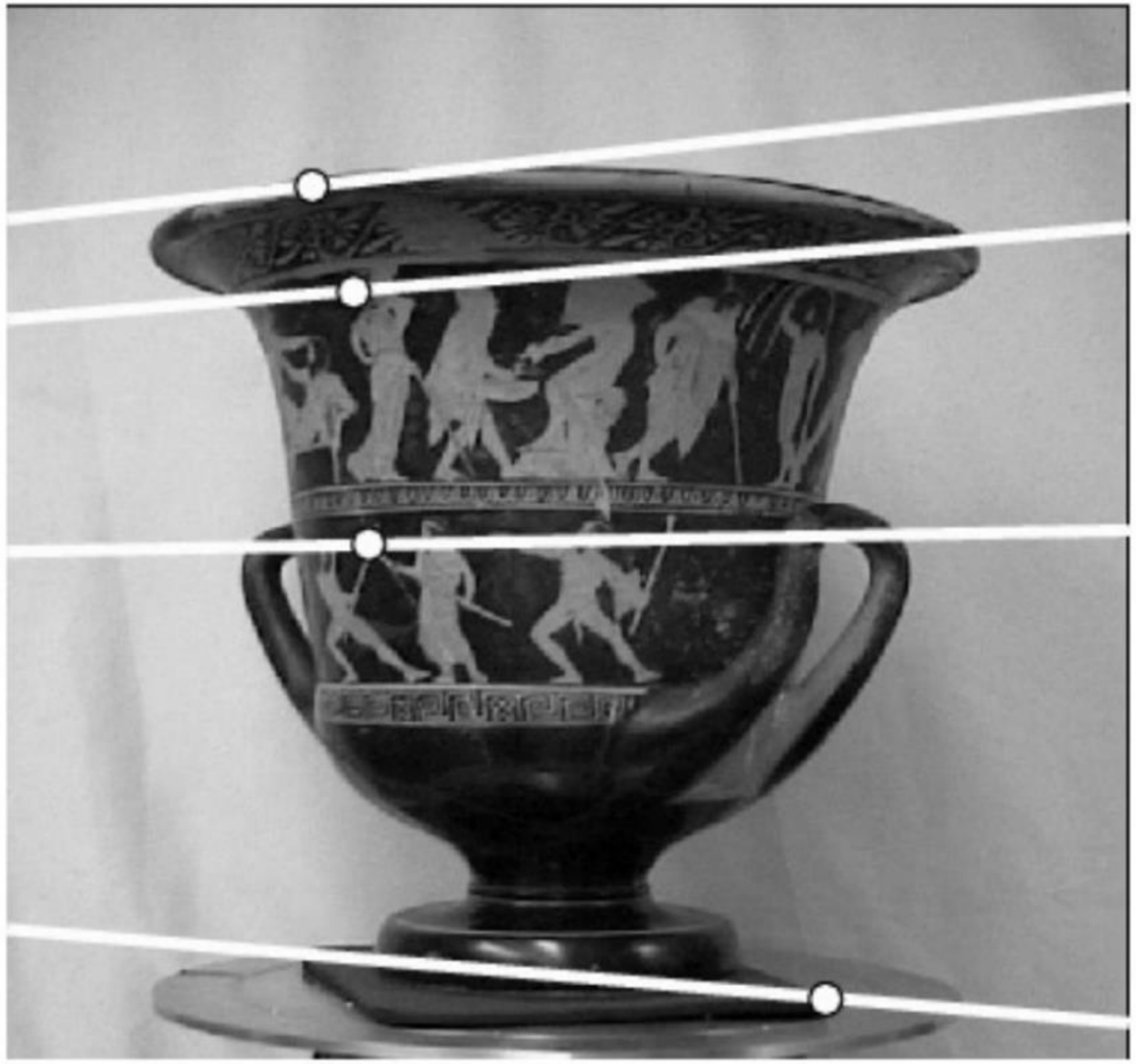
Epipolar Geometry: The Epipolar Lines

All epipolar lines converge in epipolar points (can be inside or outside of image)



Epipolar Lines



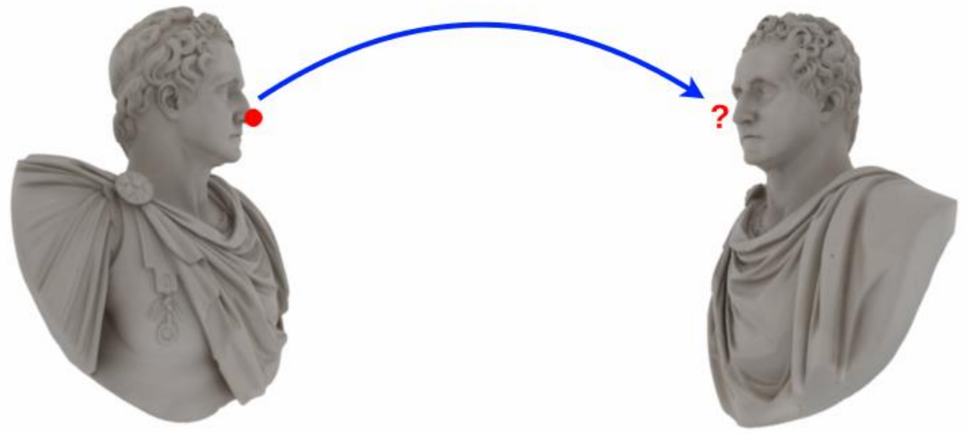


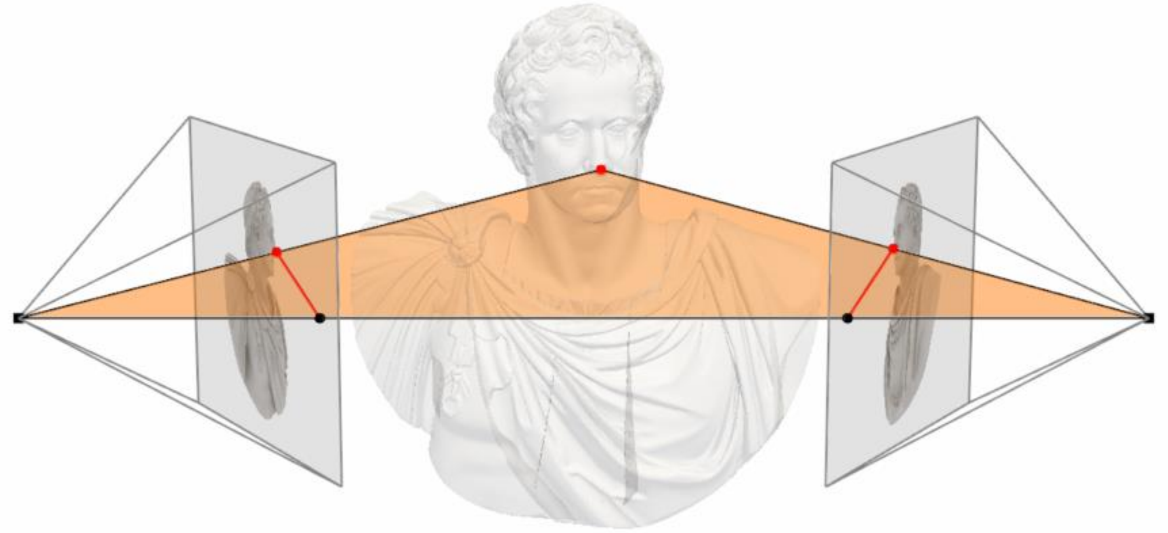
2D Point Matching with Epipolar Lines

Given:

- 2 images
- point in one image

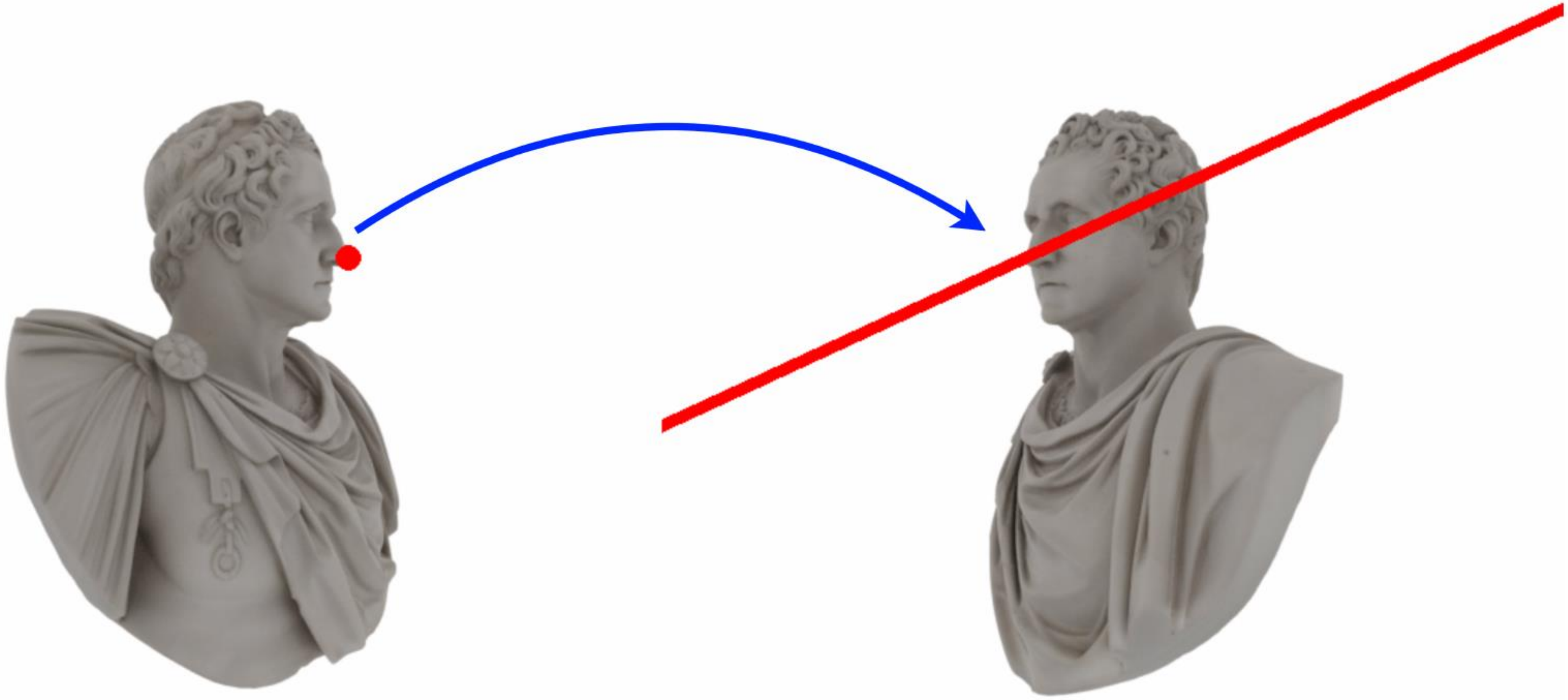
Task:
Find point in other image



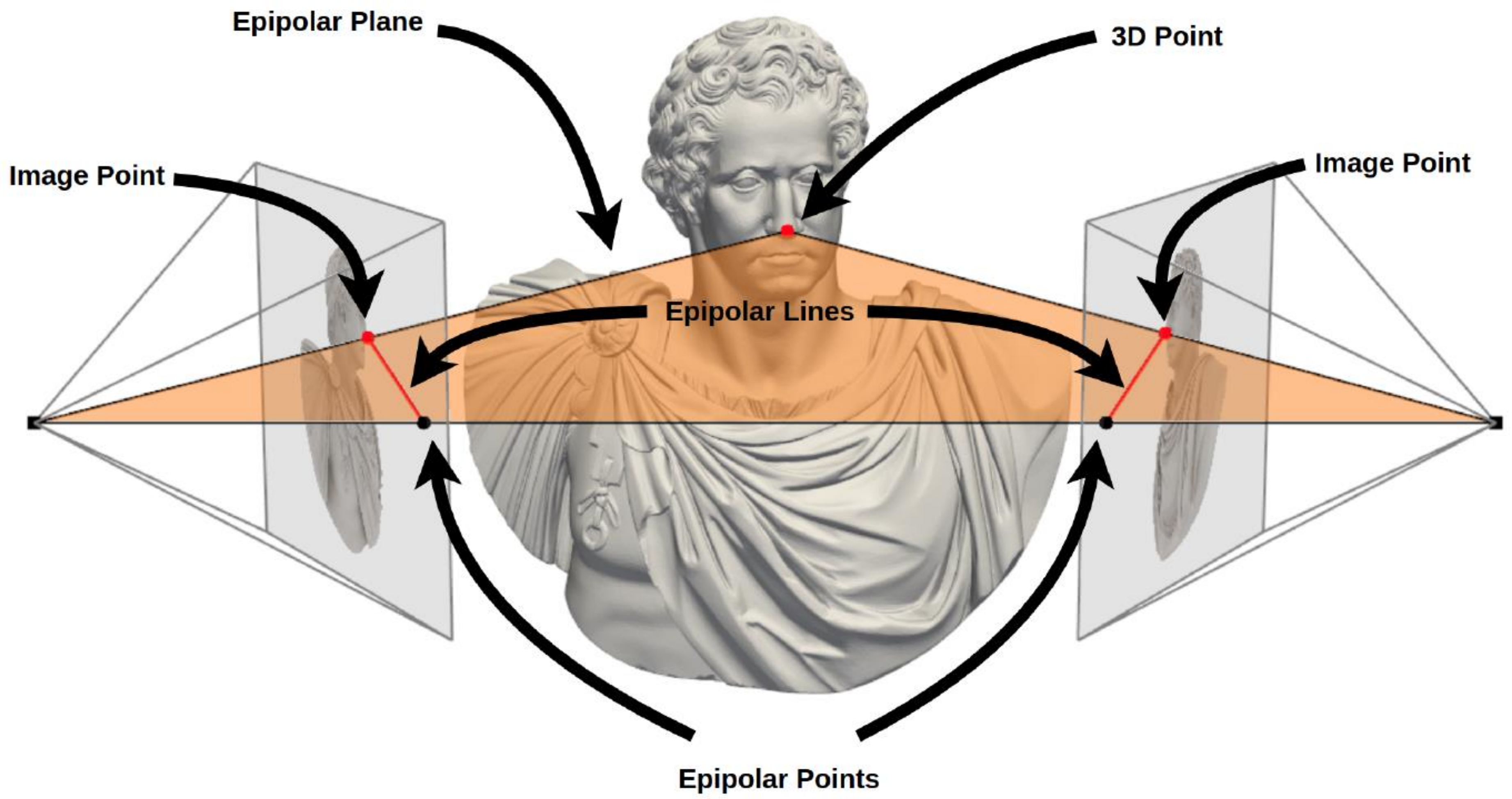


2D Point Matching with Epipolar Lines

- Corresponding point must lie on epipolar line!
- This is a 1 dimensional search space



Epipolar Geometry: Overview



We can derive a neat equation called epipolar constraint that encodes all of these relationships.

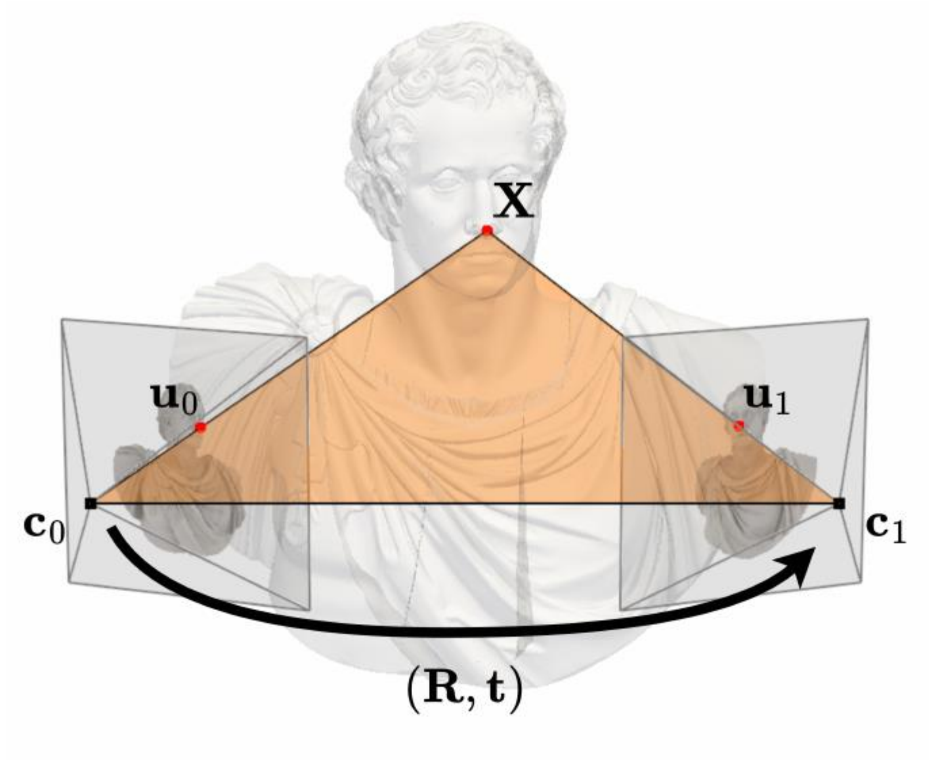
Formal Setup

Setting:
3D point $\mathbf{X}$ viewed from two cameras. We take $\mathbf{c}_0$'s frame as world frame.

- relative pose $\mathbf{R} = \mathbf{R}_{1\to 0}, \mathbf{t} = \mathbf{t}_0$
- camera matrices

$$\mathbf{P}_0 = \mathbf{K}_0[\mathbf{I}, \mathbf{0}]$$
$$\mathbf{P}_1 = \mathbf{K}_1[\mathbf{R}^T, -\mathbf{R}^T\mathbf{t}]$$

Projection of $\mathbf{X}$ onto the images planes: $\mathbf{u}_i \simeq \mathbf{P}_i\mathbf{X}$.



The Epipolar Constraint

Ray directions in local camera frames:

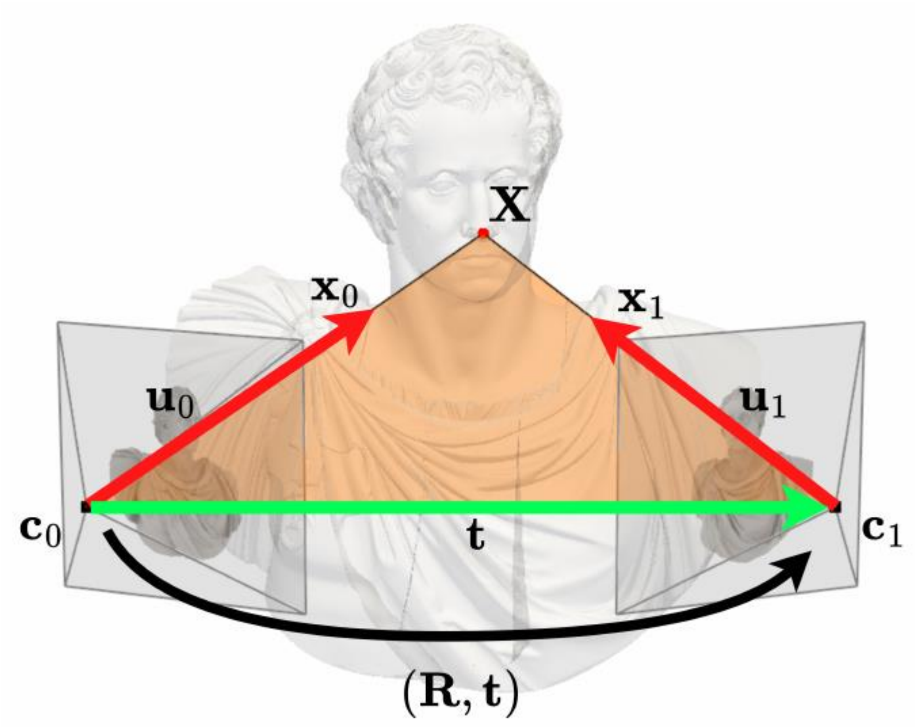
$$\mathbf{x}_0 = \mathbf{K}_0^{-1}\mathbf{u}_0 \qquad \mathbf{x}_1 = \mathbf{K}_1^{-1}\mathbf{u}_1$$

Ray directions in global frame:

$$\hat{\mathbf{x}}_0 = \mathbf{x}_0 \qquad \hat{\mathbf{x}}_1 = \mathbf{R}\mathbf{x}_1$$

Normal of epipolar plane:

$$\begin{aligned}\mathbf{n} &= \mathrm{t} \times \hat{\mathbf{x}}_1 \\ &= \mathbf{t} \times (\mathbf{R}\mathbf{x}_1) \\ &= [\mathbf{t}]_\times \mathbf{R}\mathbf{x}_1\end{aligned}$$



The Epipolar Constraint

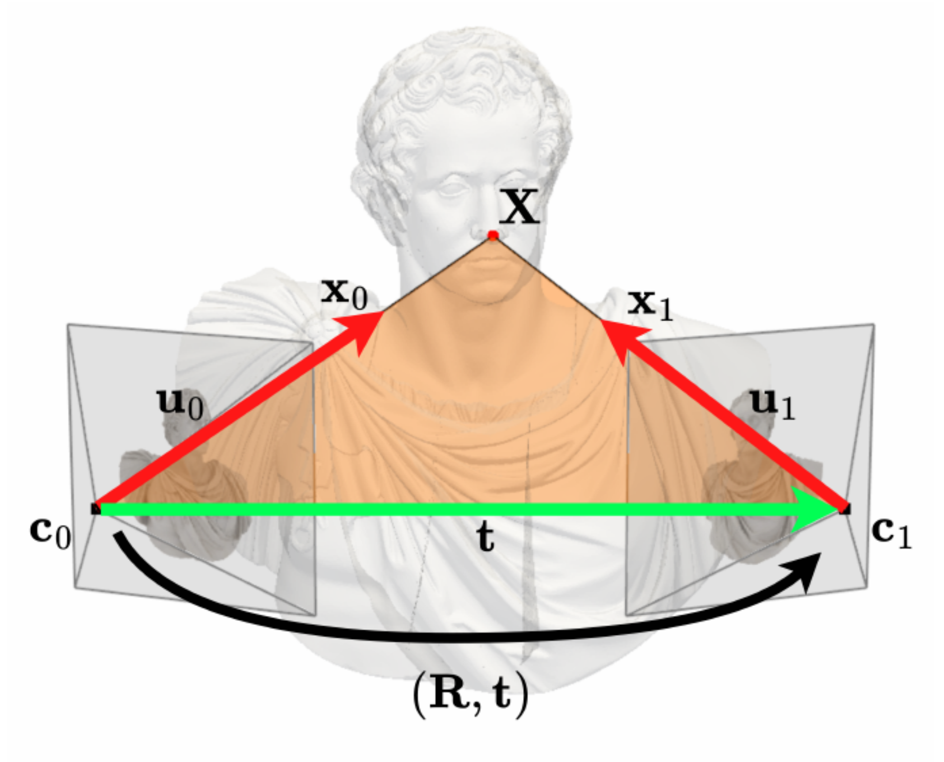
Notice that $\mathbf{x}_0$ lies on the epipolar plane, so we have

$$\mathbf{x}_0^T [\mathbf{t}]_\times \mathbf{R} \mathbf{x}_1 = 0$$

This is the epipolar constraint!

The matrix $\mathbf{E} := [\mathbf{t}]_\times \mathbf{R}$ is called essential matrix.

What are the dimensions of $\mathbf{E}$?



Epipolar Geometry: Essential Matrix

Epipolar constraint:

$$\mathbf{x}_0^T \mathbf{E} \mathbf{x}_1 = 0$$

The essential matrix $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} \in \mathbb{R}^{3\times 3}$ captures information about the epipolar geometry of the two cameras:

- $\mathbf{E}\tilde{\mathbf{e}}_1 = \mathbf{0} = \tilde{\mathbf{e}}_0^T \mathbf{E}$, i.e. the (calibrated) epipoles are zero-value left and right singular vectors respectively.
- $\tilde{\mathbf{l}}_0 = \mathbf{E}\mathbf{x}_1$ is the (calibrated) epipolar line corresponding to $\mathbf{x}_1$.
- $\tilde{\mathbf{l}}_1 = \mathbf{E}^T\mathbf{x}_0$ is the (calibrated) epipolar line corresponding to $\mathbf{x}_0$.
- $\mathbf{E}$ has rank 2 and 5 degrees of freedom.

Epipolar Geometry: Fundamental Matrix

Recall that $\mathbf{x}_i = \mathbf{K}_i^{-1}\mathbf{u}_i$. We can rewrite the epipolar constraint as

$$\mathbf{u}_0^T \boxed{\underbrace{\mathbf{K}_0^{-T}\mathbf{E}\mathbf{K}_1^{-1}}_{=:\mathbf{F}}} \mathbf{u}_1 = 0$$

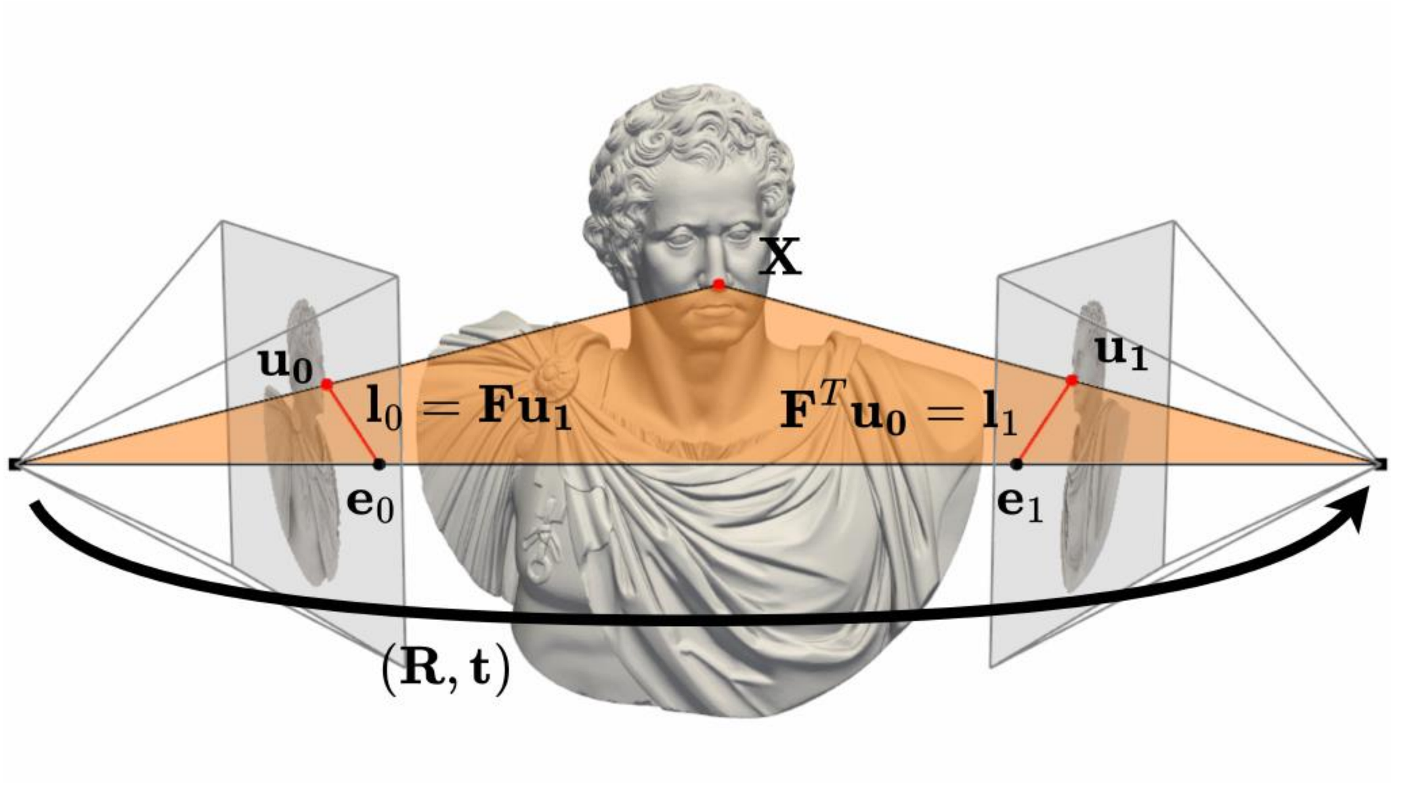
$\mathbf{F}$ is called the fundamental matrix. Similarly to the calibrated case, $\mathbf{F}$ captures information about the geometry of the two cameras:

- $\mathbf{l}_0 = \mathbf{F}\mathbf{u}_1$ is the epipolar line corresponding to $\mathbf{u}_1$.
- $\mathbf{l}_1 = \mathbf{F}^T\mathbf{u}_0$ is the epipolar line corresponding to $\mathbf{u}_0$.
- $\mathbf{F}\mathbf{e}_1 = \mathbf{0} = \mathbf{e}_0^T\mathbf{E}$, i.e. the epipoles are zero-value left and right singular vectors respectively.
- $\mathbf{F}$ has rank 2 and 7 degrees of freedom.

Epipolar Geometry: Summary

Epipolar Constraint

Each correspondence pair $\mathbf{u}_0, \mathbf{u}_1$ must satisfy the epipolar constraint $\mathbf{u}_0^T \mathbf{F} \mathbf{u}_1 = 0$


$\mathbf{X}$
$\mathbf{u_0}$
$\mathbf{u_1}$
$\mathbf{l}_0 = \mathbf{F}\mathbf{u_1}$
$\mathbf{F}^T\mathbf{u_0} = \mathbf{l}_1$
$\mathbf{e}_0$
$\mathbf{e}_1$
$(\mathbf{R}, \mathbf{t})$

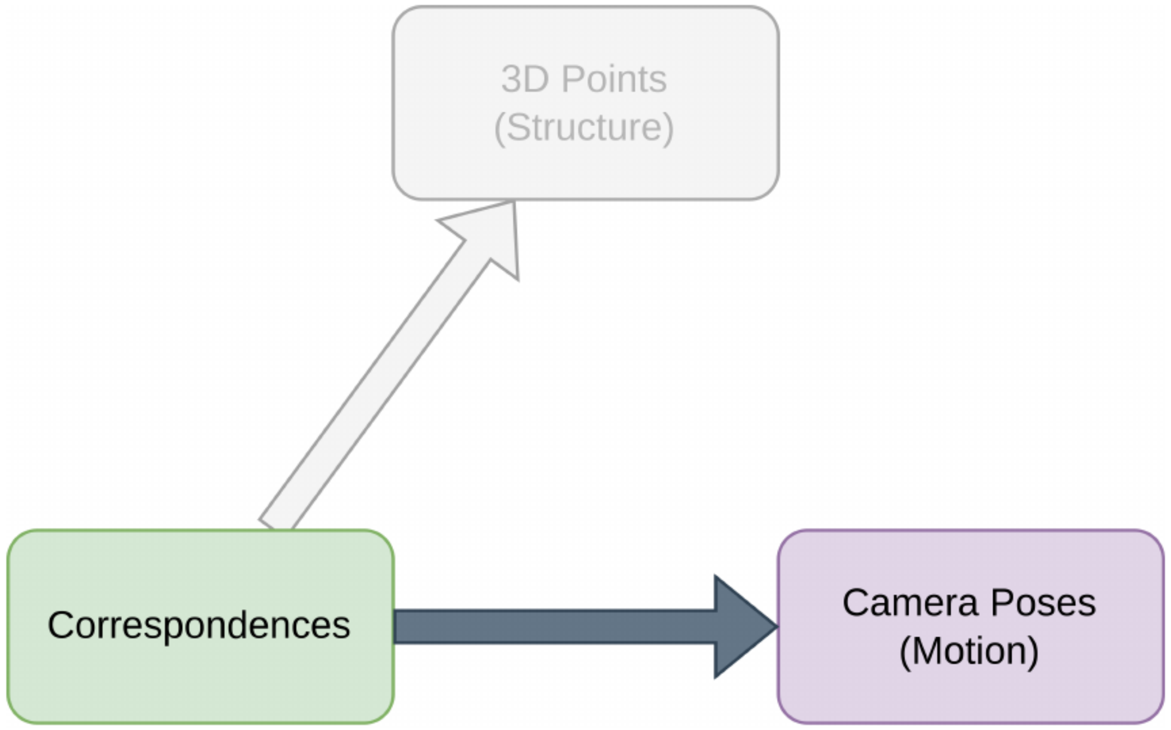
If we know $\mathbf{F}$ or $\mathbf{E}$, we can derive a lot!

Recovering Relative Pose from E

- Our goal was to estimate relative camera poses from 2D-2D correspondences
- Essential matrix is product of $\mathbf{t}$ and $\mathbf{R}$

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

- Can we decompose $\mathbf{E}$ into $\mathbf{t}$ and $\mathbf{R}$?



Recovering Relative Pose from E

Decomposition of E, see Hartley, Zisserman 9.6

The essential matrix has an SVD of the form

$$\mathbf{E} = \mathbf{U}\,\mathrm{diag}(1,1,0)\mathbf{V}^T$$

where

$$\mathbf{U} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix} \text{ and } \mathbf{V}^T = \underbrace{\begin{bmatrix} & -1 & \\ 1 & & \\ & & 1\end{bmatrix}}_{\mathbf{W}}\mathbf{U}^T\mathbf{R}.$$

We can thus recover $\mathbf{R}$ and $\mathbf{t}$ (up to scale) as

- $\mathbf{t} = \mathbf{u}_3$ or $-\mathbf{u}_3$
- $\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^T$ or $\mathbf{U}\mathbf{W}^T\mathbf{V}^T$

The four solutions

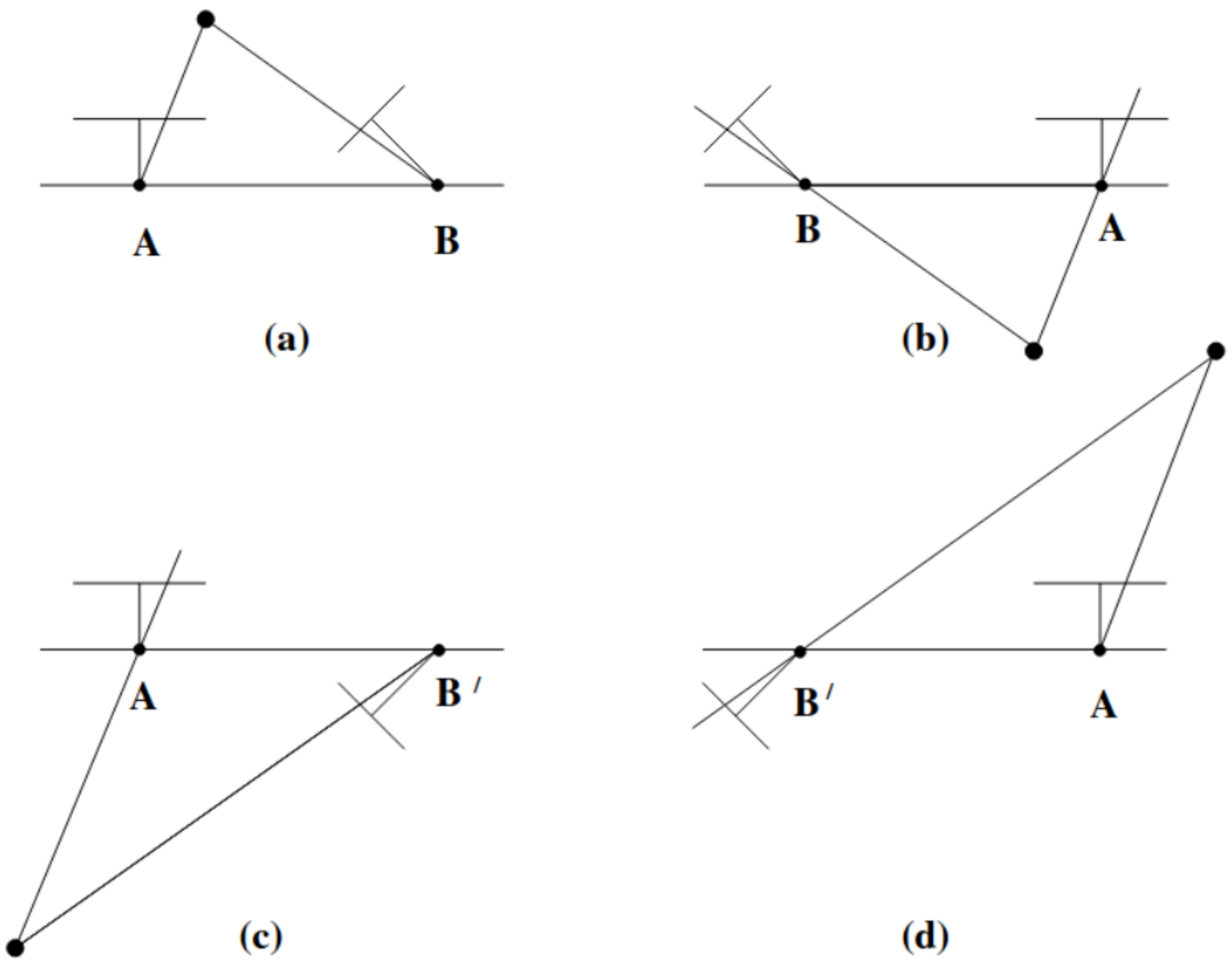


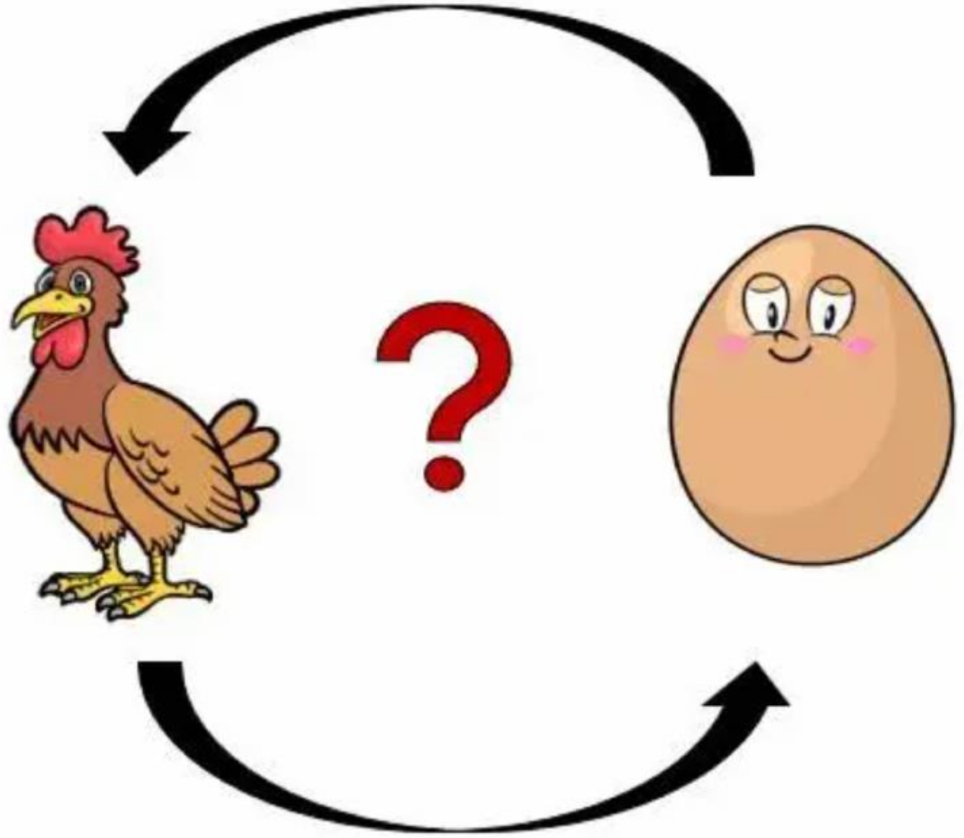
Fig. 9.12. **The four possible solutions for calibrated reconstruction from E.** *Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates* 180° *about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.*

A Chicken and Egg Problem

$$\mathbf{t} = \pm \mathbf{u}_3$$

$$\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^T \mid \mathbf{U}\mathbf{W}^T\mathbf{V}^T$$

$\mathbf{R}, \mathbf{t}$



$\mathbf{E}$

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

Estimating Fundamental / Essential Matrix

Estimating $\mathbf{F}$, $\mathbf{E}$

So far:

- Derived formulas for essential and fundamental matrix and epipolar constraint
- Capture information about epipolar geometry of two cameras
- Can be used to find epipolar lines etc.
- Assumed known relative cameras poses

What we want:

- Relative camera poses are unknown!
- Given a set of correspondences, extract the relative camera poses and 3D points.
- For this, we can estimate fundamental / essential matrix from correspondences

Estimating Essential/Fundamental Matrix from Correspondences

A correspondence pair $\mathbf{u}_0 = (x_0, y_0, 1), \mathbf{u}_1 = (x_1, y_1, 1)$ must satisfy the epipolar constraint $\mathbf{u_0}^T\mathbf{F}\mathbf{u_1} = 0$. This is equivalent to

$$\begin{array}{ccccccc} & x_0 f_{11} x_1 & + & x_0 f_{12} y_1 & + & x_0 f_{13} & \\ + & y_0 f_{21} x_1 & + & y_0 f_{22} y_1 & + & y_0 f_{23} & = 0 \\ + & f_{31} x_1 & + & f_{32} y_1 & + & f_{33} & \end{array}$$

How many correspondences do we need to estimate $\mathbf{F}$?

Estimating Essential/Fundamental Matrix from Correspondences

A correspondence pair $\mathbf{u}_0 = (x_0, y_0, 1), \mathbf{u}_1 = (x_1, y_1, 1)$ must satisfy the epipolar constraint $\mathbf{u_0}^T\mathbf{F}\mathbf{u_1} = 0$. This is equivalent to

$$\begin{array}{ccccccl} & x_0 f_{11} x_1 & + & x_0 f_{12} y_1 & + & x_0 f_{13} & \\ + & y_0 f_{21} x_1 & + & y_0 f_{22} y_1 & + & y_0 f_{23} & = 0 \\ + & f_{31} x_1 & + & f_{32} y_1 & + & f_{33} & \end{array}$$

How many correspondences do we need to estimate $\mathbf{F}$?

- For fundamental matrix 8 points suffice → 8 Point Algorithm
- $\mathbf{F}$ has 7 degrees of freedom
- Even 7 points suffice (but no longer linear) → 7 Point Algorithm
- For the essential matrix even 5 points are enough → 5 Point Algorithm

The 8-Point Algorithm

Given M correspondences (e.g. $M = 8$, but more = better) we can rewrite the M linear equations in the form

$$\begin{bmatrix} x_{1,0}x_{1,1} & x_{1,1}y_{1,1} & x_{1,0} & y_{1,0}x_{1,1} & y_{1,0}y_{1,1} & y_{1,0} & x_{1,1} & y_{1,1} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{M,0}x_{M,1} & x_{M,1}y_{M,1} & x_{M,0} & y_{M,0}x_{M,1} & y_{M,0}y_{M,1} & y_{M,0} & x_{M,1} & y_{M,1} & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ \vdots \\ f_{33} \end{bmatrix} = \mathbf{W}\mathbf{f} = 0$$

In practice we use $M > 8$ correspondences to reduce the effect of noisy measurements.

How do we solve this in the least squares sense?

$$\min_{\|\mathbf{f}\|=1} \|\mathbf{W}\mathbf{f}\|_2^2$$

The 8-Point Algorithm

Given M correspondences (e.g. $M = 8$, but more = better) we can rewrite the M linear equations in the form

$$\begin{bmatrix} x_{1,0}x_{1,1} & x_{1,1}y_{1,1} & x_{1,0} & y_{1,0}x_{1,1} & y_{1,0}y_{1,1} & y_{1,0} & x_{1,1} & y_{1,1} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{M,0}x_{M,1} & x_{M,1}y_{M,1} & x_{M,0} & y_{M,0}x_{M,1} & y_{M,0}y_{M,1} & y_{M,0} & x_{M,1} & y_{M,1} & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ \vdots \\ f_{33} \end{bmatrix} = \mathbf{W}\mathbf{f} = 0$$

In practice we use $M > 8$ correspondences to reduce the effect of noisy measurements.

How do we solve this in the least squares sense?

$$\min_{\|\mathbf{f}\|=1} \|\mathbf{W}\mathbf{f}\|_2^2$$

SVD!

The 8-Point Algorithm

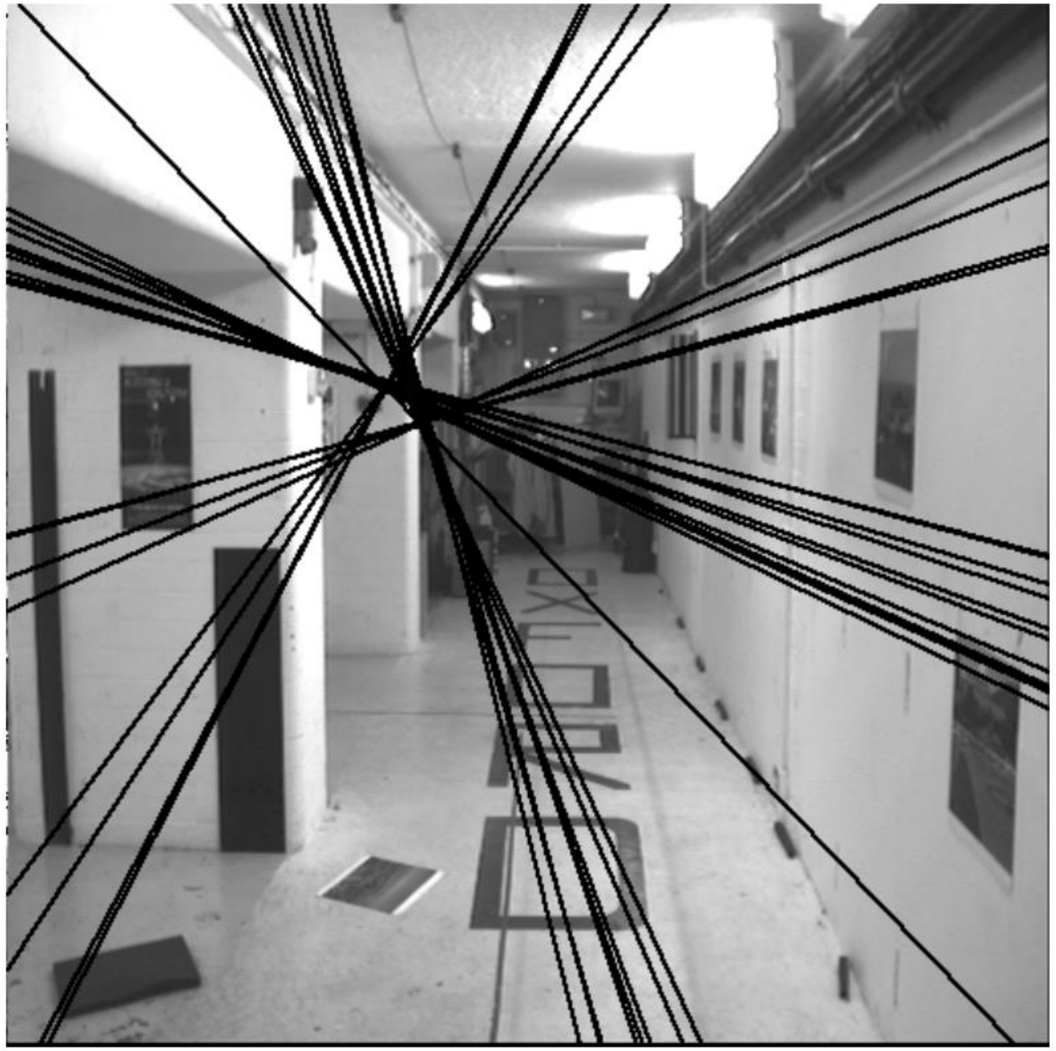
Compute a SVD of $\mathbf{W}$:

$$\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

The least squares solution is given by the right-singular vector corresponding to the smallest singular value!

We get estimate $\hat{\mathbf{F}}$ from $\mathbf{f}$ by reshaping.

Problems?



The 8-Point Algorithm

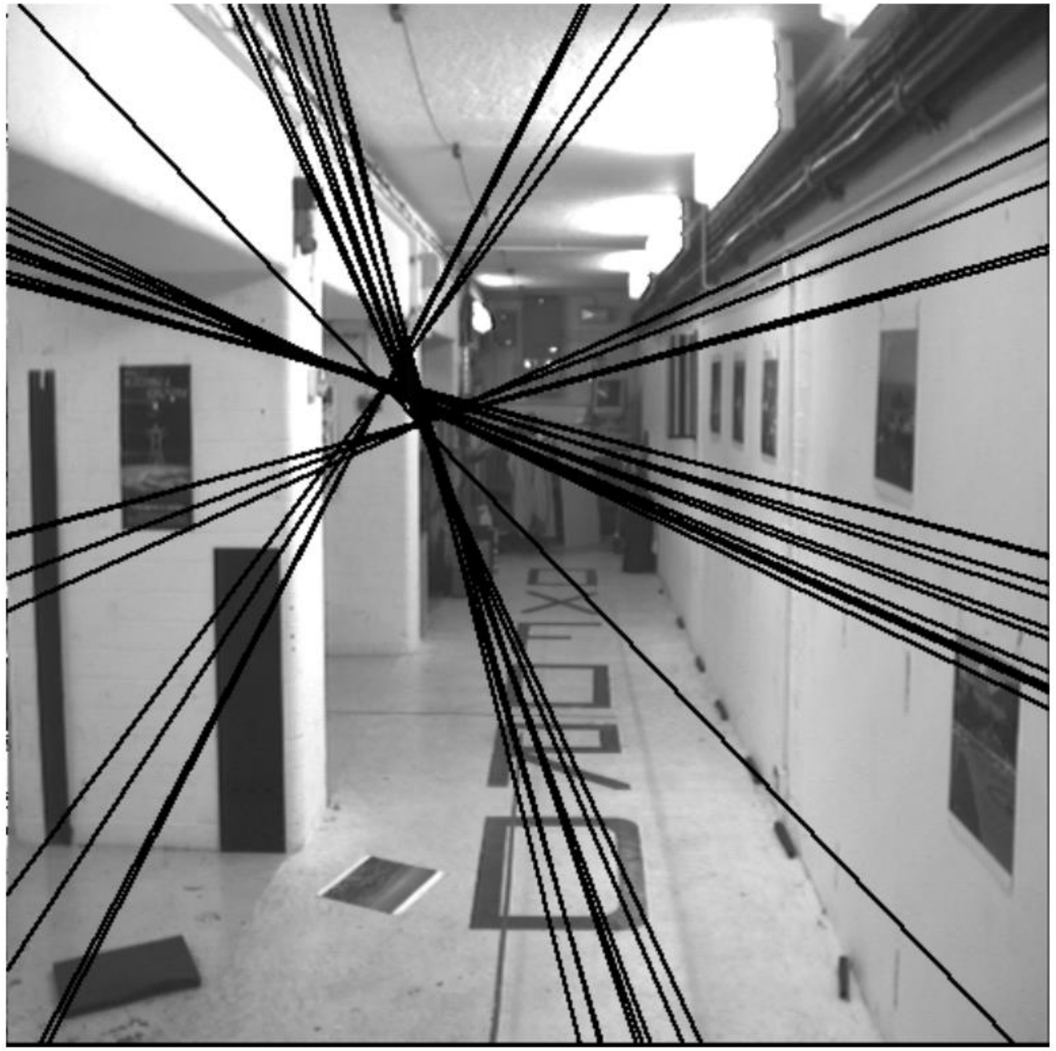
Compute a SVD of $\mathbf{W}$:

$$\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

The least squares solution is given by the right-singular vector corresponding to the smallest singular value!

We get estimate $\hat{\mathbf{F}}$ from $\mathbf{f}$ by reshaping.

Problem: $\hat{\mathbf{F}}$ may have rank full rank. But essential/fundamental matrix has rank 2!



The 8-Point Algorithm

Goal: Find best rank 2 approximation of $\hat{\mathbf{F}}$:

$$\min_{\det \mathbf{F}=0} \|\mathbf{F} - \hat{\mathbf{F}}\|_F$$

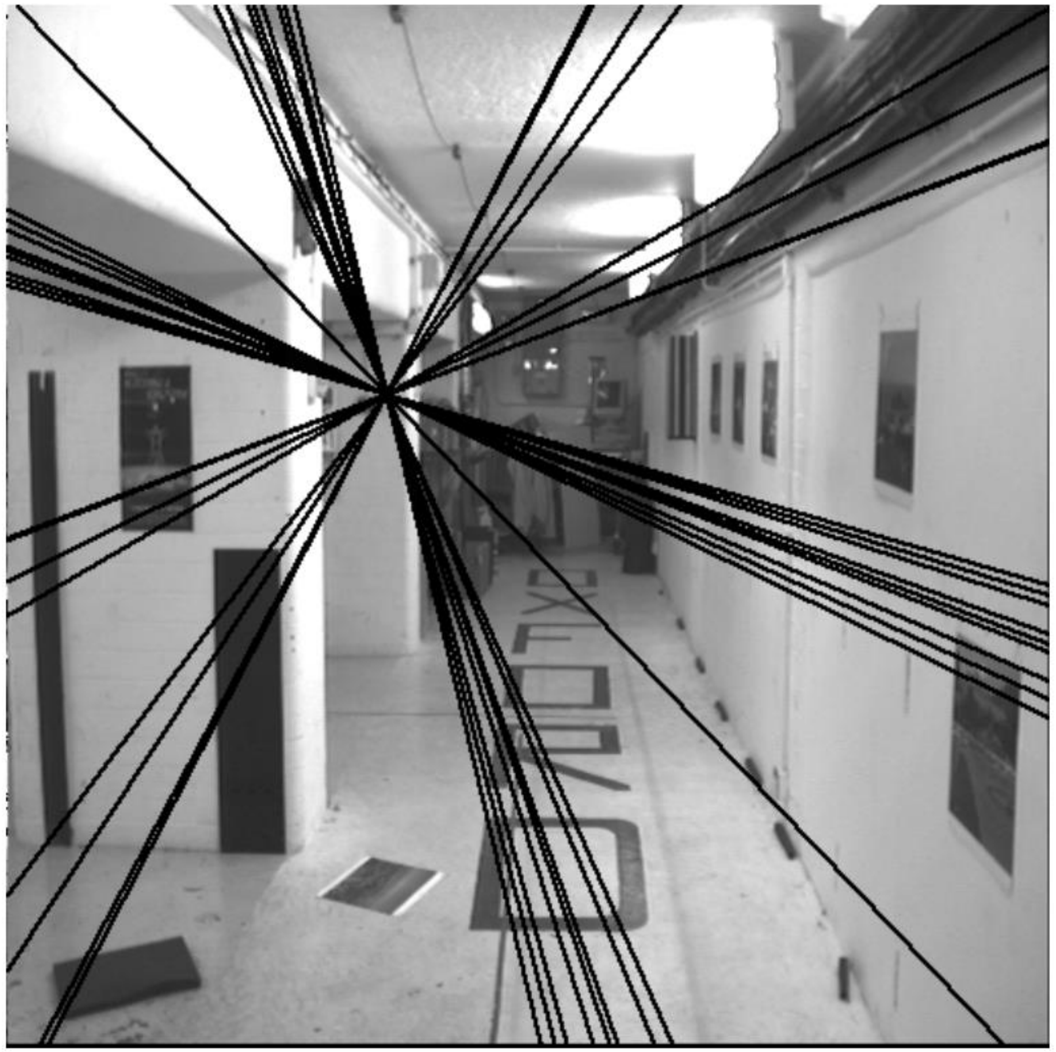
Solution:

- Compute SVD of estimate:

$$\hat{\mathbf{F}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

- Keep only the two largest singular values:

$$\mathbf{F} = \mathbf{U}\begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & 0 \end{bmatrix}\mathbf{V}^T$$



The 8-Point Algorithm: Normalization

One row of $\mathbf{W}$:

$$\begin{bmatrix} x_{1,0}x_{1,1} & x_{1,1}y_{1,1} & x_{1,0} & y_{1,0}x_{1,1} & y_{1,0}y_{1,1} & y_{1,0} & x_{1,1} & y_{1,1} & 1 \end{bmatrix}$$

Typically $x, y \in [0, 2000]$

- Entries differ in order of magnitudes!
- Highly un-balanced and not well conditioned
- Creates problems during SVD

Solution: Transform image coordinates such that $\mathbf{W}$ becomes better conditioned

The Normalized 8-Point Algorithm

For each image i, apply a transformation $\mathbf{T}_i$ as follows

- translate image points s.th. their centroid is at the origin
- uniformly scale image points s.th. mean squared distance of image points from origin is ~ 2px.

New homogeneous coordinates $\bar{\mathbf{x}}_i = \mathbf{T}_i \mathbf{x}_i$.
Epipolar constraint:

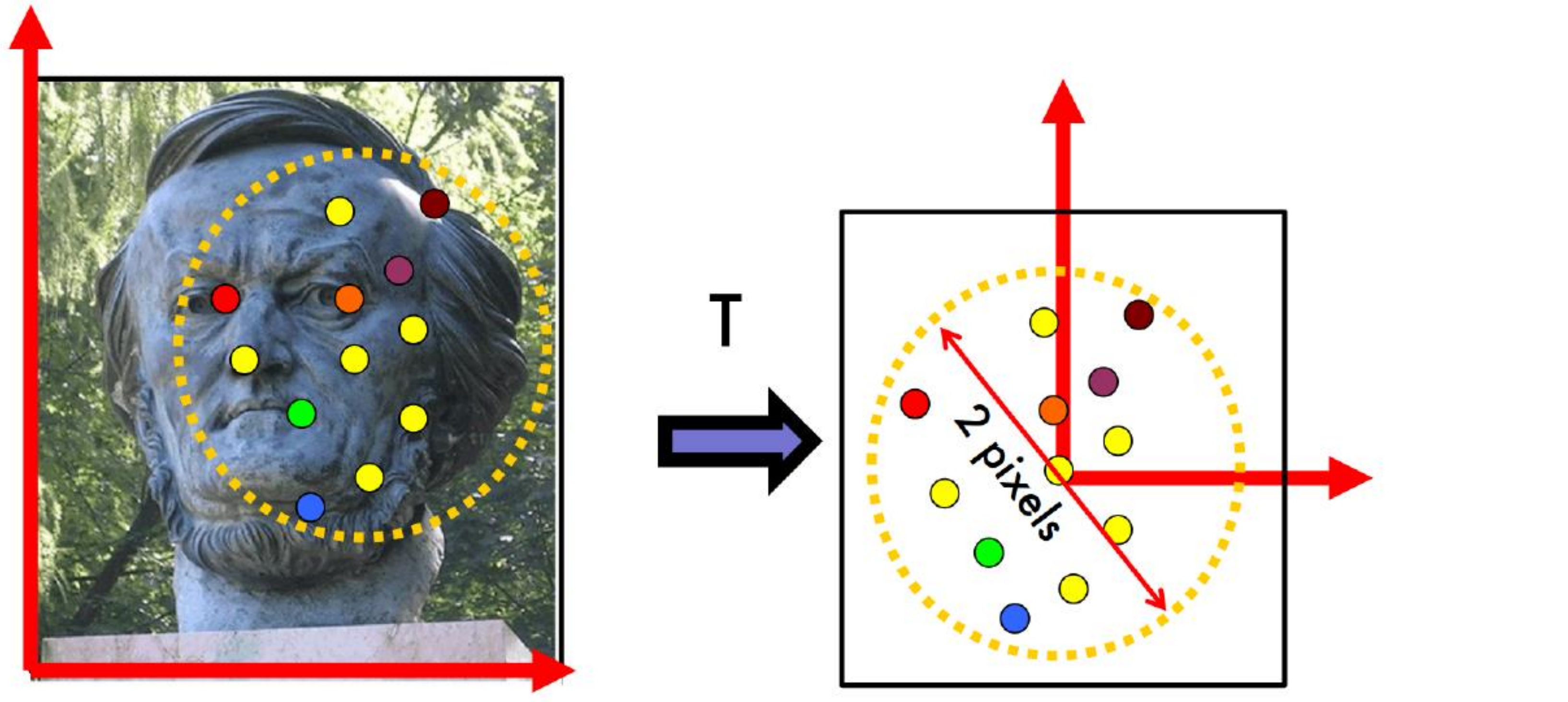
$$0 = \bar{\mathbf{x}}_0^T \mathbf{T}_0^{-T} \mathbf{F} \mathbf{T}_1^{-1} \bar{\mathbf{x}}_1 = \bar{\mathbf{x}}_0^T \bar{\mathbf{F}} \bar{\mathbf{x}}_1$$

where $\bar{\mathbf{F}} = \mathbf{T}_0^{-T} \mathbf{F} \mathbf{T}_1^{-1}$.

$\rightarrow$ we can use 8-Point Algorithm to estimate $\bar{\mathbf{F}}$, and from this recover $\mathbf{F}$ according to

$$\mathbf{F} = \mathbf{T}_0^T \bar{\mathbf{F}} \mathbf{T}_1$$

Normalization



Coordinate system of the image before applying T

Coordinate system of the image after applying T

The 8-Point Algorithm: Summary

1. Normalize points
2. Construct the $M \times 9$ matrix $\mathbf{W}$
3. Find SVD of $\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$
4. Entries of $\hat{\mathbf{F}}$ are the elements of the column of $\mathbf{V}$ corresponding to the smallest singular value
5. Find SVD of $\hat{\mathbf{F}} = \mathbf{U}'\,\boldsymbol{\Sigma}'\mathbf{V}'^T$
6. Set $\mathbf{F} = \mathbf{U}\,\mathrm{diag}(\sigma_1', \sigma_2', 0)\mathbf{V}'^T$
7. Denormalize $\mathbf{F}$

RANSAC: Robustness to Outliers

In case of contaminated correspondences, least squares is very unstable. Outliers have very strong effect because of squared norm.

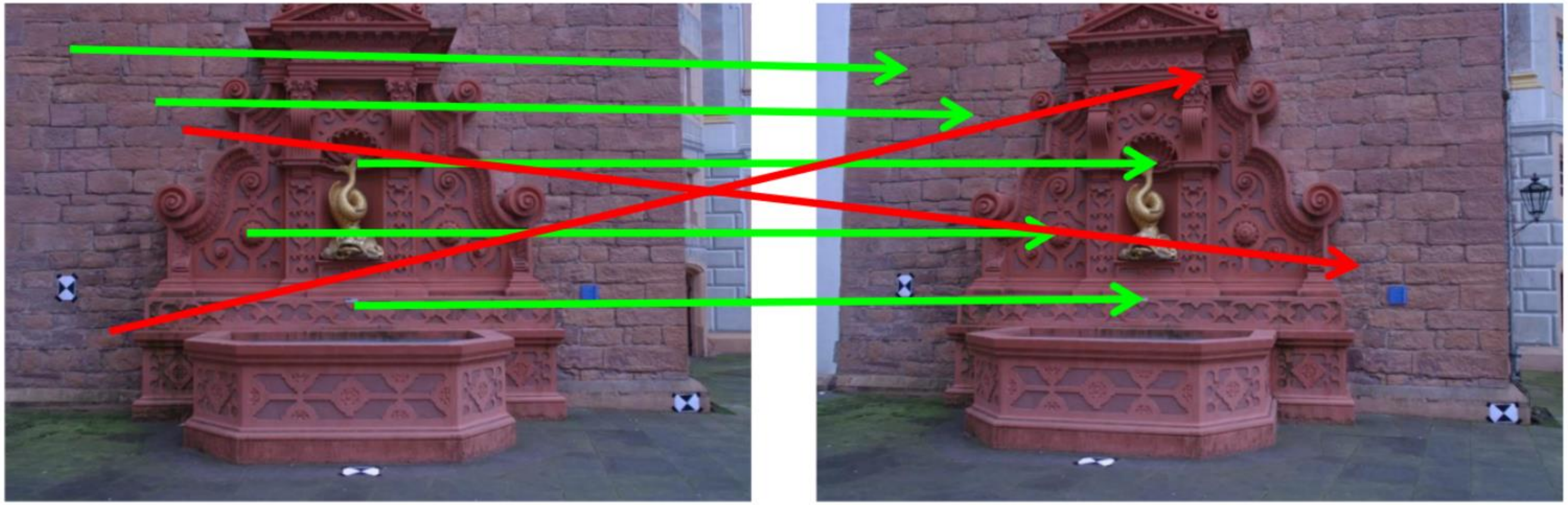
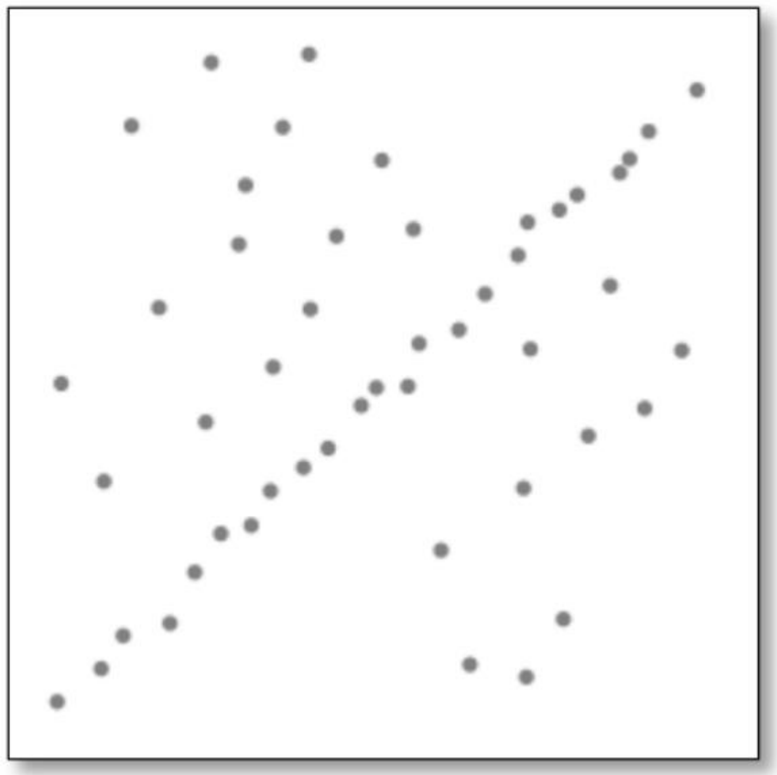
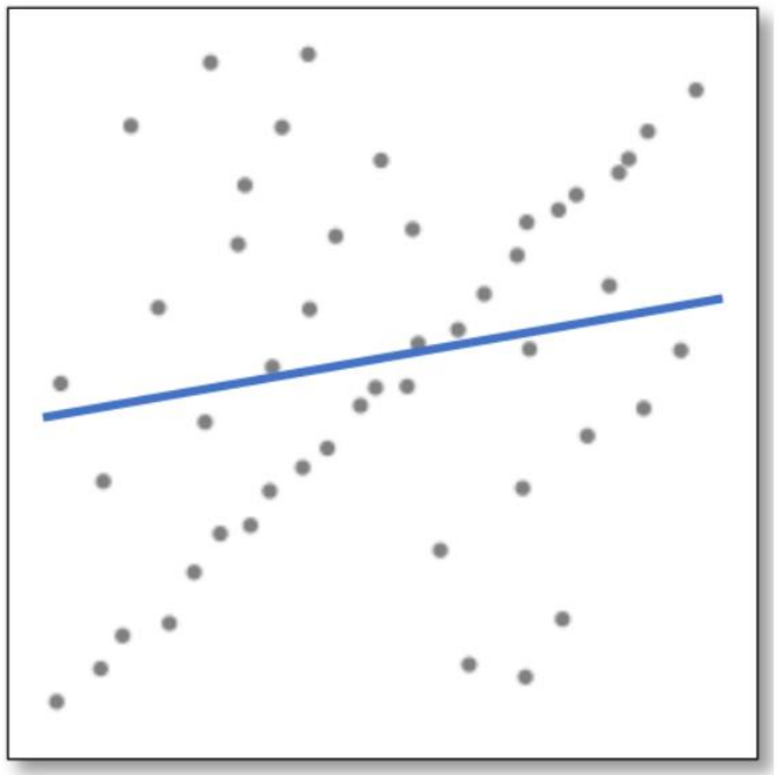


Image 1 Image 2

RANdom SAmpling Consensus (RANSAC)

RANSAC: Line fitting example


Problem: Fit a line to these datapoints

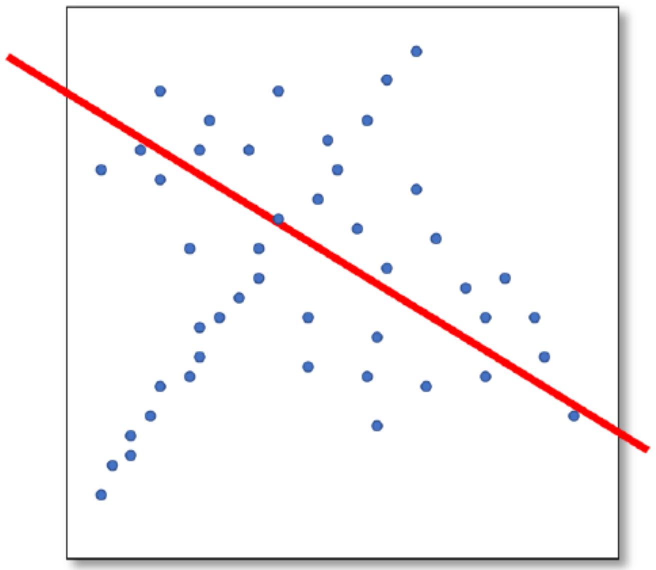

Least squares fit

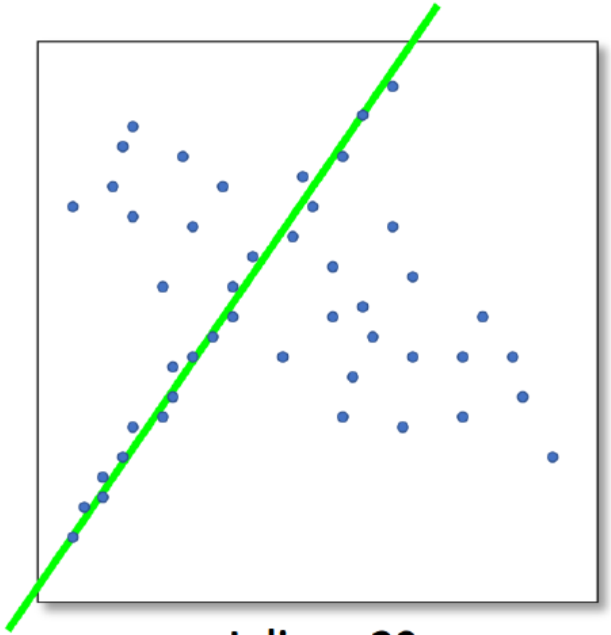
RANSAC: Idea

For a given model we can count the number of samples that ”agree” with this model
Among all possible models, select the one with which the most samples agree.

Intuition:

- If we estimate a model only from inliers, then most of the inliers will agree
- If we estimate with outliers, only few samples agree


Inliers: 3


Inliers: 20

RANSAC: Algorithm

1. Randomly choose s samples
   - Typically s = min sample size that lets you fit a model
   - Line fitting: 2, Essential matrix estimation: 5
2. Fit a model to those samples
3. Count the number of inliers
4. Update best solution found so far
5. Repeat

How many iterations?

- If we assume an outlier ratio e
- And we desire correctness probability p

$$N \geq \frac{\log(1-p)}{\log(1-(1-e)^2)}$$

From Fundamental to Essential Matrix

- We now have a robust way of computing the fundamental matrix $\mathbf{F}$ via the 8-Point Algorithm.
- We assume that we are able to recover intrinsic camera matrices $\mathbf{K}_0, \mathbf{K}_1$ (often come as part of metadata)
- Thus, we can recover the essential matrix from

$$\mathbf{F} = \mathbf{K}_0^{-T}\mathbf{E}\mathbf{K}_1^{-1}$$

- We know that an ideal essential matrix has two identical singular values, i.e. up to scale

$$\mathbf{E} = \mathbf{U}\,\mathrm{diag}(1, 1, 0)\mathbf{V}^T$$

- Project estimated $\mathbf{E}$ such that singular values are 1.

Hartley, Zisserman, Section 9.6

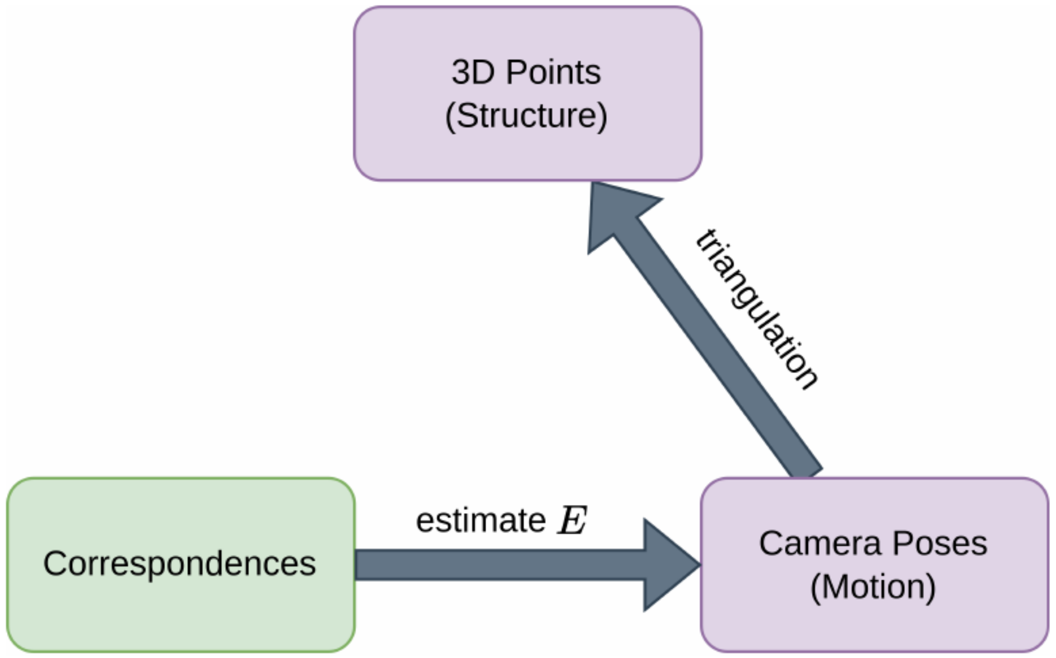
Bundle Adjustment and SfM

From two views to N views

So far: Use epipolar geometry to go from correspondences on two images to

- relative cameras poses (8-Point Algorithm)
- 3D points (Triangulation)

Extensions to more views possible (trifocal, quadrifocal tensor).



Instead, SfM pipelines often incrementally integrate images into a model, and update model with new observation.

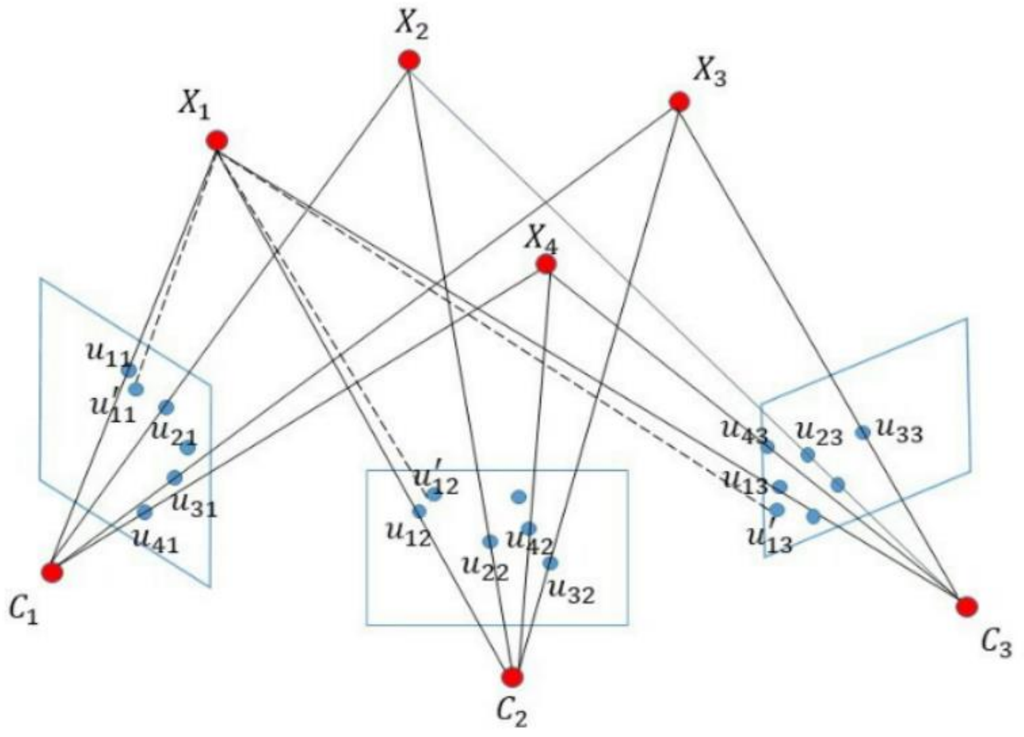
Integral ingredient: Bundle Adjustment

Bundle Adjustment

Minimizes total reprojection errors:

$$\min_{\mathbf{X},\mathbf{P}} \sum_{i,j} w_{i,j} \|\mathbf{u}_{i,j} - \pi(\mathbf{P}_i, \mathbf{X}_j)\|_2^2$$

- $w_{i,j}$ indicates whether point j is visible in camera i
- $\mathbf{u}_{i,j}$ is the 2D image observation of point j in camera i
- $\pi(P_i, X_j)$ is the projection of point j onto image plane i



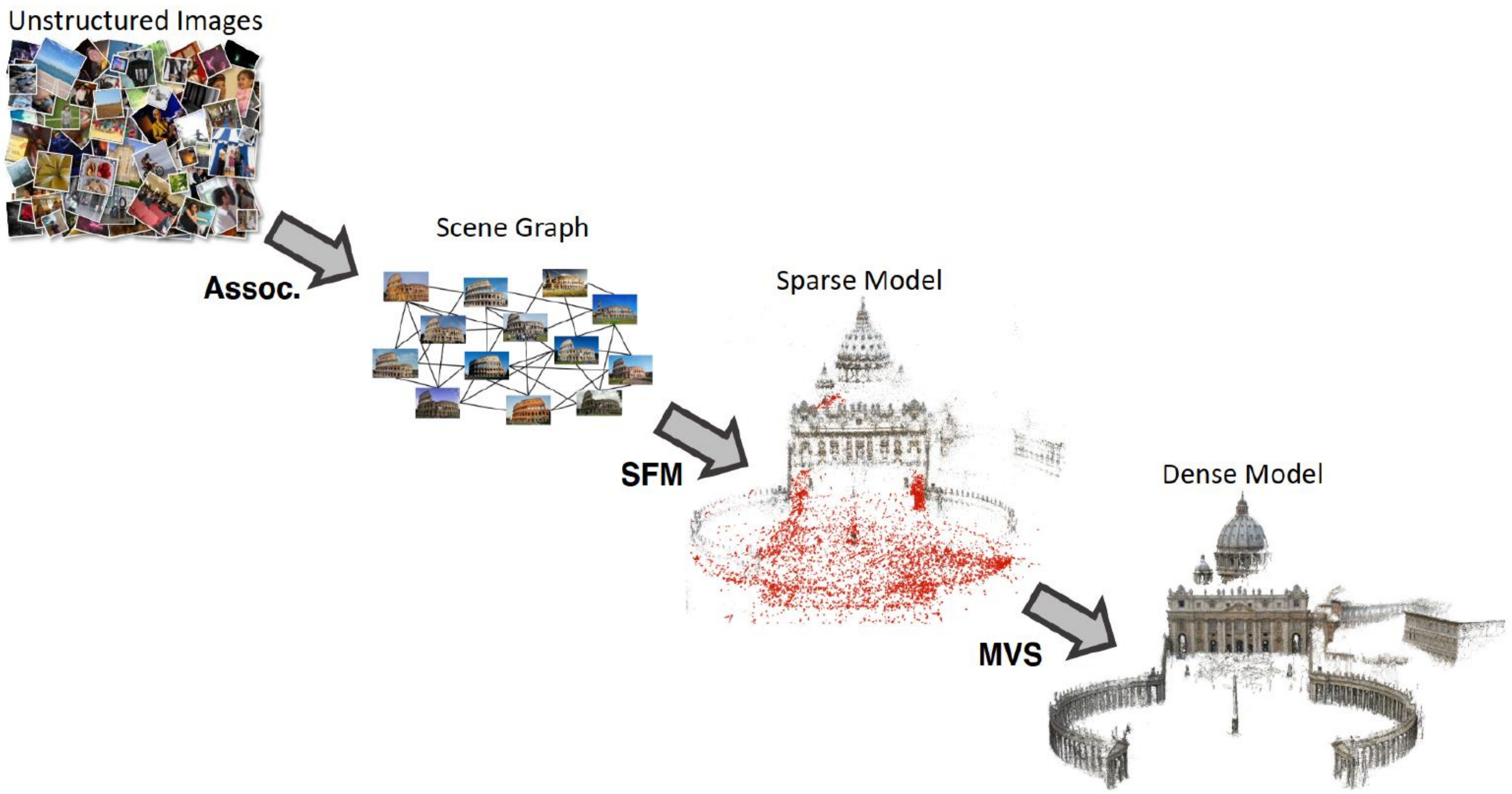
Bundle Adjustment

Why can't we just solve SfM using Bundle Adjustment?

- Typically, this minimization problem is solved using Gauß-Newton or Levenberg–Marquardt. For large systems, efficient open surce tools exist (e.g. Google Ceres).
- SfM implementations can take advantage of sparse structure, but still time consuming
- Exact methods: cubic runtime, approximate methods linear runtime
- Sensitive to initialization

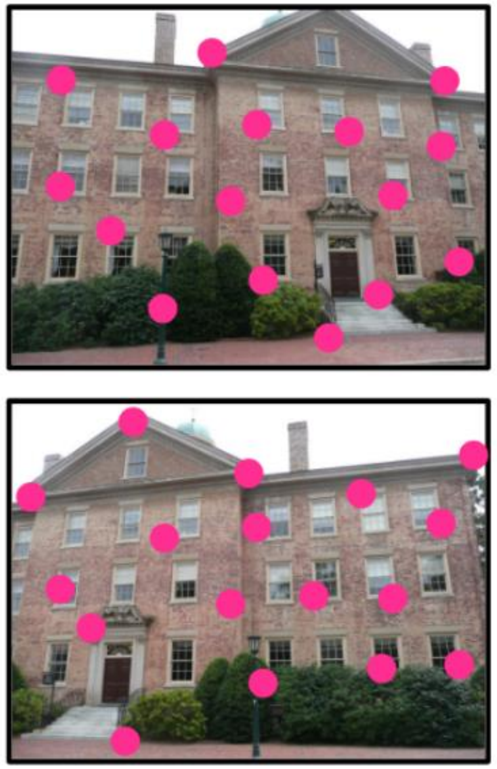
Therefore, Bundle Adjustment is only used as a refinement step in SfM pipelines.

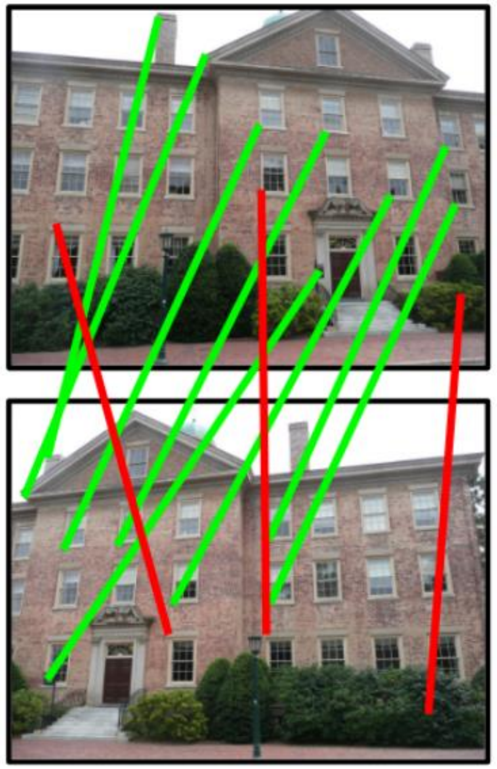
3D Reconstruction Pipeline

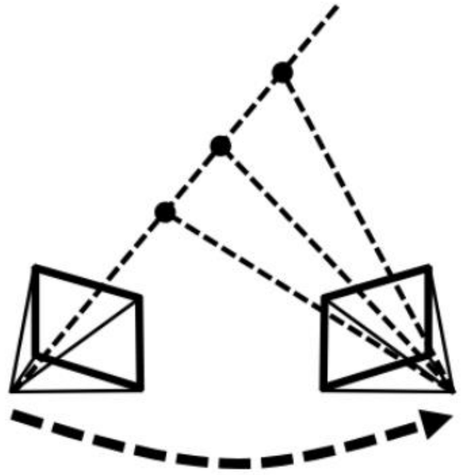


Data Association

1. Feature Extraction → 2. Feature Matching → 3. Geometric Verification

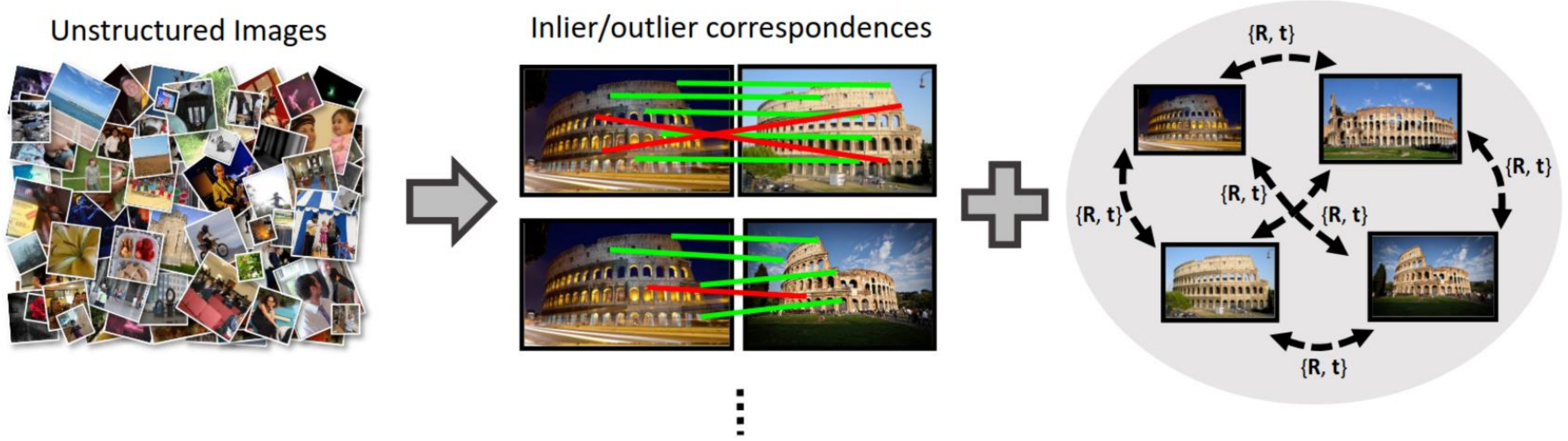






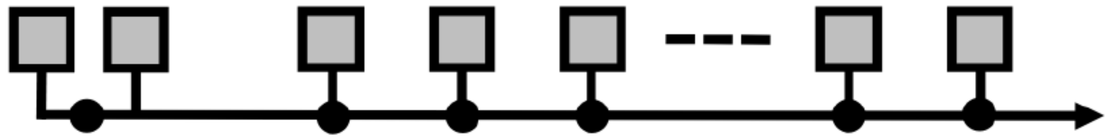
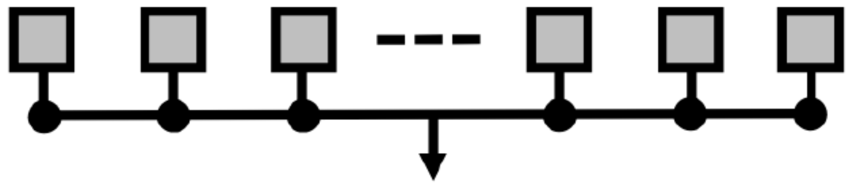
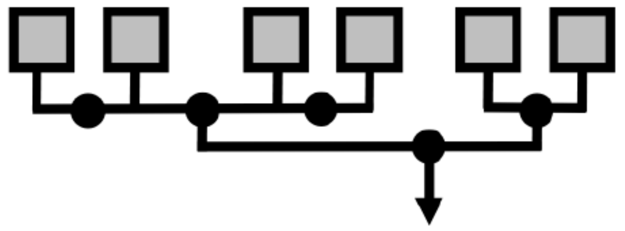
Robust Two-View Geometry Estimation

Scene Graph

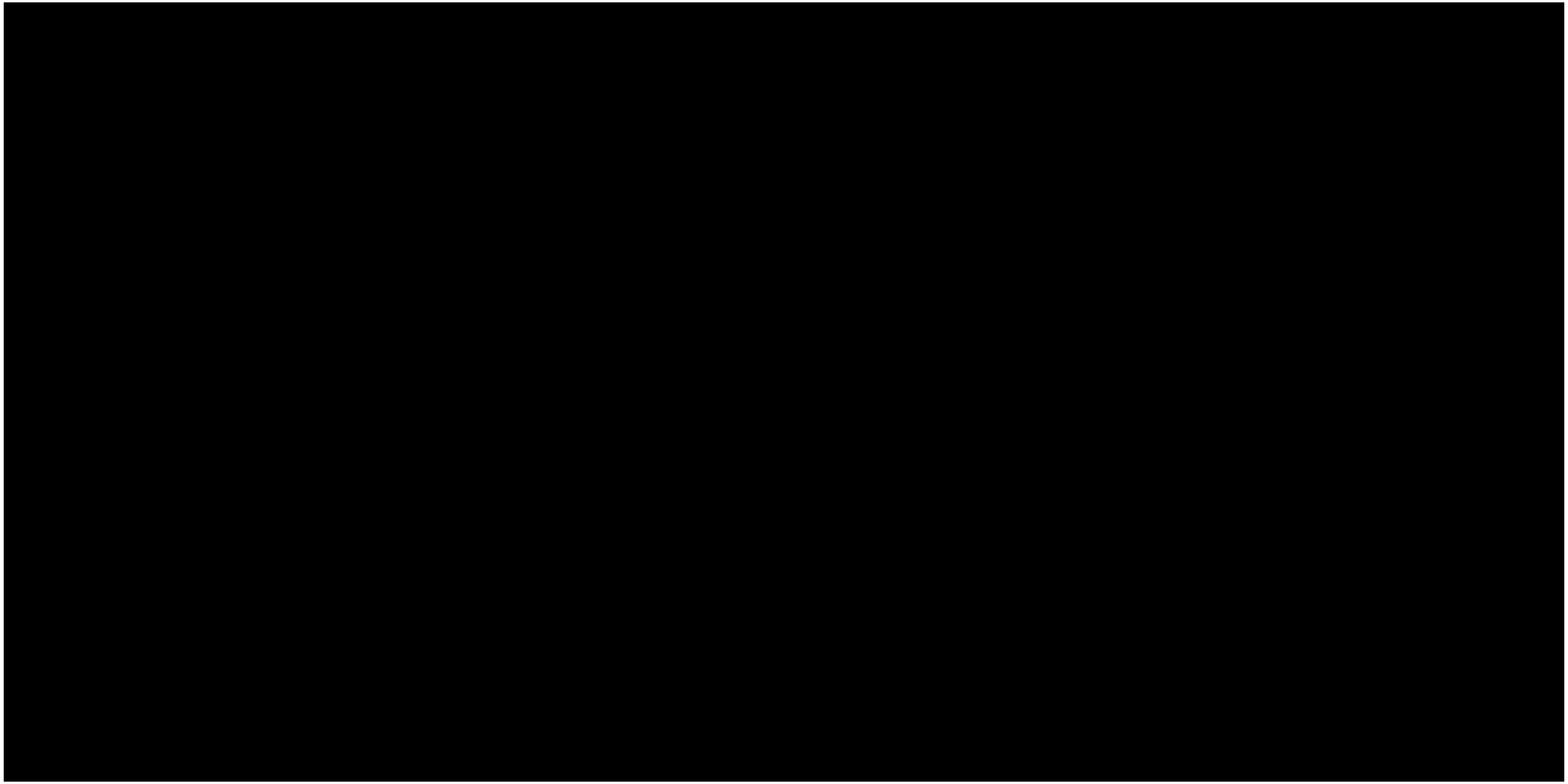

Unstructured Images
Inlier/outlier correspondences
{R, t}
{R, t}
{R, t}
{R, t}
{R, t}
{R, t}

Structure from Motion Paradigms

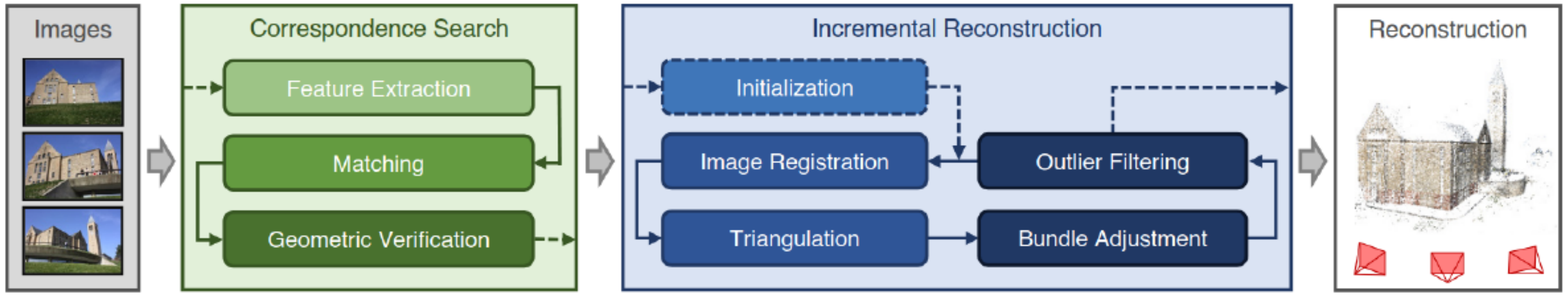
- 3 paradigms
  - Incremental
  - Global
  - Hierarchical

Incremental SfM



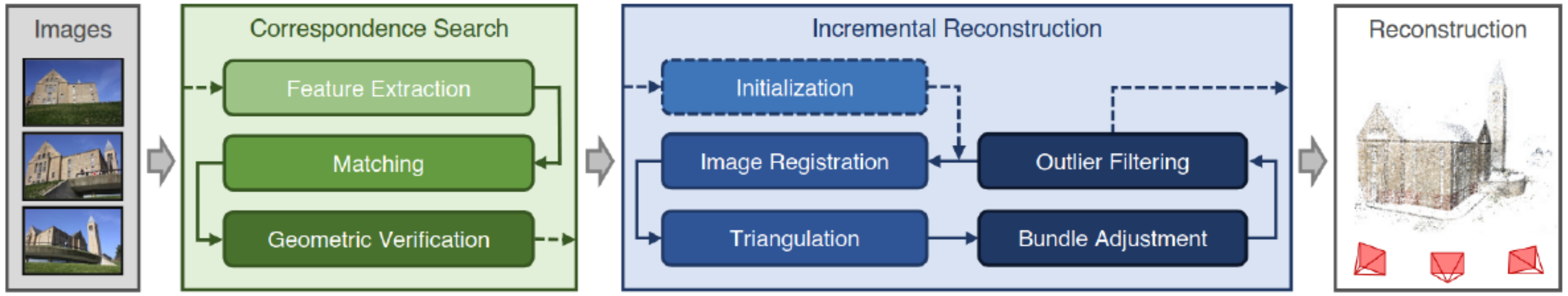
Incremental SfM



Initialization:

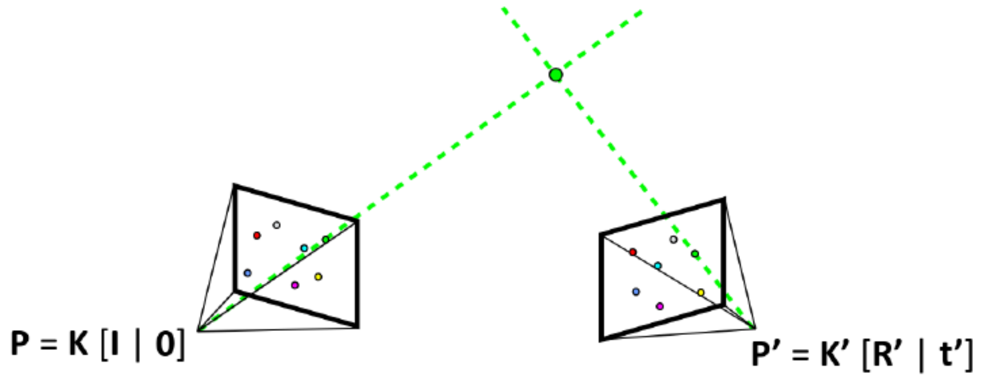
- Pick a pair of images with lots of inliers
- Estimate extrinsic parameters $\mathbf{R}, \mathbf{t}$ using robust 5-Point or 8-Point algorithm
- Triangulate to initialize 3D points of correspondences
- Refine using Bundle Adjustment

Incremental SfM

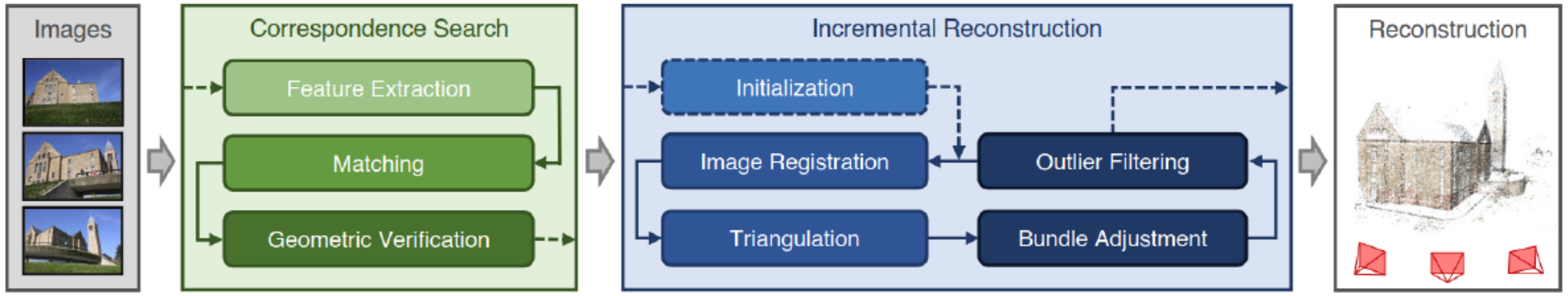


Initialization:

- Pick a pair of images with lots of inliers
- Estimate extrinsic parameters $\mathbf{R}, \mathbf{t}$ using robust 5-Point or 8-Point algorithm
- Triangulate to initialize 3D points of correspondences
- Refine using Bundle Adjustment

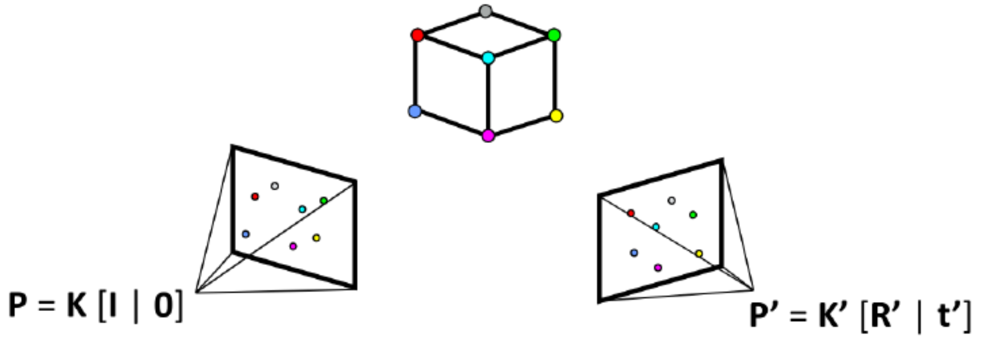


Incremental SfM

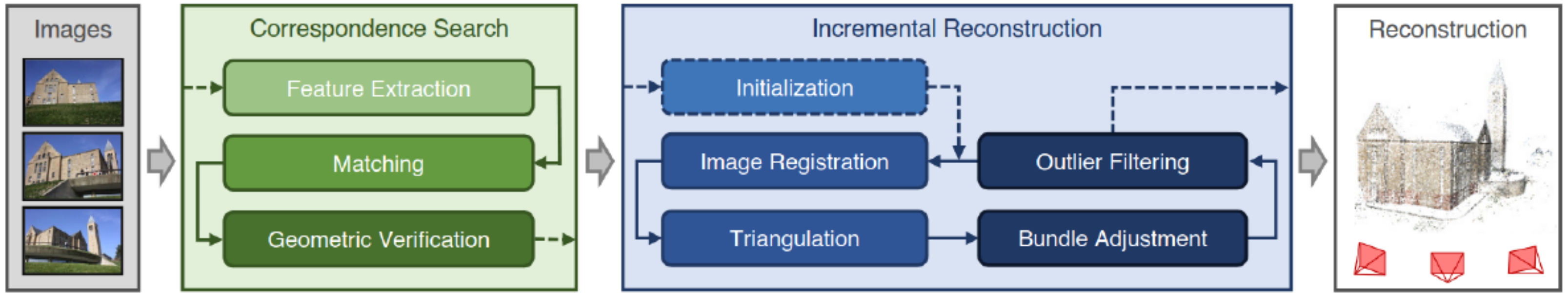


Initialization:

- Pick a pair of images with lots of inliers
- Estimate extrinsic parameters $\mathbf{R}, \mathbf{t}$ using robust 5-Point or 8-Point algorithm
- Triangulate to initialize 3D points of correspondences
- Refine using Bundle Adjustment

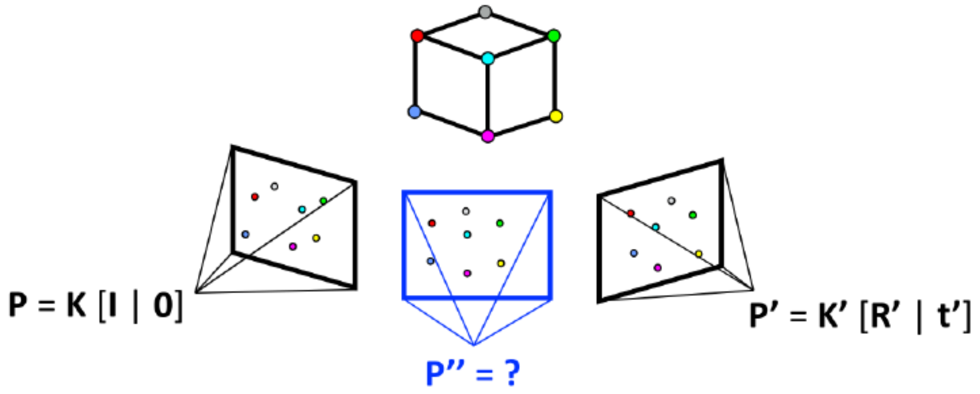


Incremental SfM

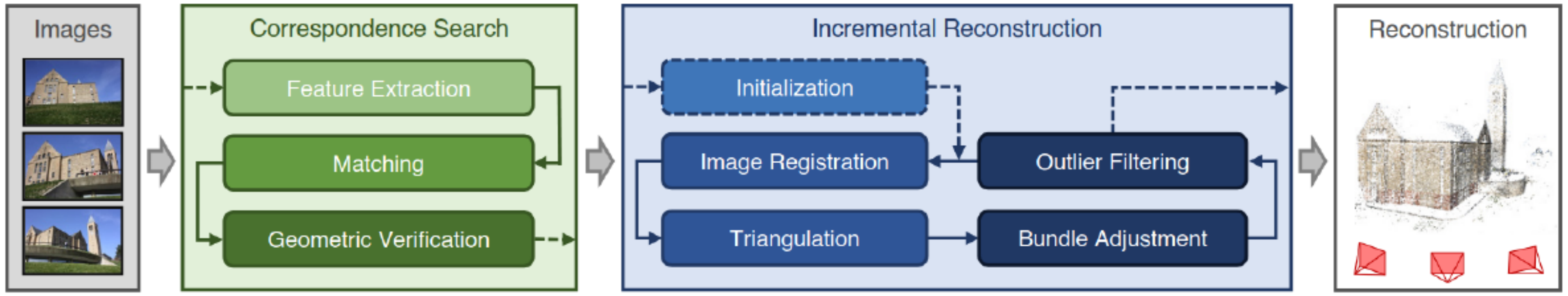


While there are remaining images:

- Find image with many feature matches

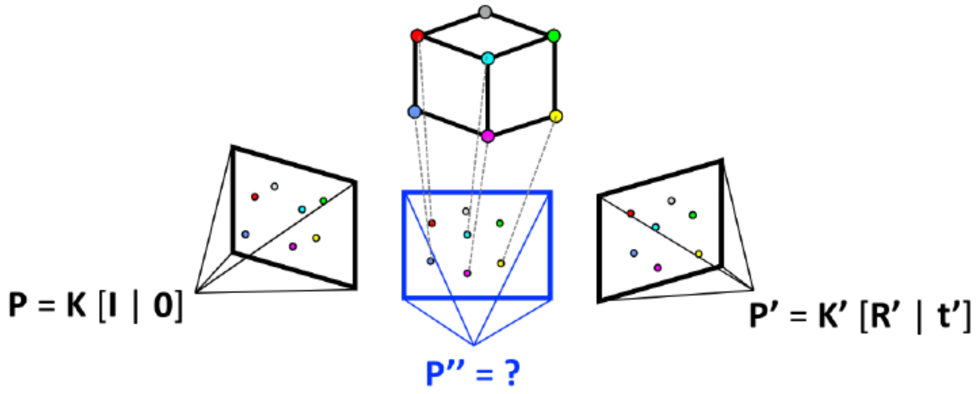


Incremental SfM

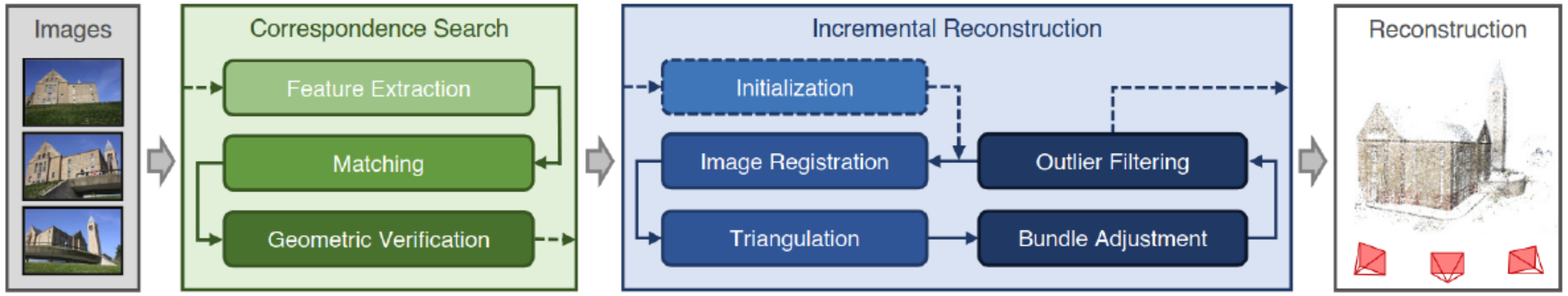


While there are remaining images:

- Find image with many feature matches
- Find 2D-3D correspondences

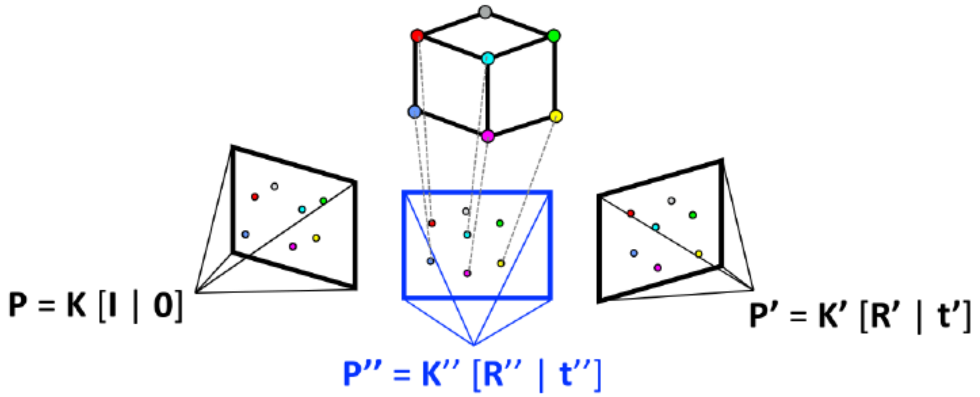


Incremental SfM

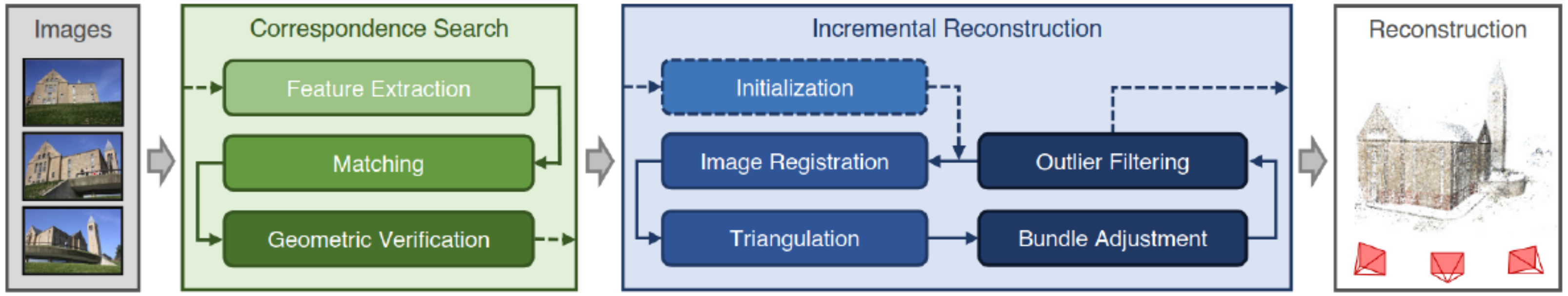


While there are remaining images:

- Find image with many feature matches
- Find 2D-3D correspondences
- Estimate camera pose to obtain camera registration

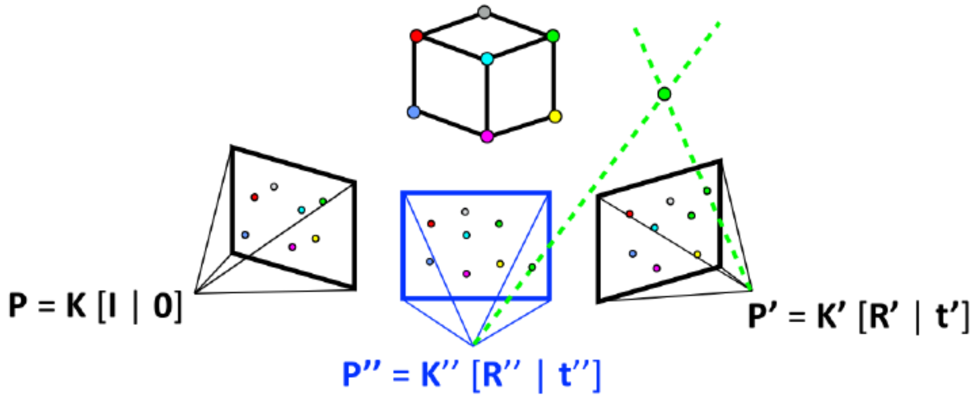


Incremental SfM



While there are remaining images:

- Find image with many feature matches
- Find 2D-3D correspondences
- Estimate camera pose to obtain camera registration
- Triangulate new points
- Refine using bundle adjustment
- Filter outliers



Incremental SfM: Challenges

Initial pair must be chosen carefully

- bad choice → unrecoverable local minimum
- cameras should be far apart for robust initialization
- dense feature regions: more robust because redundancies, but BA slower

Next Best View Problem

- almost identical view → high uncertainty in triangulation
- very different view → low overlap and high camera uncertainty
- single bad choice may impact whole reconstruction

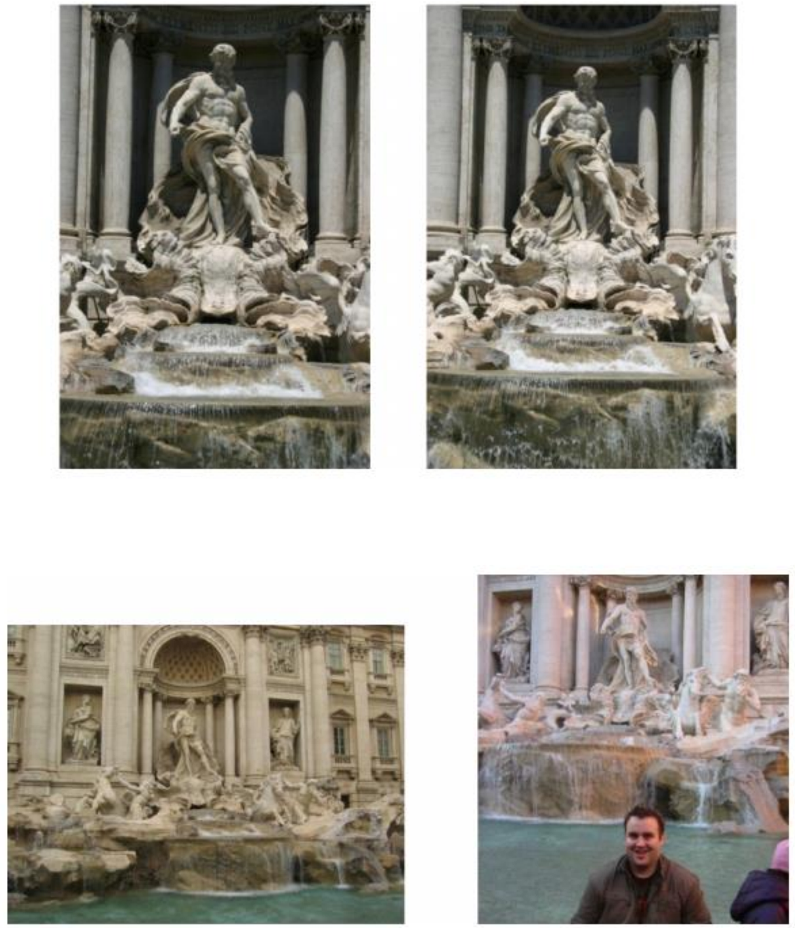
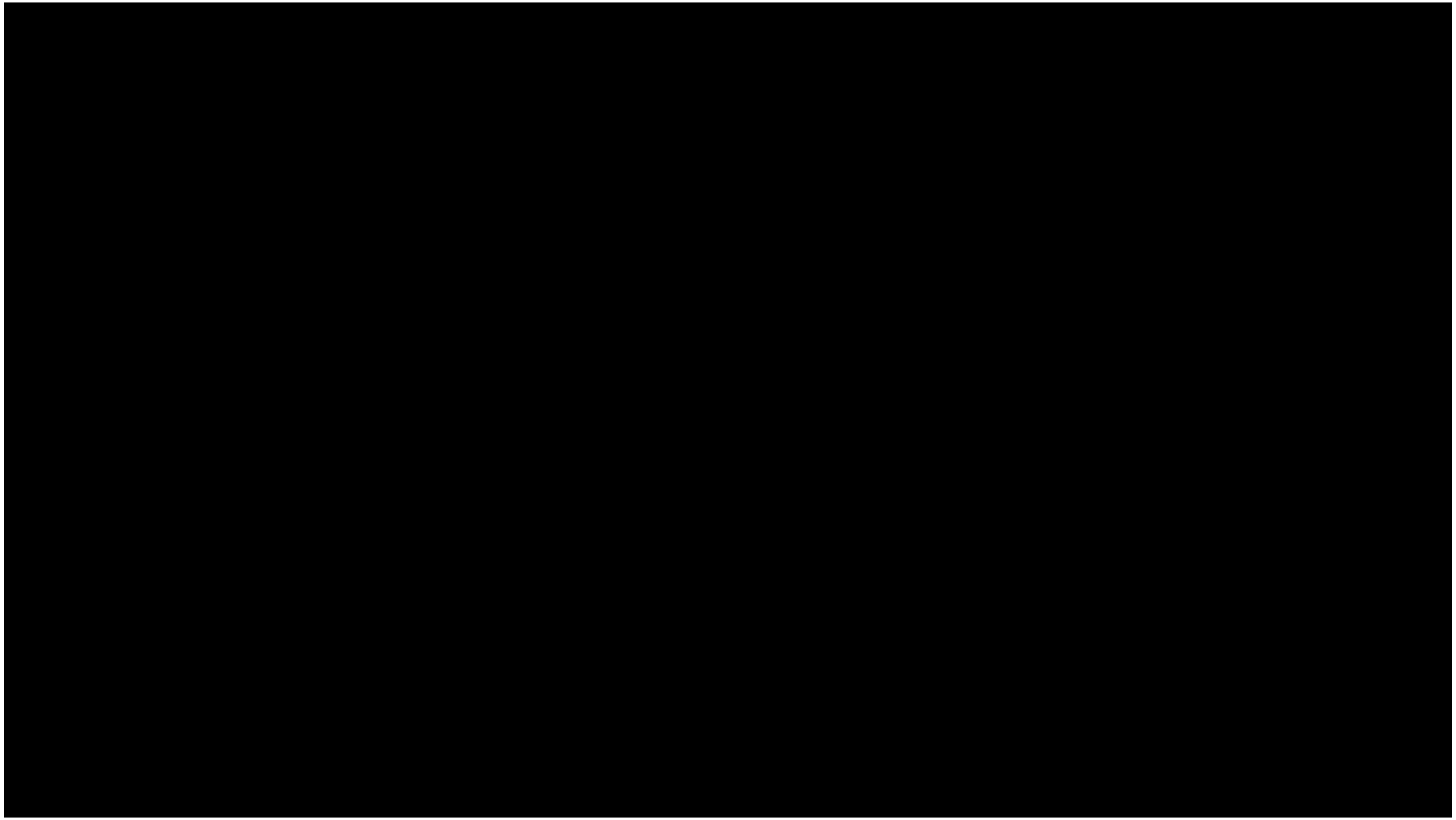


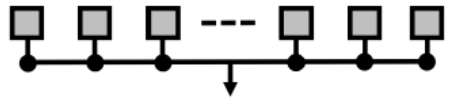
Figure: Bad init (top) vs good init (bottom)

Global SfM

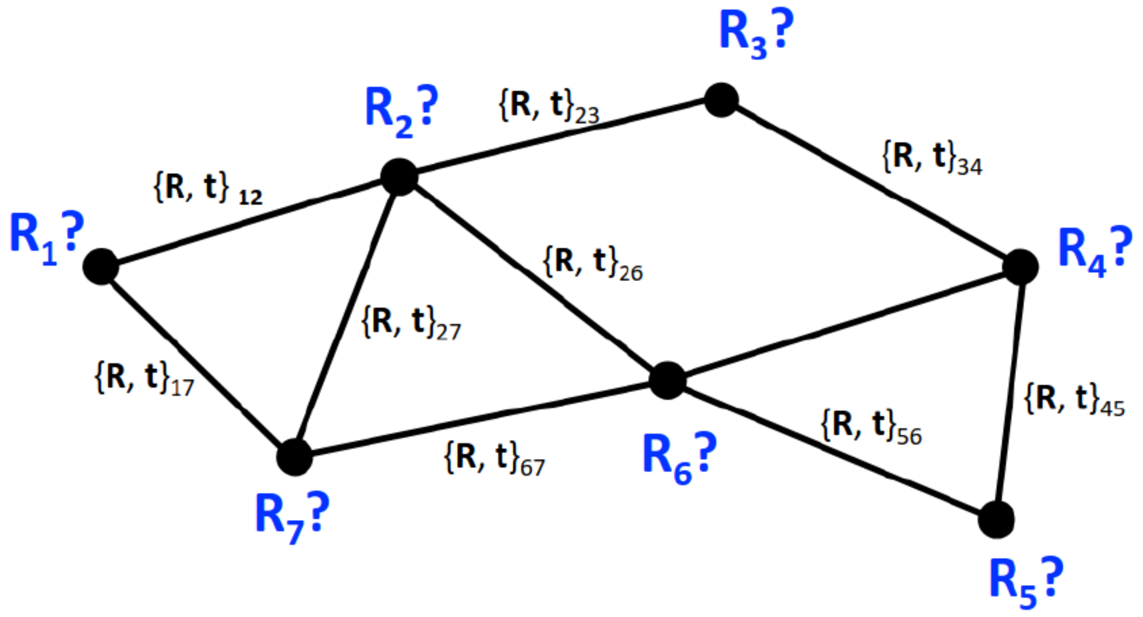


Global SfM

Don't add views incrementally. Solve for global poses in one pass.



Given relative rotation and translation estimates, can we recover the global rotations and translations of cameras?


R_1?
R_2?
R_3?
R_4?
R_5?
R_6?
R_7?
$\{\mathbf{R}, \mathbf{t}\}_{12}$
$\{\mathbf{R}, \mathbf{t}\}_{23}$
$\{\mathbf{R}, \mathbf{t}\}_{34}$
$\{\mathbf{R}, \mathbf{t}\}_{26}$
$\{\mathbf{R}, \mathbf{t}\}_{27}$
$\{\mathbf{R}, \mathbf{t}\}_{17}$
$\{\mathbf{R}, \mathbf{t}\}_{67}$
$\{\mathbf{R}, \mathbf{t}\}_{56}$
$\{\mathbf{R}, \mathbf{t}\}_{45}$

This is known as rotation and translation averaging

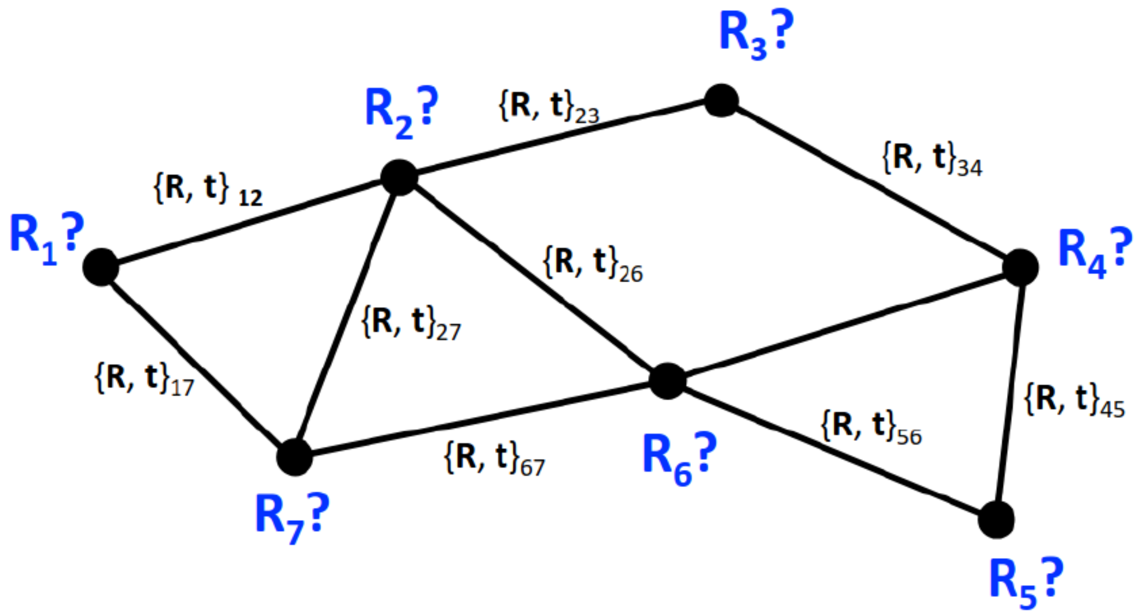
Rotation Averaging

Goal: Estimate global rotations to minimize relative rotation error.

$$\min_{\hat{\mathbf{R}}} \|\mathbf{R}_{ij} - \hat{\mathbf{R}}_j \hat{\mathbf{R}}_i^T\|$$

where

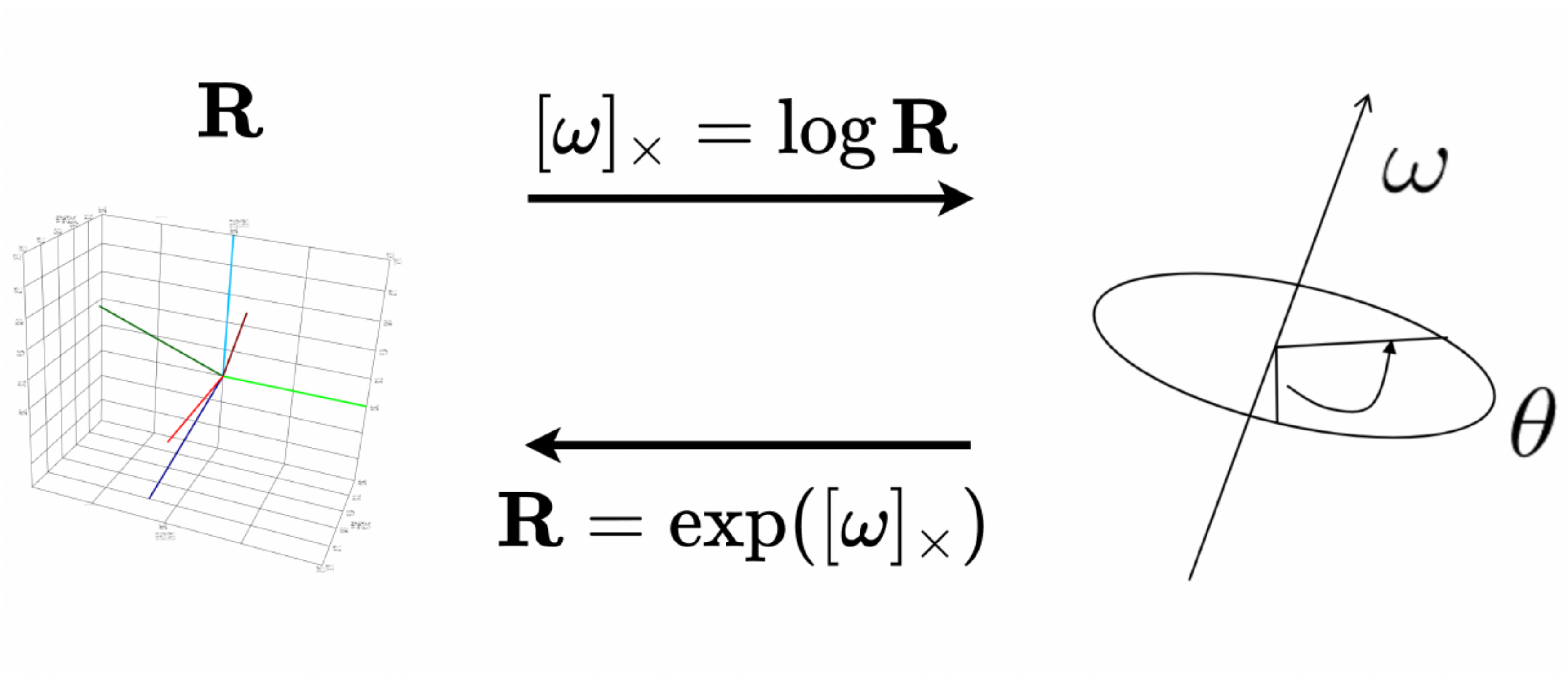
- $\hat{\mathbf{R}} = [\hat{\mathbf{R}}_1, \dots, \hat{\mathbf{R}}_N]$ are the global camera rotations we optimize for
- $\mathbf{R}_{ij}$ is the relative rotation estimate between cameras i and j



Rotation Averaging

Alternative Representation of Rotations:

- Rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$
- Axis angle $\boldsymbol{\omega} = \theta \mathbf{n} \in \mathbb{R}^3$


$\mathbf{R}$
$[\omega]_\times = \log \mathbf{R}$
$\mathbf{R} = \exp([\omega]_\times)$
ω
θ

Rotation Averaging

A single relationship $\mathbf{R}_{ij} = \mathbf{R}_j \mathbf{R}_i^T$ has the first-order approximation

$$\boldsymbol{\omega}_{ij} = \boldsymbol{\omega}_j - \boldsymbol{\omega}_i$$

Rewrite as linear equations

$$\begin{bmatrix} \cdots & -\mathbf{I} & \cdots & \mathbf{I} & \cdots \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_1 & \cdots & \boldsymbol{\omega}_C \end{bmatrix}^T = \boldsymbol{\omega}_{ij}$$

Three linear equations for each edge in the view graph. Stack these on top of each other:

$$\mathbf{A}\boldsymbol{\omega}_{glob} = \boldsymbol{\omega}_{rel}$$

Can be solved in the least squares sense via SVD (pseudo-inverse $\mathbf{A}^+ = \mathbf{V}\boldsymbol{\Sigma}^+\mathbf{U}^T$).

Rotation Averaging

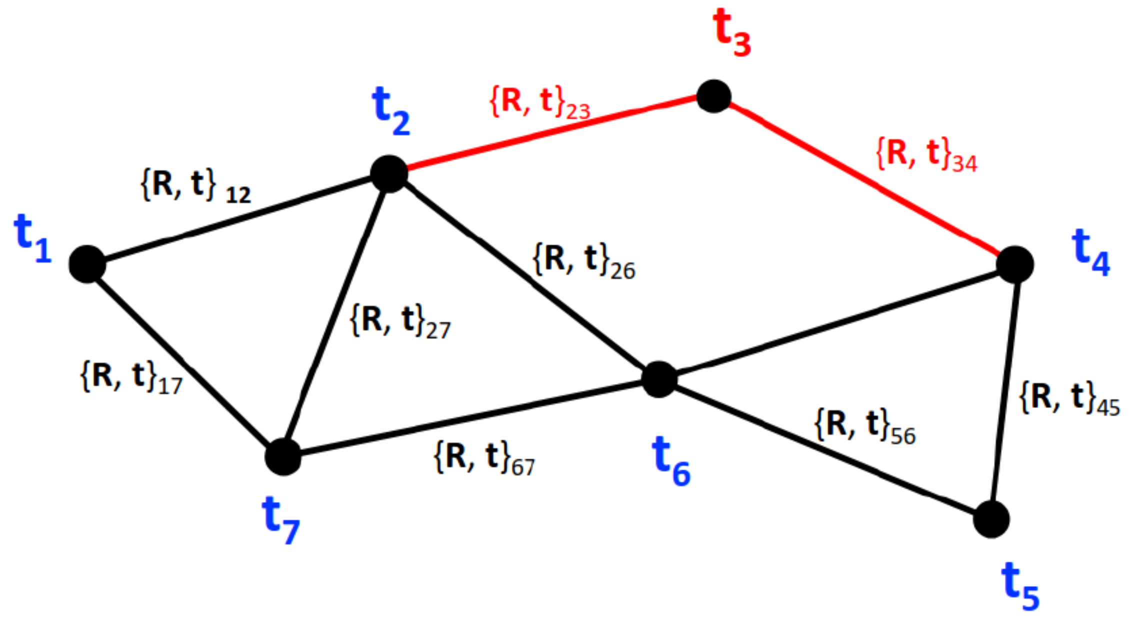
- Problem: $\boldsymbol{\omega}_{ij} = \boldsymbol{\omega}_j - \boldsymbol{\omega}_i$ is only a first-order approximation.
- But we can do this repeatedly (a bit like gradient descent)

```
Algorithm 1 Lie-Algebraic Relative Rotation Averaging
Input: $\{\mathbf{R}_{ij1}, \cdots, \mathbf{R}_{ijk}\}$ ($|\mathcal{E}|$ relative rotations)
Output: $\mathbf{R}_{global} = \{\mathbf{R}_1, \cdots, \mathbf{R}_N\}$ ($|\mathcal{V}|$ absolute rotations)
Initialisation: $\mathbf{R}_{global}$ to an initial guess
    while $||\Delta\boldsymbol{\omega}_{rel}|| > \epsilon$ do
        1. $\Delta\mathbf{R}_{ij} = \mathbf{R}_j{}^{-1}\mathbf{R}_{ij}\mathbf{R}_i$
        2. $\Delta\boldsymbol{\omega}_{ij} = \log(\Delta\mathbf{R}_{ij})$
        3. Solve $\mathbf{A}\Delta\boldsymbol{\omega}_{global} = \Delta\boldsymbol{\omega}_{rel}$
        4. $\forall k \in [1, N], \mathbf{R}_k = \mathbf{R}_k exp(\Delta\boldsymbol{\omega}_k)$
    end while
```

Chatterjee et al. ”Efficient and Robust Large-Scale Rotation Averaging”, ICCV’13

Rotation Averaging

Given relative rotation and translation estimates, can we recover the global rotations and translations of cameras?



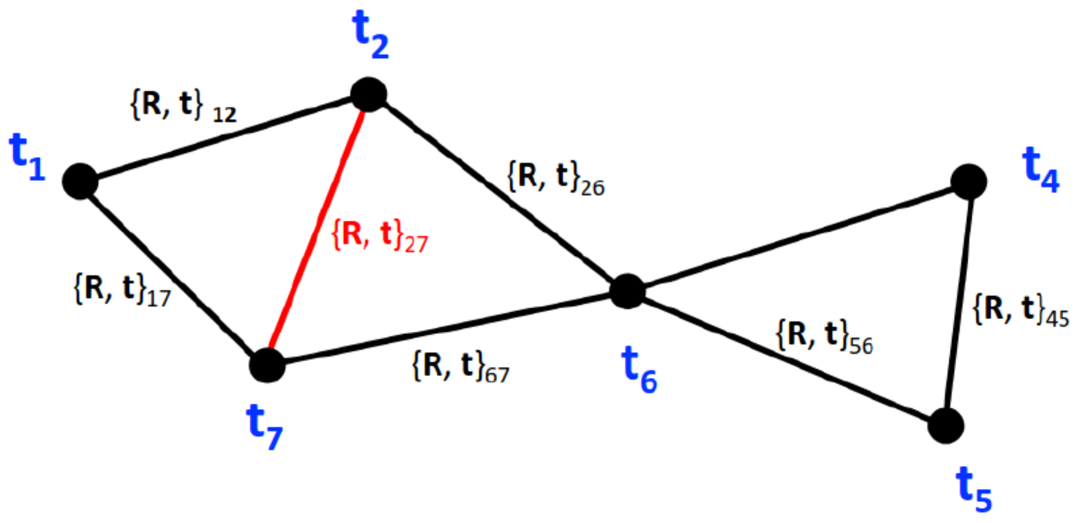
After estimating global rotations, filter out outliers:

$$\|\mathbf{R}_{ij} - \mathbf{R}_j\mathbf{R}_i^T\| > \epsilon$$

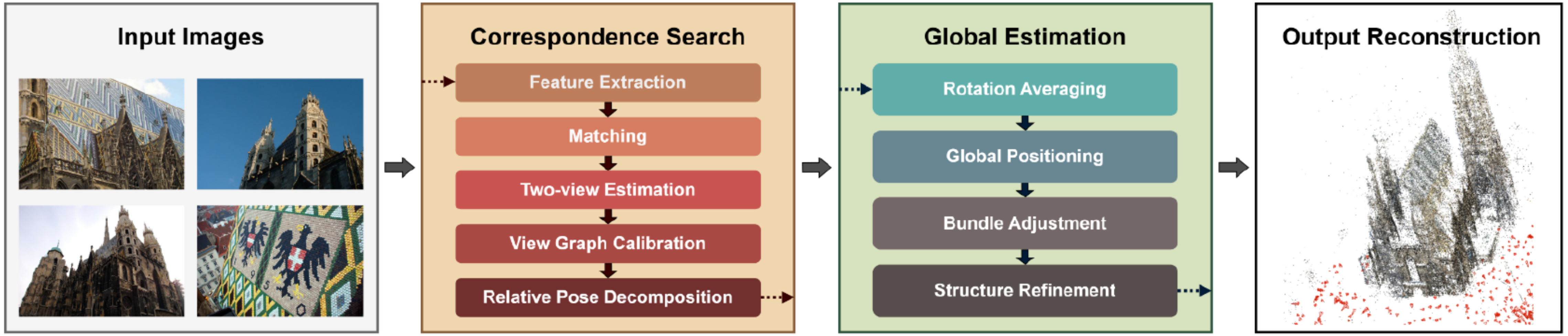
Translation Averaging

Similarly, estimate and filter global translations:

$$\min_{\hat{\mathbf{t}}} \left\| \mathbf{t}_{ij} - \frac{\hat{\mathbf{t}}_i - \hat{\mathbf{t}}_j}{\|\hat{\mathbf{t}}_i - \hat{\mathbf{t}}_j\|} \right\|$$


$\mathbf{t}_2$
$\{\mathbf{R}, \mathbf{t}\}_{12}$
$\mathbf{t}_1$
$\{\mathbf{R}, \mathbf{t}\}_{26}$
$\mathbf{t}_4$
$\{\mathbf{R}, \mathbf{t}\}_{27}$
$\{\mathbf{R}, \mathbf{t}\}_{17}$
$\{\mathbf{R}, \mathbf{t}\}_{45}$
$\{\mathbf{R}, \mathbf{t}\}_{56}$
$\{\mathbf{R}, \mathbf{t}\}_{67}$
$\mathbf{t}_6$
$\mathbf{t}_7$
$\mathbf{t}_5$

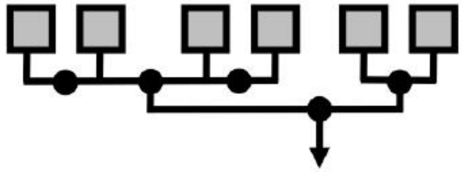
GLOMAP: A Global SfM Pipeline

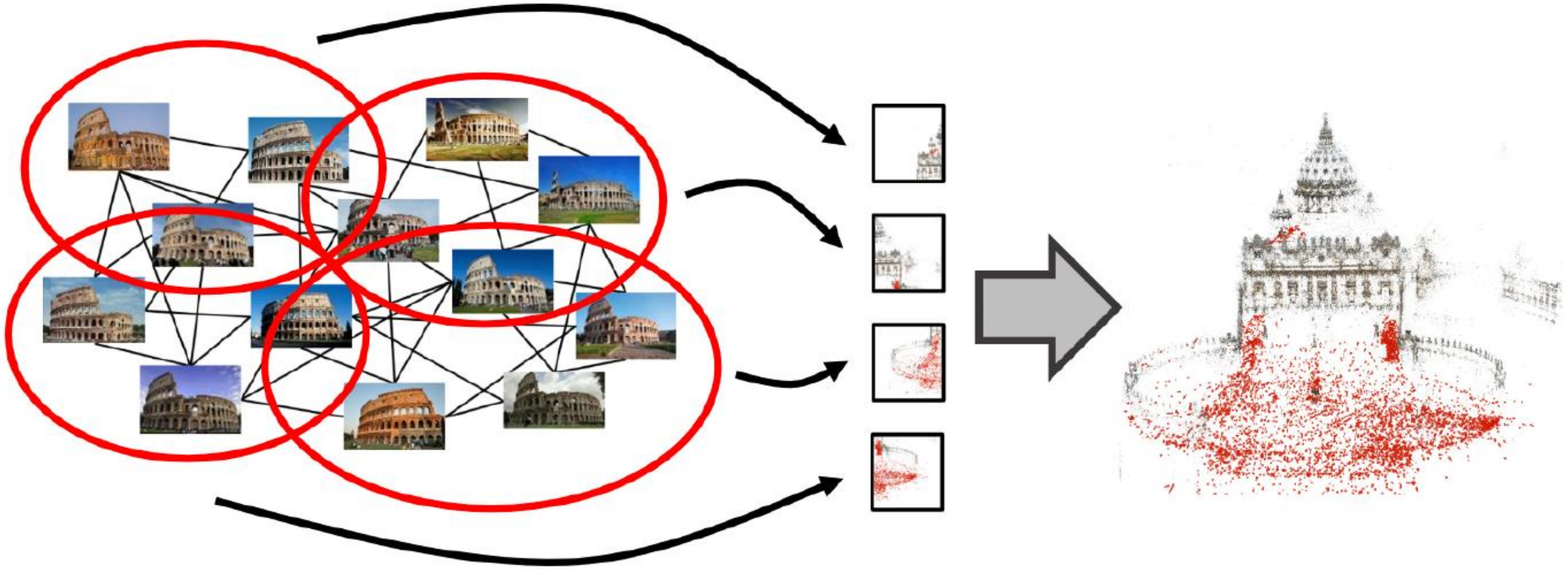


Fan et al. ”GLOMAP: Global Structure-from-Motion Revisited”, ECCV’24

Hierarchical SfM

1. Hierarchically cluster the scene graph
2. Reconstruct each cluster independently
3. Merge clusters using similarity transformations





Comparison

Method	Efficiency	Robustness	Accuracy
Incremental	-	++	+
Global	+	+	+
Hierarchical	++	-	-

Incremental SfM

- OpenSfM
- COLMAP
- etc.

Global SfM

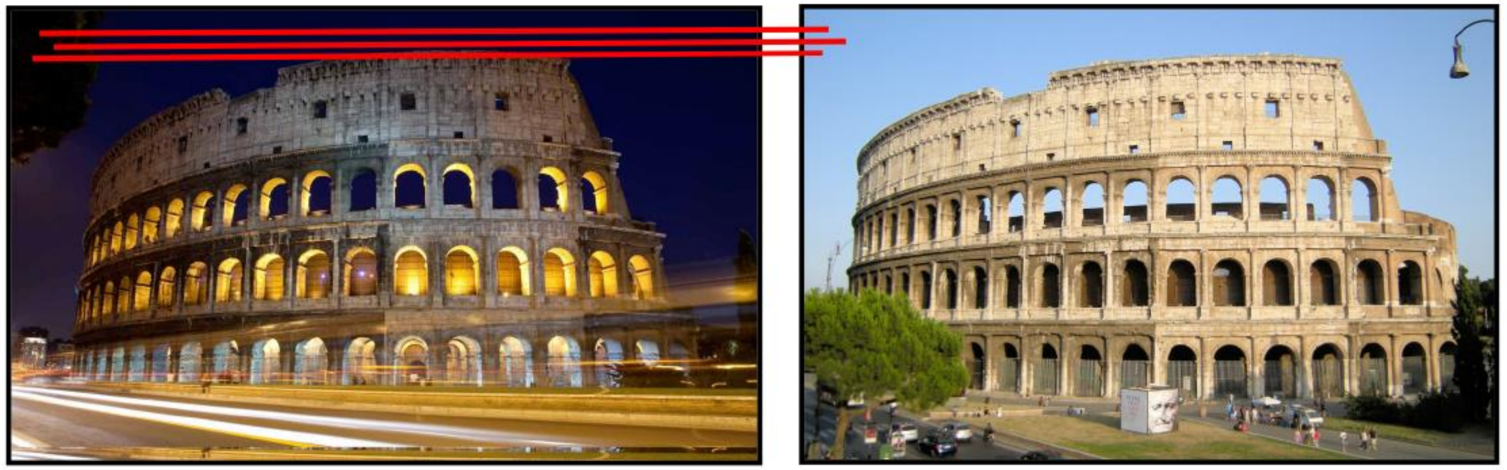
- OpenMVG
- Theia
- GLOMAP

Teaser: Learning-based methods are taking over!

Challenges

WTFs

- Watermarks, Timestamps, Frames (WTFs)
- Detect translation at image border



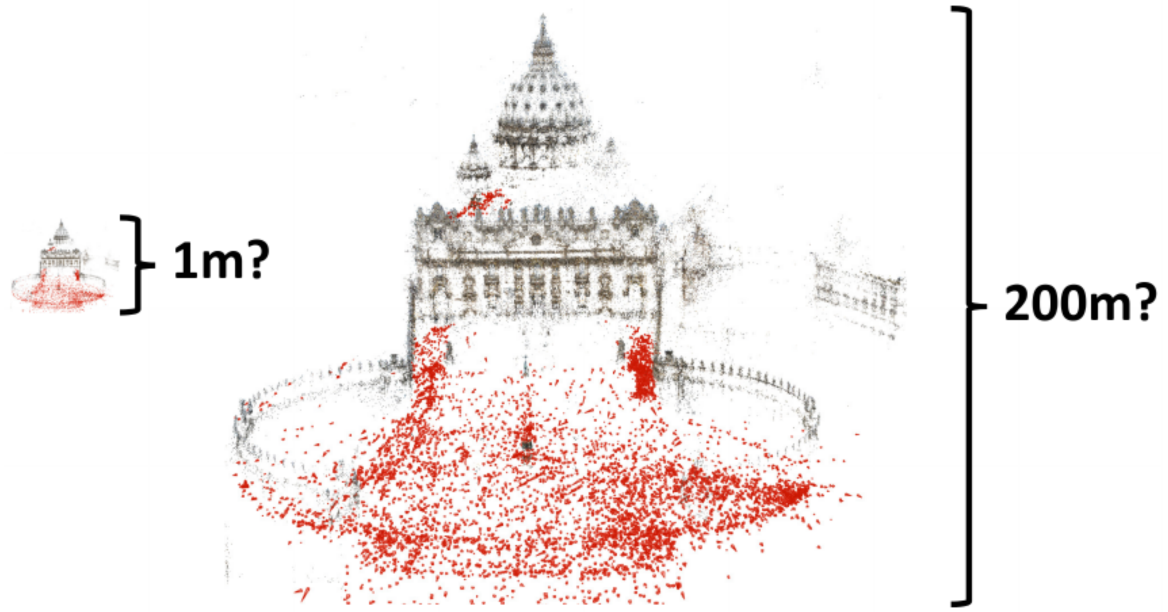
Ambiguities

- Is the output of SfM uniquely determined, given a set ob correspondences?

Ambiguities

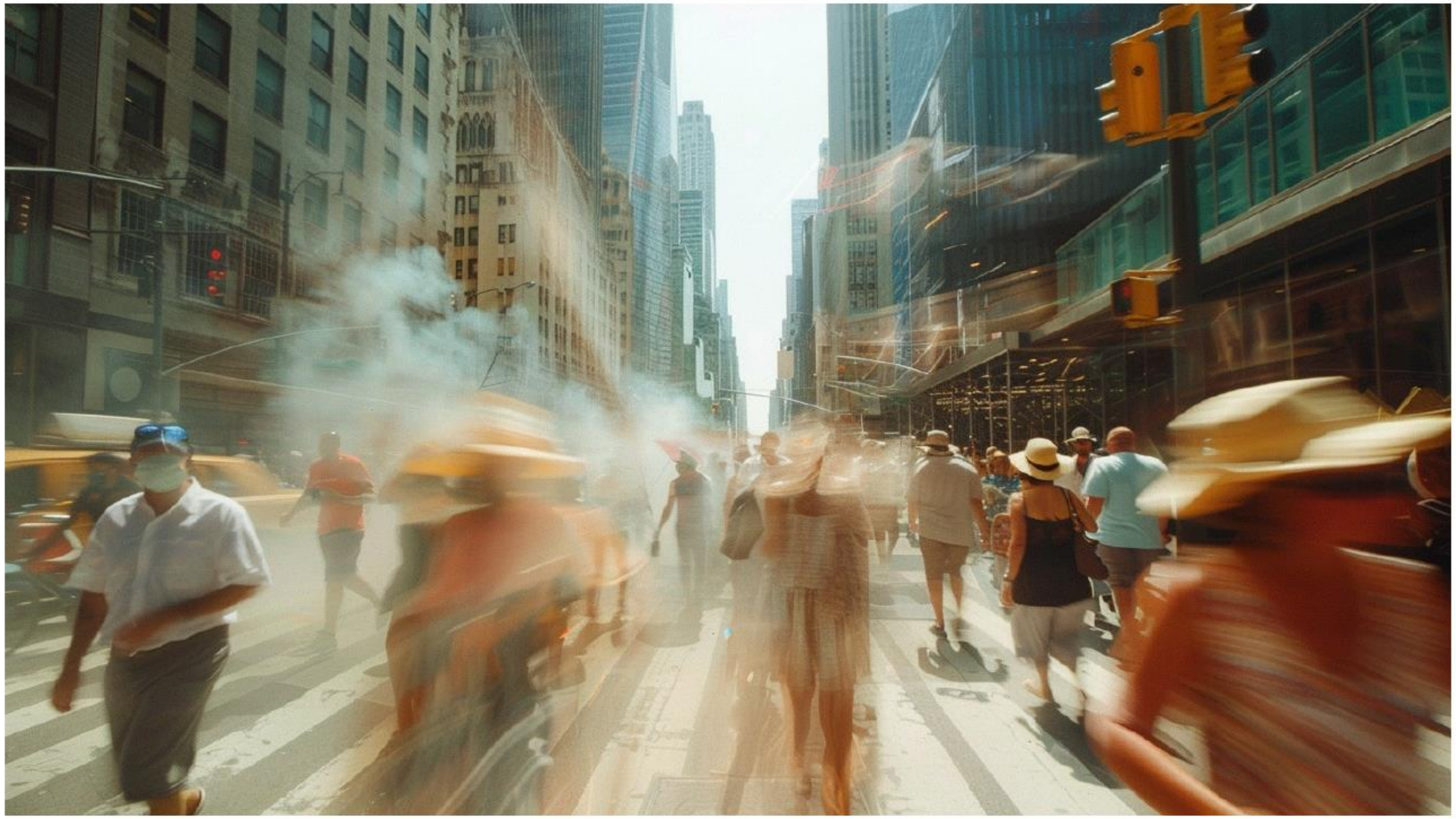
- Is the output of SfM uniquely determined, given a set ob correspondences?
- No! SfM is inherently scale ambiguous
- If we scale the entire scene by factor α, and scale camera matrices by factor α^{-1}, the projections of the scene points remain the same:

$$\mathbf{u} \simeq (\alpha^{-1}\mathbf{P})(\alpha\mathbf{X})$$



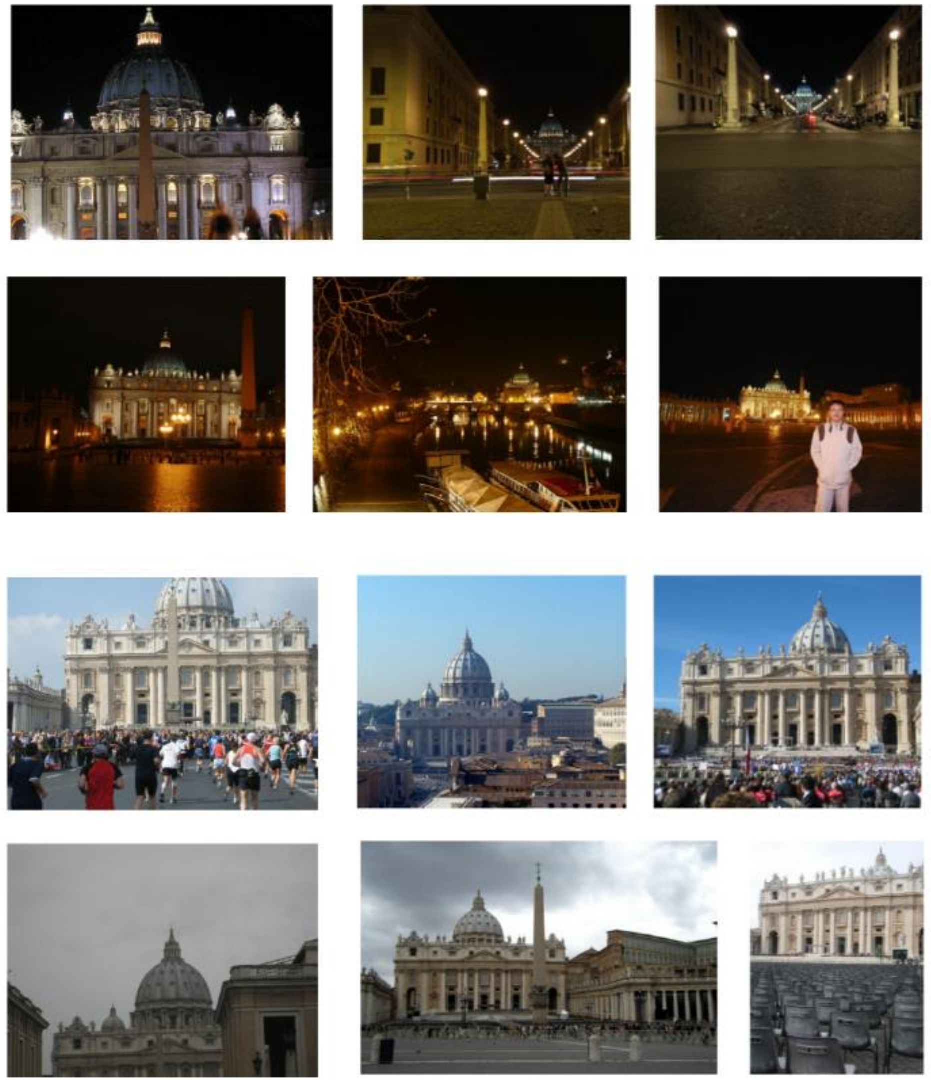
Repetitive Structures

Dynamic Scenes



https://stockcake.com/i/city-life-blur_1010336_976993

Illumination / Weather change



Slide Credits

Slides are adapted from the following sources

- Schönberger, Tutorial: Large-scale 3D Modeling, CVPR ’17
- Silvio Savarese, CS231A
- Marc Pollefeys, 3D Vision
- Soumyadip Sengupta, Computer Vision in 3D World
- Andreas Geiger, Computer Vision