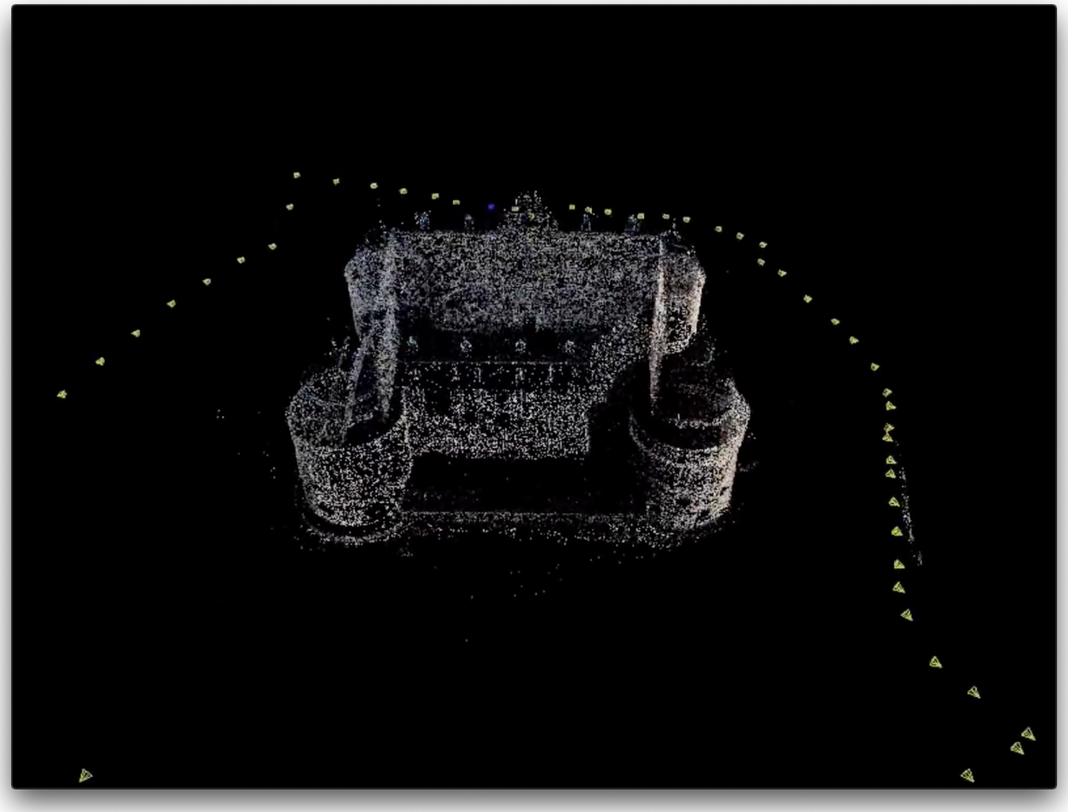
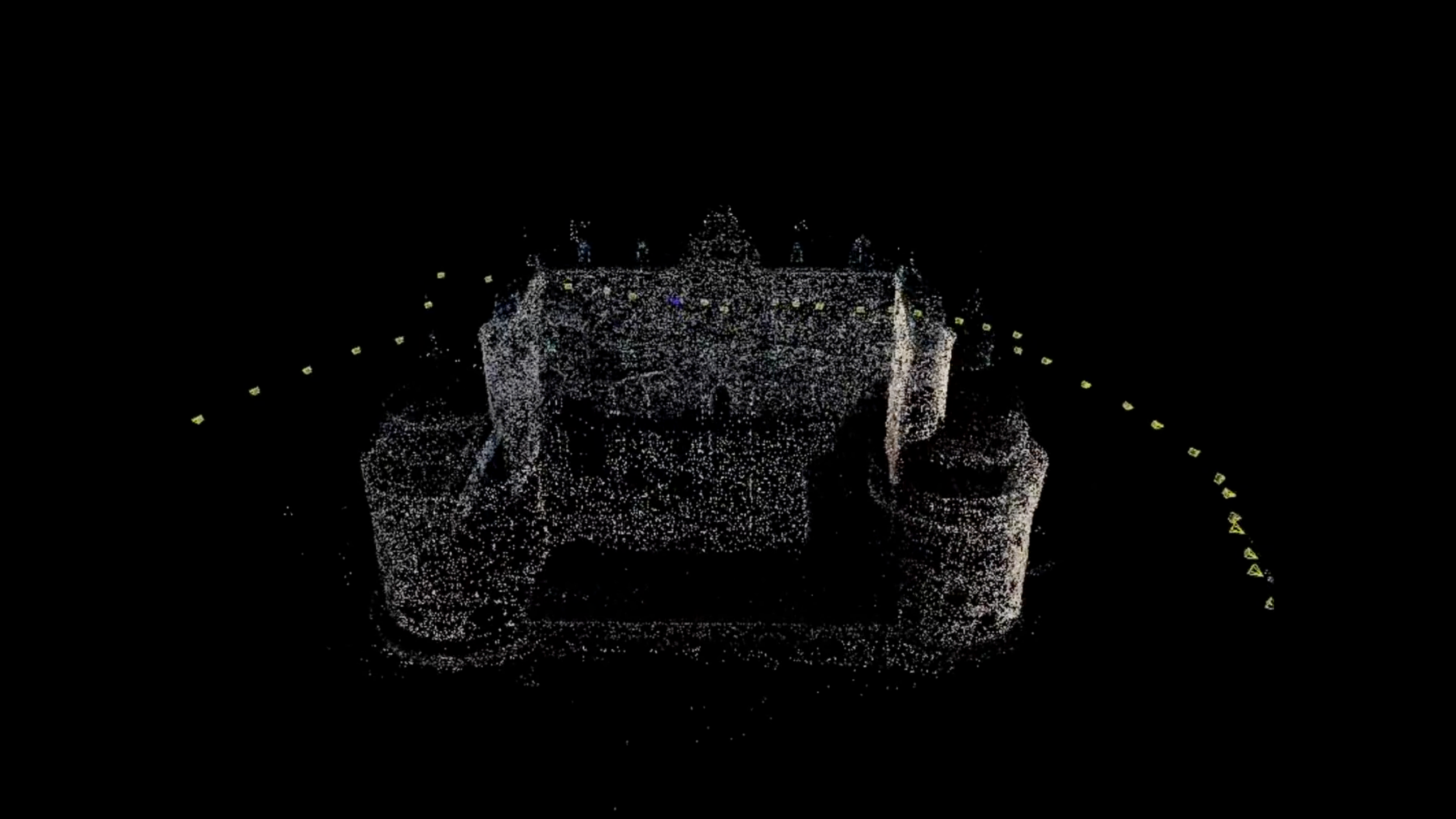
Structure From Motion

CS 650: Computer Vision

Scene Reconstruction from Multiple Cameras

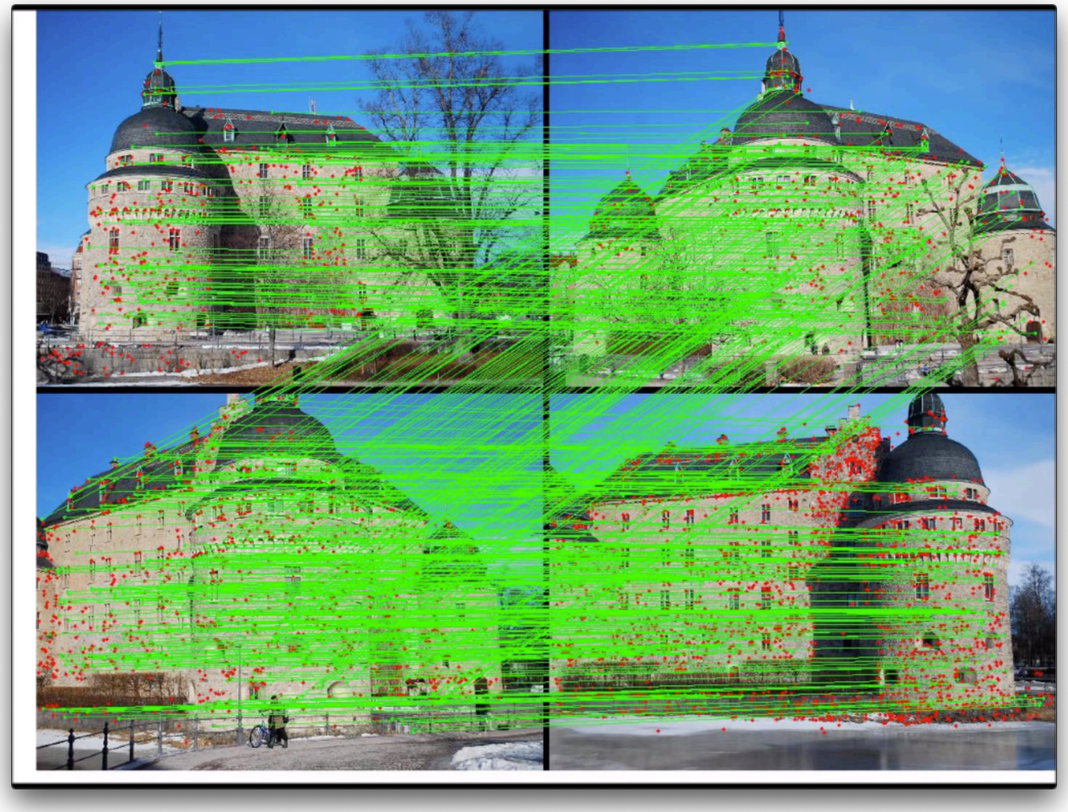
- Given multiple images:
  - Same scene (at least partial overlap in views)
  - Different unknown viewpoints
  - Possibly different cameras (calibrated or not)
- Goal:
  - Determine relative poses **R**, **t** of the cameras
  - Recover the geometry (position of 3D points)
  - Calibrate the cameras in the process (if needed)
- Can do *dense reconstruction* after that





Scene Reconstruction from Multiple Cameras

- Basic Idea:
  - Identify some number of corresponding points in the images, manually or automatically (we've seen this before)
  - Use this to determine the relative camera poses (position and orientation)
- Then if desired:
  - Use the now-known geometry to constrain a denser point-to-point correspondence search
  - Solve for a dense depth map using stereo



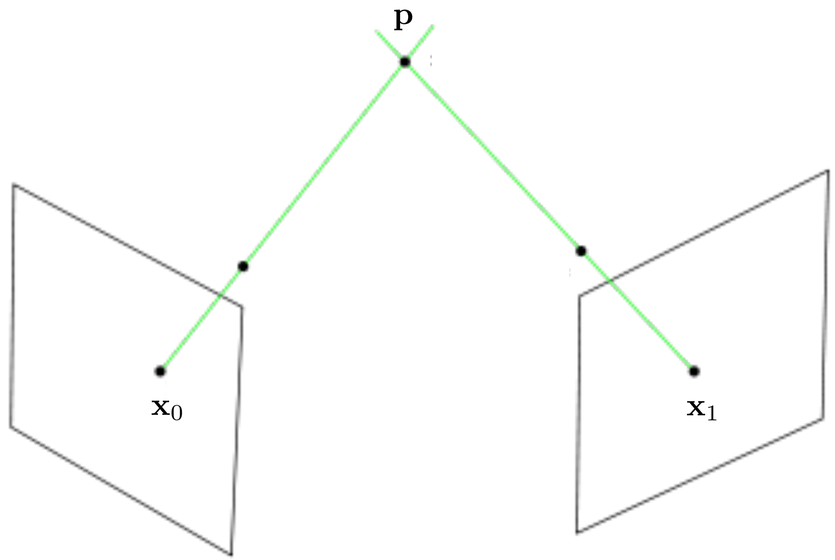
Projection (revisited)

$$z_c \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\underbrace{\begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

$$\mathbf{x} \sim \mathbf{P}\ \mathbf{p}$$

Triangulation

- Simpler problem:
  - Two cameras with projection matrices **P_0** and **P_1**
  - Both see a common unknown 3D point **p** at image points **x_0** and **x_1**

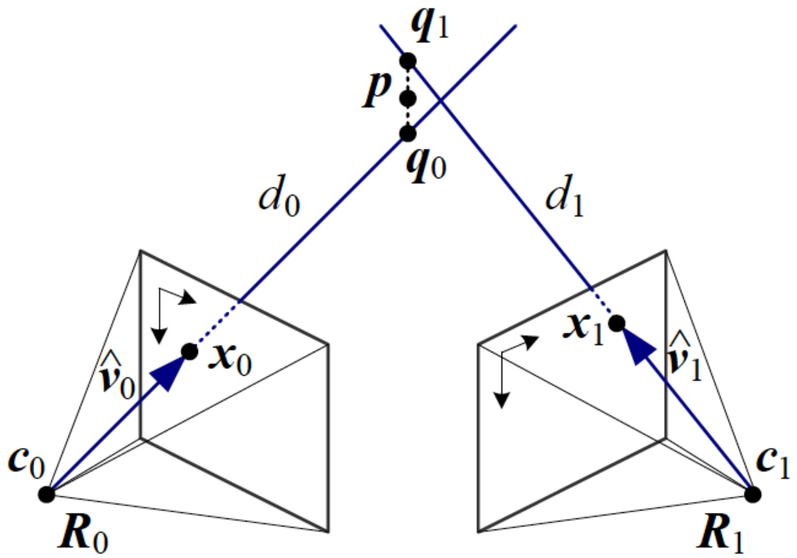


$$\mathbf{x}_0 \sim \mathbf{P}_0 \ \mathbf{p}$$

$$\mathbf{x}_1 \sim \mathbf{P}_1 \ \mathbf{p}$$

Triangulation

- Problem:
  the two rays may not intersect, so no solution
- Temping approach:
  Find the 3D point that's the least-squares solution to this system of equations
- Better approach:
  Find point minimizing **reprojection error**



$$\mathbf{x}_0 \sim \mathbf{P}_0 \ \mathbf{p}$$

$$\mathbf{x}_1 \sim \mathbf{P}_1 \ \mathbf{p}$$

Triangulation as Optimization

- Corresponding points:

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

- Projections of some possible solution **p**:

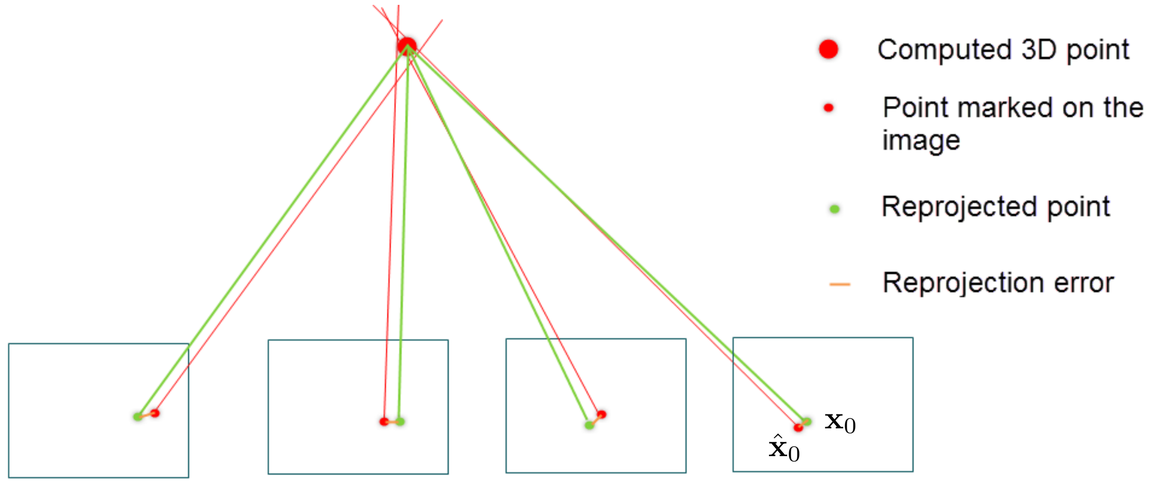
$$\mathbf{x}_0 \sim \mathbf{P}_0\ \mathbf{p} \qquad \mathbf{x}_1 \sim \mathbf{P}_1\ \mathbf{p}$$

- Reprojection error:

$$\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|^2 + \|\mathbf{x}_1 - \hat{\mathbf{x}}_1\|^2$$

- Can generalize to as many cameras as see the point **p**:

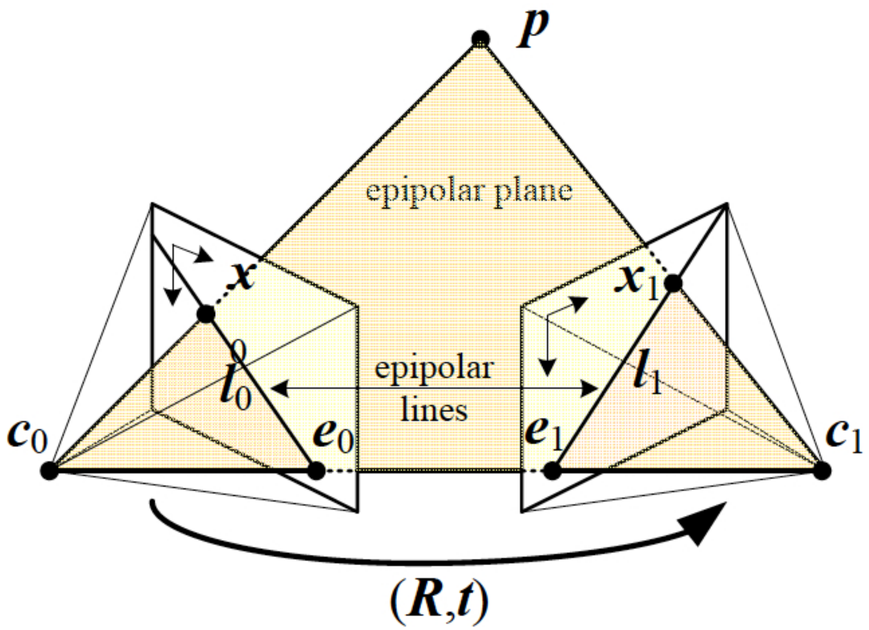
$$\sum_i \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$$



Two cameras,
Calibrated case

Epipolar Geometry

- Consider just one point $\mathbf{x_0}$ in one image
- The set of possible 3-D locations **p** for this point is constrained to be a ray through the projected position
- When viewed from another camera, this ray forms a single line, *and the corresponding point* $\mathbf{x_1}$ *in the other image must be on it*

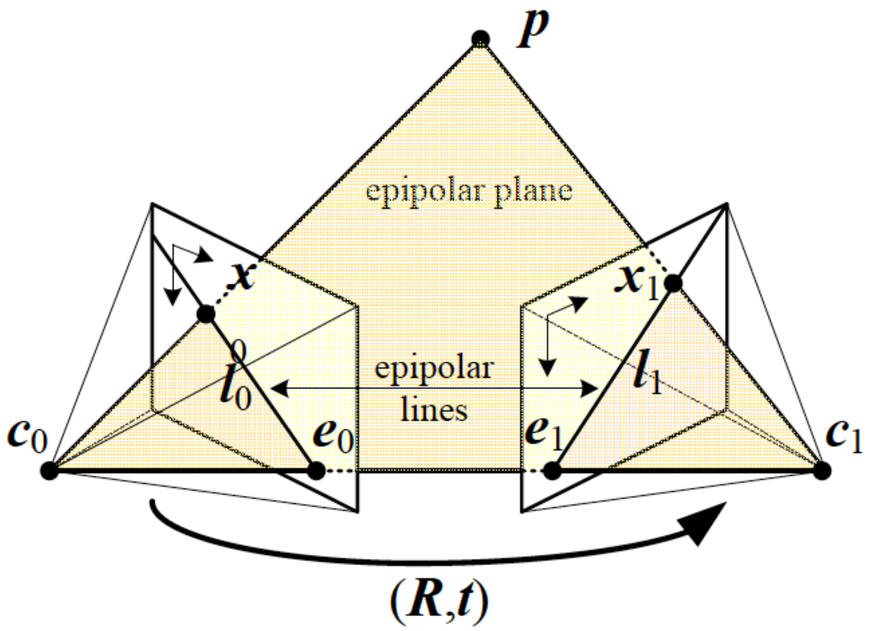


Epipolar Geometry

- **Epipolar plane:**
  The plane formed by a point and two camera centers of projection

- **Epipole:**
  One camera's center of projection as seen from the other camera (may be out of view)

- **Epipolar lines:**
  Intersection of the epipolar plane and the cameras' image planes.



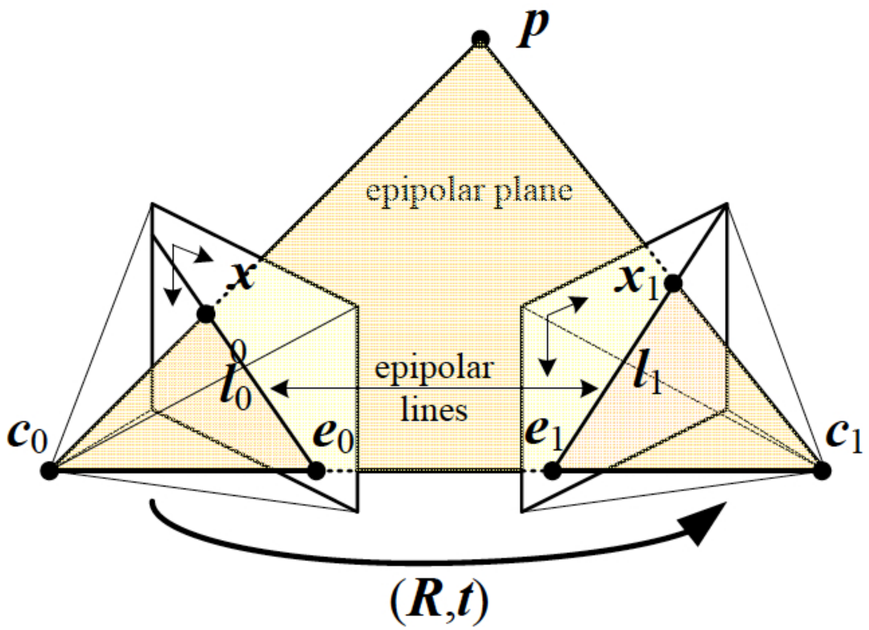
Epipolar Geometry

- Since we don't an absolute reference frame, define one camera's pose in terms of other:

$$\begin{aligned} \mathbf{c}_0 &= \mathbf{0} \\ \mathbf{R}_0 &= \mathbf{I} \\ \mathbf{c}_1 &= \mathbf{t} \\ \mathbf{R}_1 &= \mathbf{R} \end{aligned}$$

- Assume known camera matrices $\mathbf{K}_i$ and use calibrated coordinates

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \mathbf{K}_0^{-1}\mathbf{x}_0 \\ \hat{\mathbf{x}}_1 &= \mathbf{K}_1^{-1}\mathbf{x}_1 \end{aligned}$$

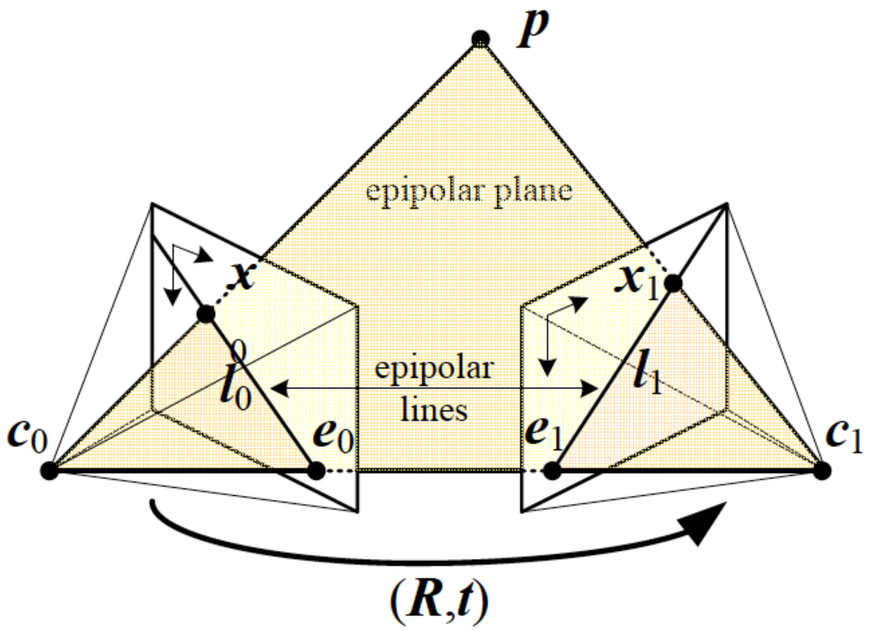


The Epipolar Constraint

- We can encode the epipolar constraint using the *essential matrix* **E** as

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

- Really important:
  - **E** has rank 2, not 3
    (a point in one image corresponds to a line of possibilities in the other)
  - **E** is unique only up to a constant scaling
    (8 degrees of freedom)



Notational Detour

Vector cross products

$$\mathbf{v}_1 \times \mathbf{v}_2$$

can also be written as a matrix operation

$$[\mathbf{v}_1]_\times \times \mathbf{v}_2$$

where

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_x & v_y \\ v_x & 0 & -v_z \\ -v_y & v_z & 0 \end{bmatrix}$$

The Epipolar Constraint

$\mathbf{t}$ is in the epipolar plane

$\hat{\mathbf{x}}_1$ is in the epipolar plane

$\mathbf{R}\ \hat{\mathbf{x}}_0$ is in the epipolar plane

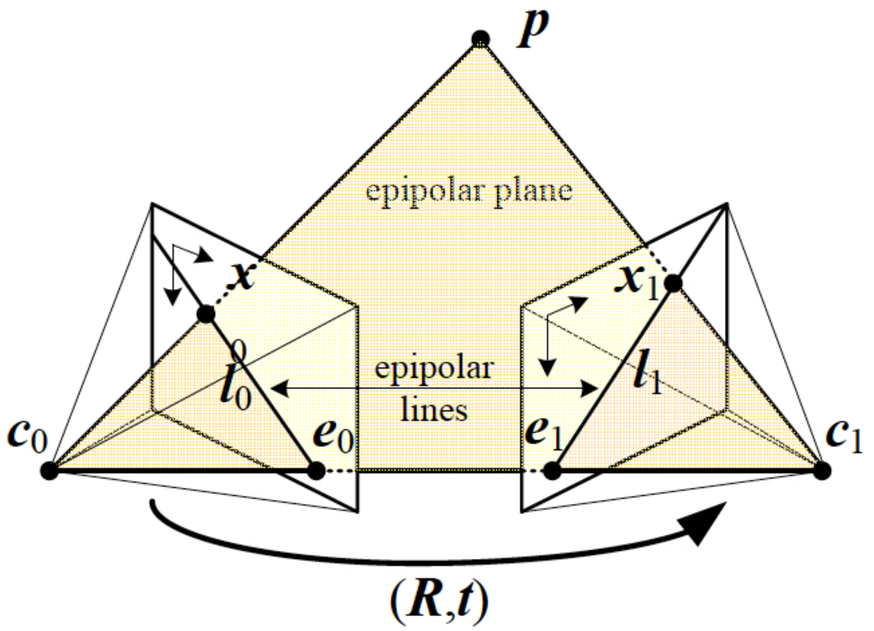
$\hat{\mathbf{x}}_1 \times \mathbf{t}$ is orthogonal to the epipolar plane

$$(\hat{\mathbf{x}}_1 \times \mathbf{t}) \cdot \mathbf{R}\ \hat{\mathbf{x}}_0 = 0$$

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

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

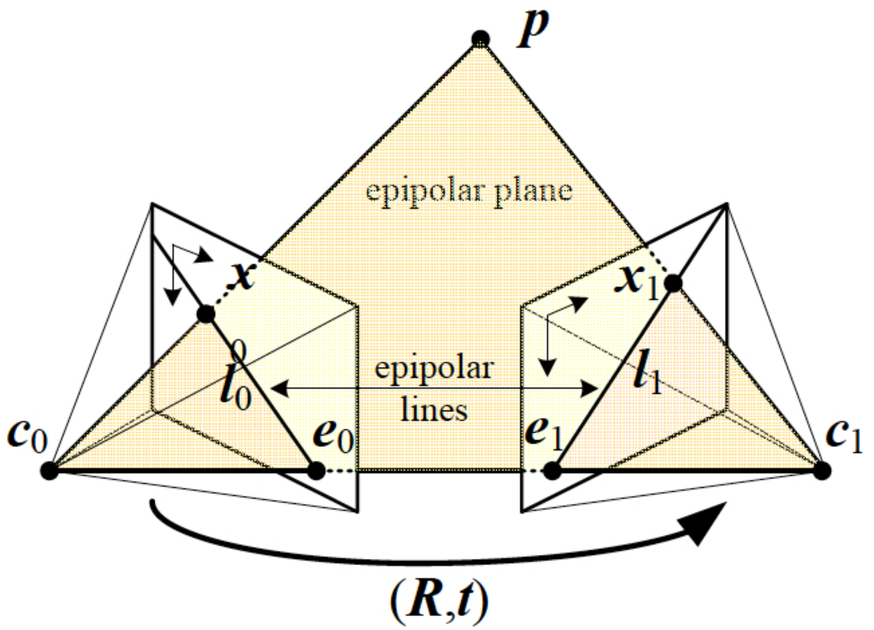
$$\hat{\mathbf{x}}_1^T\ \mathbf{E}\ \hat{\mathbf{x}}_0 = 0$$



The Epipolar Constraint

- So why do we care?
  - For each point in one image, **E** constrains the space of possible matching points in the other image to a single line
  - If we know **E**, we can factor to get **R** and **t**

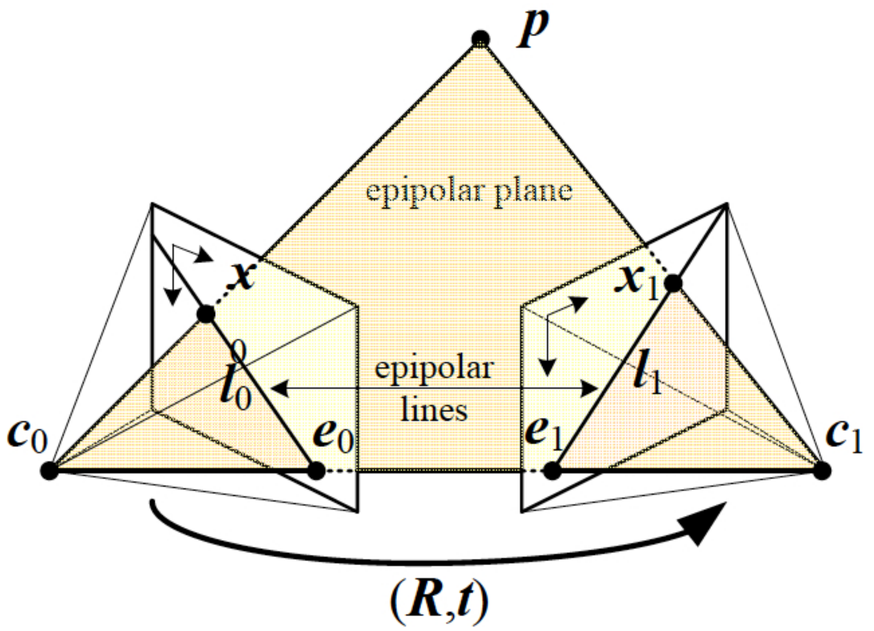
$$\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R}$$



Calculating the Essential Matrix

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

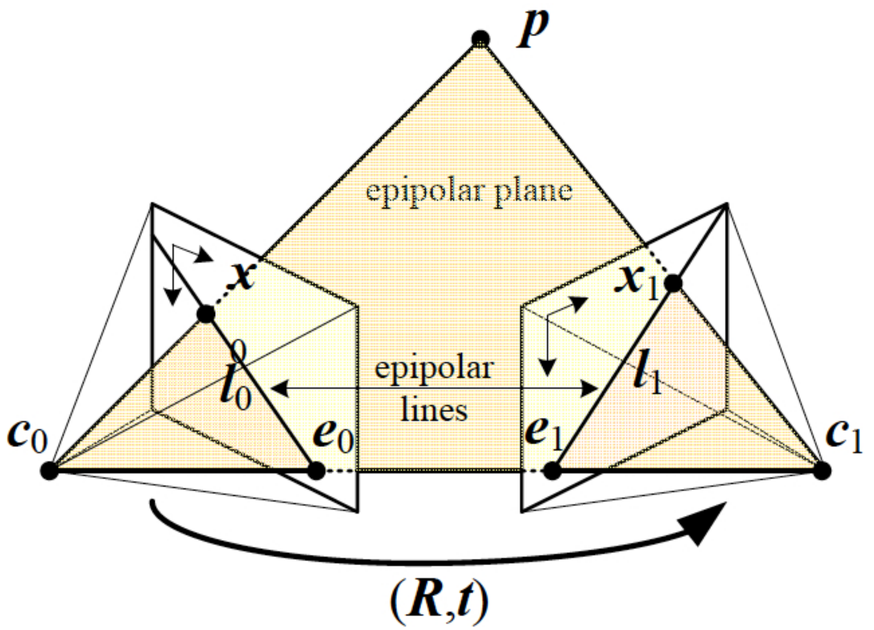
- Solution (look familiar?):
  - Identify at least **eight** corresponding points in the two images
  - Solve for the essential matrix by setting up a system of 8 equations based on at least 8 correspondences
- This is called the *8-point algorithm*
- Nonlinear solutions with 7, 6, or 5 points



Calculating the Essential Matrix

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

- Process (like we did before):
  - Find candidate points and descriptors
  - Greedily match descriptors
  - Reject ambiguous matches
  - Use 8-point algorithm and RANSAC to find the best solution (largest consensus set)



Calculating the Essential Matrix

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

- Caution: the 8-point algorithm requires points in a “general 3D configuration”
- If there is a substantial subset of the putative matches that are coplanar, RANSAC solution can be erroneous (but you can detect this degeneracy)

(from Frahm and Pollefeys, CVPR 2006)

Fixing the Essential Matrix

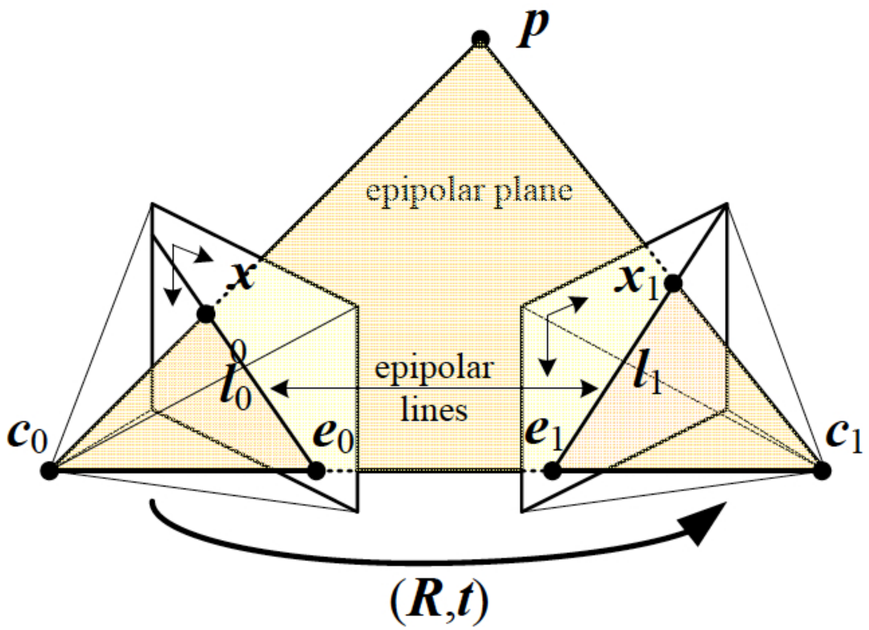
$$\hat{\mathbf{x}}_1^T \; \mathbf{E} \; \hat{\mathbf{x}}_0 = 0$$

- The essential matrix should be rank 2 with *exactly two equal non-zero singular values*, but the 8-point algorithm doesn't guarantee
- Solution:
  - Use singular-value decomposition:

$$\mathbf{E} = \mathbf{U} \; \mathrm{diag}[k_1, k_2, k_3] \; \mathbf{V}^T$$

  - and replace with suitable diagonal matrix

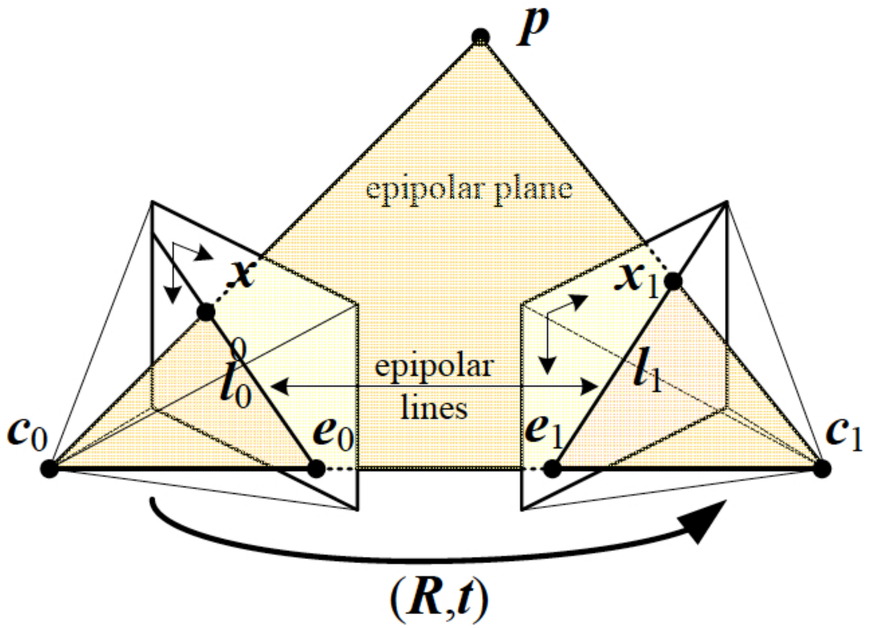
$$\mathbf{E} = \mathbf{U} \; \mathrm{diag}[1, 1, 0] \; \mathbf{V}^T$$



Recovering Pose

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

- The essential matrix can be factored into two matrices $[\mathbf{t}]_x$ and **R** of known forms (skew-symmetric and orthonormal)
- Gives four ambiguous solutions known as the *twisted-pair ambiguity*
- These basically correspond to look out the front and back of both cameras — only one solution is physically plausible
- ★ Now that you know **R** and **t** (we already know **K**) you can compute the projection matrix **P** and triangulate to solve for where the corresponding points are in 3D *in real Euclidean geometry*

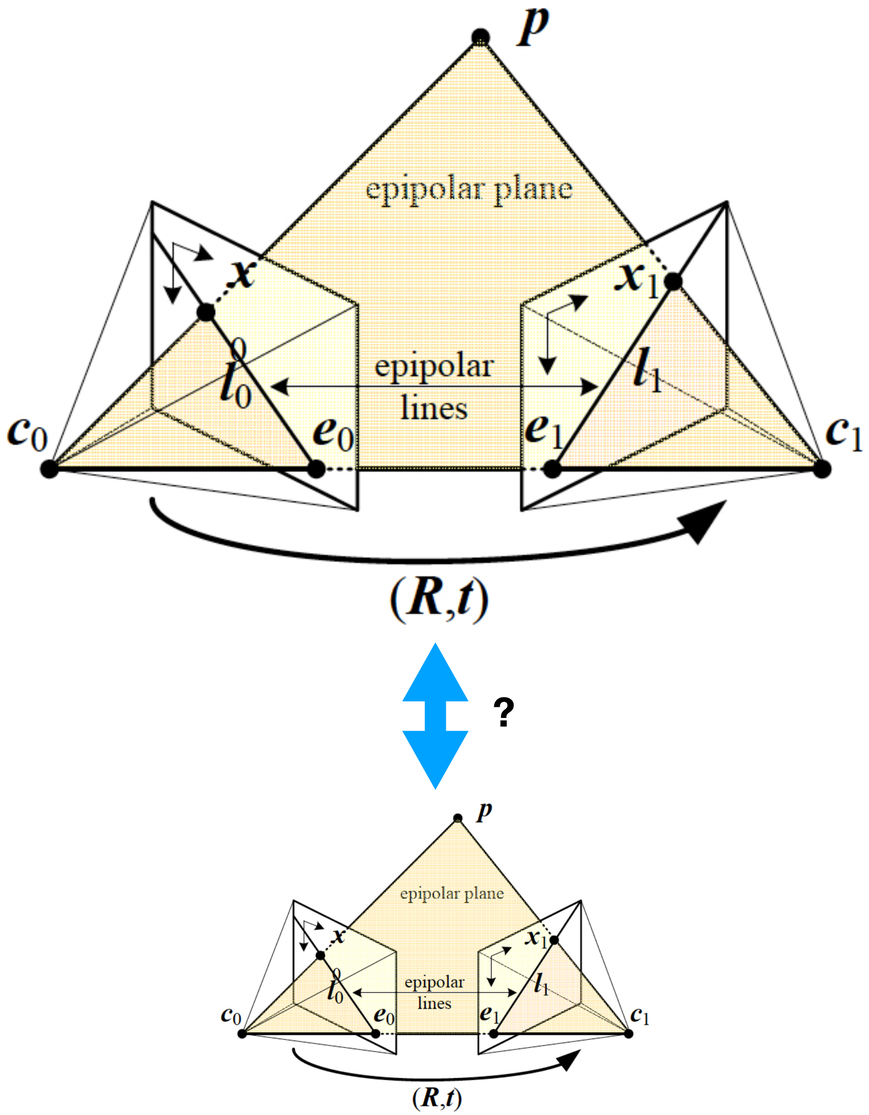


Unsolvable Scale Ambiguity

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

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

- Even with calibrated cameras, we still can only recover 3D geometry up to an unknown constant scale.
- Since **E** is unique up to a scale, you only get the translation baseline **t** up to a scale
- Even more fundamental: moving a little bit in a little world looks *identical* to moving a larger amount in a larger world



Two cameras,
Uncalibrated case

Uncalibrated Case

- With calibrated points:

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

- With uncalibrated points:

$$(\hat{\mathbf{x}}_1^T \mathbf{K}_1^{-T}) \, \mathbf{E} \, (\mathbf{K}_0^{-1} \hat{\mathbf{x}}_0) = 0$$

- and regrouping:

$$\hat{\mathbf{x}}_1^T (\mathbf{K}_1^{-T} \mathbf{E} \, \mathbf{K}_0^{-1}) \hat{\mathbf{x}}_0 = 0$$

The fundamental matrix **F**

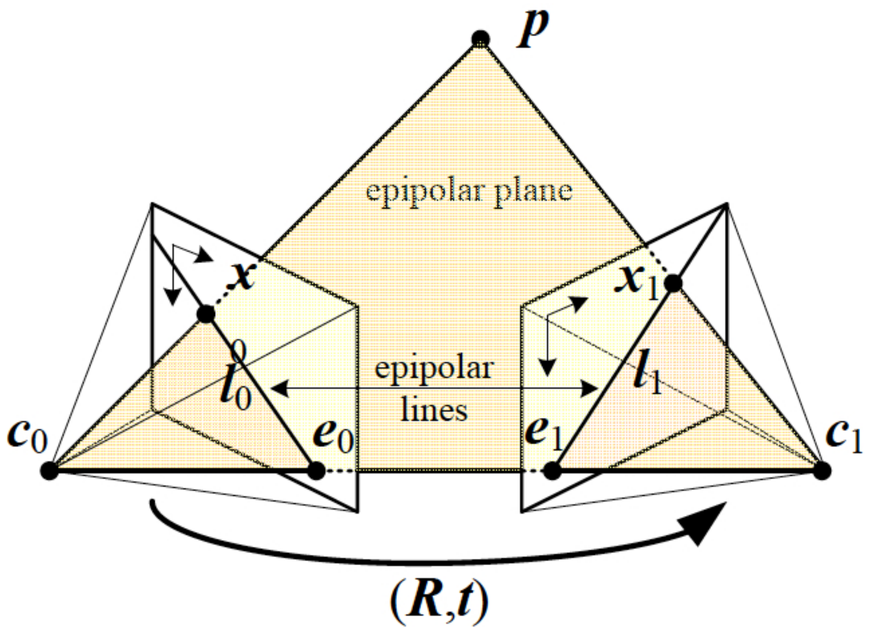
The Fundamental Matrix

- The fundamental matrix **F** encodes epipolar geometry just like the essential matrix **E**, *but for uncalibrated points*:

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

- Solve for **F** the same way you solve for **E** (just use raw pixel coordinates)

- Common approach even for calibrated cameras:
  - Solve for **F** using uncalibrated coordinates
  - Solve for **E** using F and camera matrices **K**
  - Factor **E** into **R** and **t**



Fixing the Fundamental Matrix

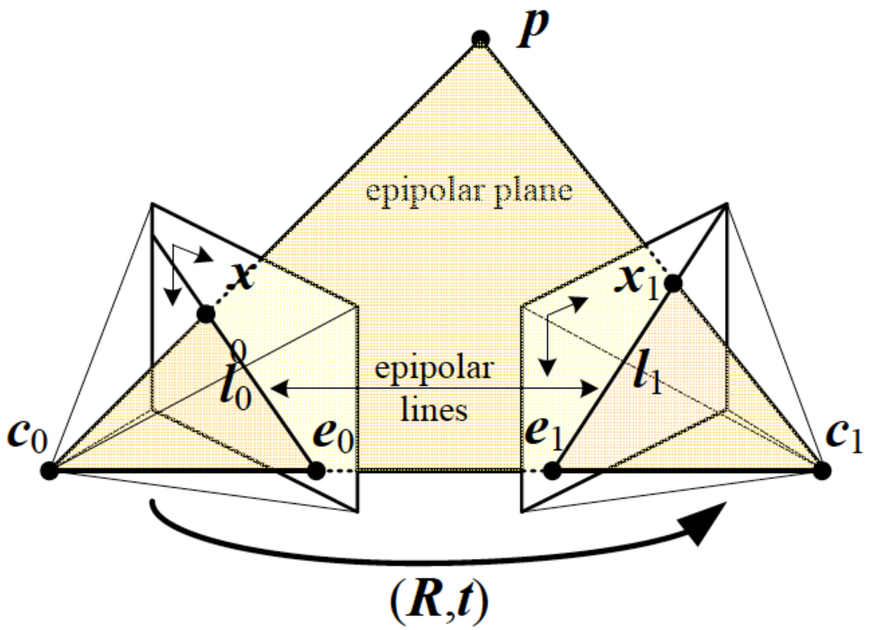
$$\hat{\mathbf{x}}_1^T \ \mathbf{F} \ \hat{\mathbf{x}}_0 = 0$$

- The essential matrix should be rank 2 with *exactly two possibly non-equal non-zero singular values*, but the 8-point algorithm doesn't guarantee
- Solution (like before):
  - Use singular-value decomposition:

$$\mathbf{E} = \mathbf{U} \ \mathrm{diag}[k_1, k_2, k_3] \ \mathbf{V}^T$$

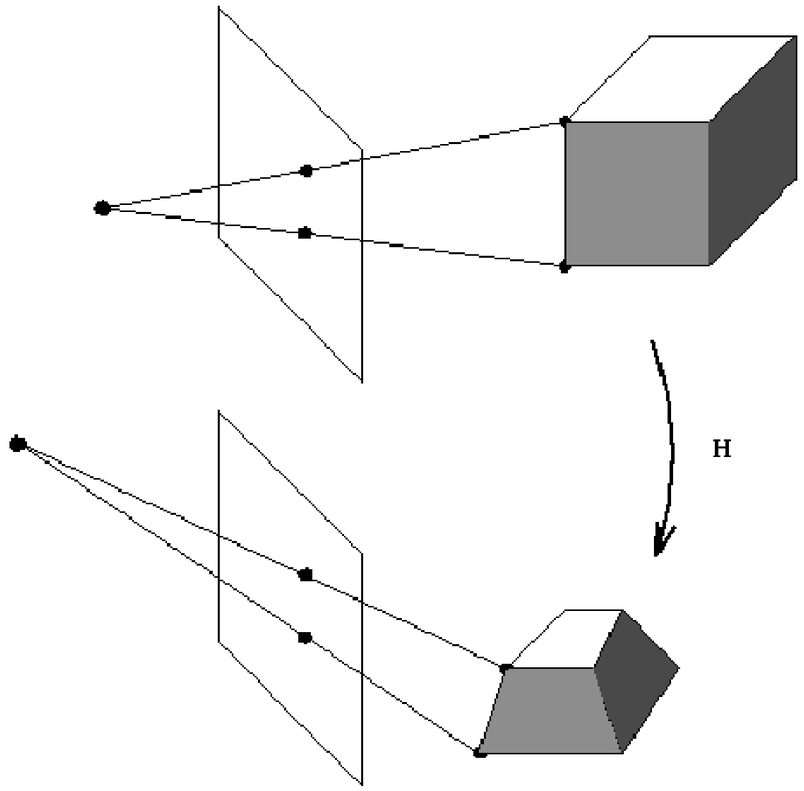
  - and replace with suitable diagonal matrix

$$\mathbf{E} = \mathbf{U} \ \mathrm{diag}[k_1, k_2, 0] \ \mathbf{V}^T$$



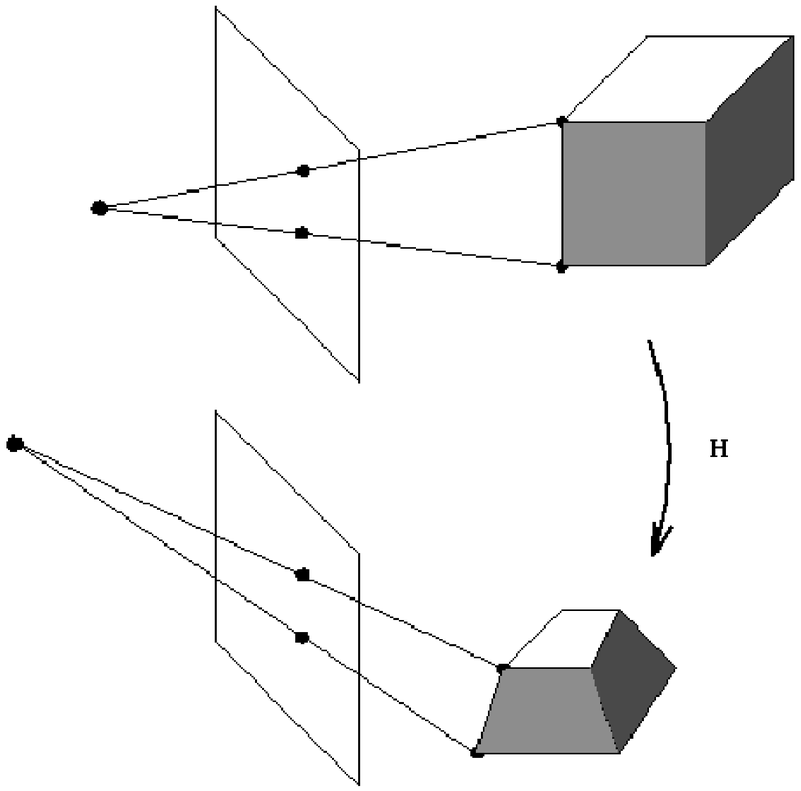
What Can We Do With The Fundamental Matrix?

- **Projective ambiguity:**
  - Can't separate ambiguity between real world and normal camera vs. skewed world and a skewed camera
  - You can *at best* reconstruct the 3-D geometry to within a 4 x 4 homography (may be OK for some things)



Self Calibration

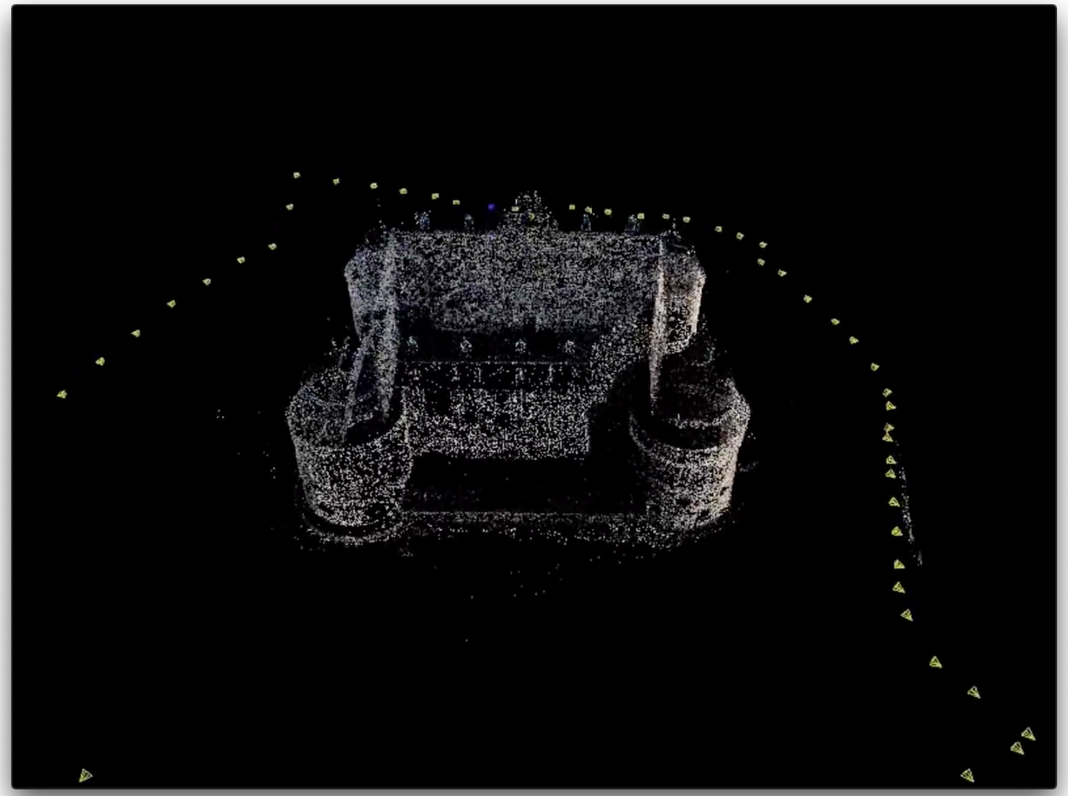
- Can we calibrate the cameras from multiple images?
- Basically, this is resolving the projective ambiguity
- We can if we have some external information:
  - Known parallel lines (and hence vanishing points)
  - Known perpendicular lines
  - ...



Multiple Cameras

Multiple Images

- If you have multiple pictures, you can extend the process by aligning each image to another in a pairwise fashion, but…
- Same problems of drift as stitching panoramas together, so...
- Do a global bundle adjustment (all cameras, all images, all 3D points, etc.)

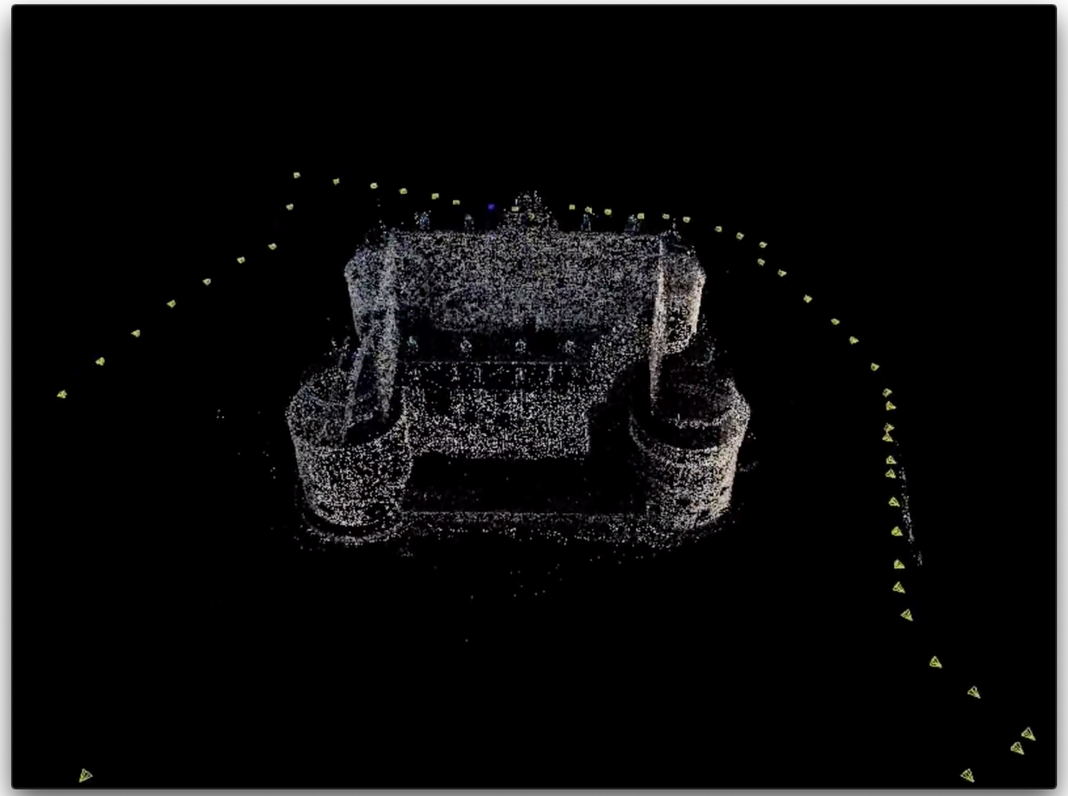


Bundle Adjustment

- Optimize over the entire set of
  - Unknown camera projection matrices $\mathbf{P}_i$
  - Unknown 3D world points $\mathbf{p}_j$
  - Known projections $\mathbf{x}_{ij}$
    (point $\mathbf{p}_j$ as seen with projection $\mathbf{P}_i$)

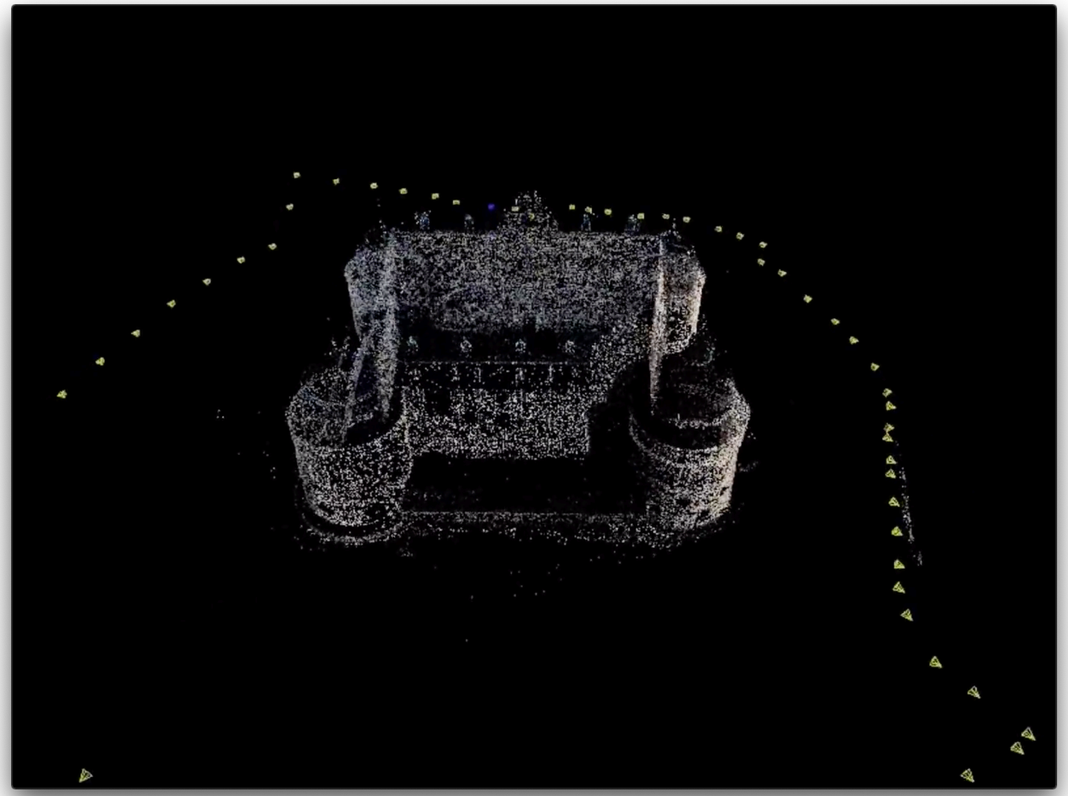
$$\sum_i \sum_j \|\mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij}\|^2$$

$$\mathbf{x}_{ij} \sim \mathbf{P}_i \ \mathbf{p}_j$$



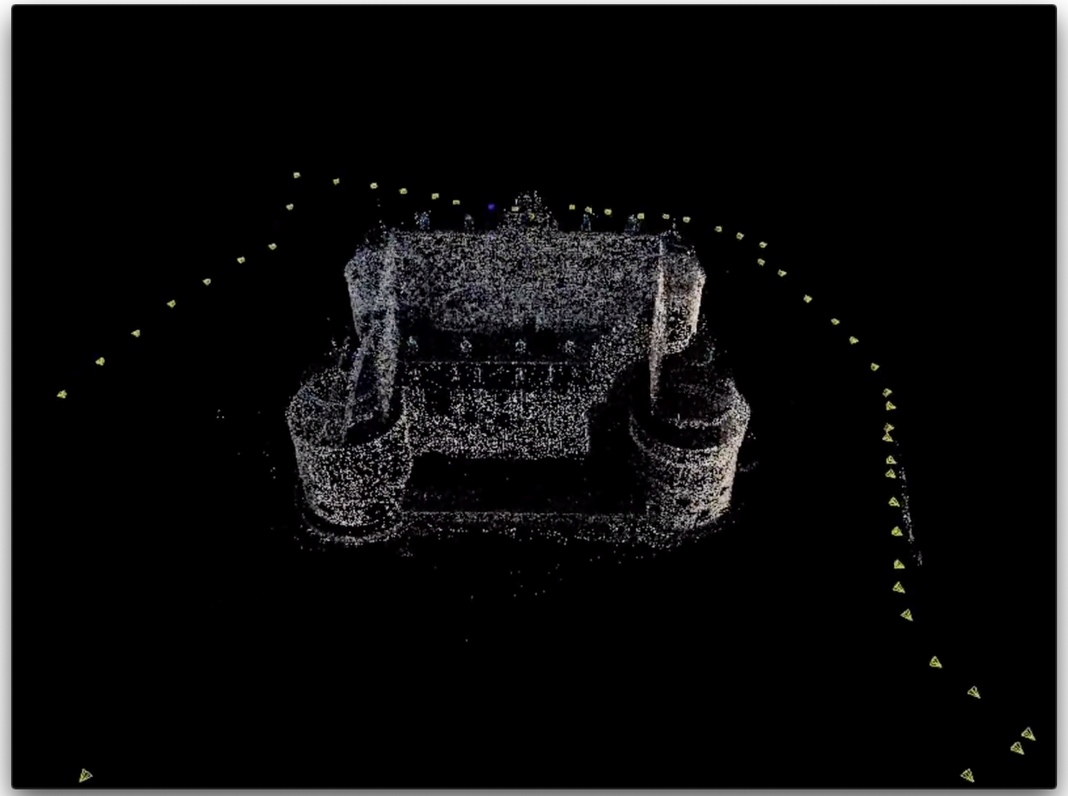
Bundle Adjustment

- This is an incredibly large non-linear minimization
- But it's also very, very sparse
  (lots of views don't see points in lots of others)
- Use the pairwise estimates for $\mathbf{P}_i$ and $\mathbf{p}_j$ to seed the optimization
- Much like we saw with image stitching, we can do the bundle adjustment hierarchically



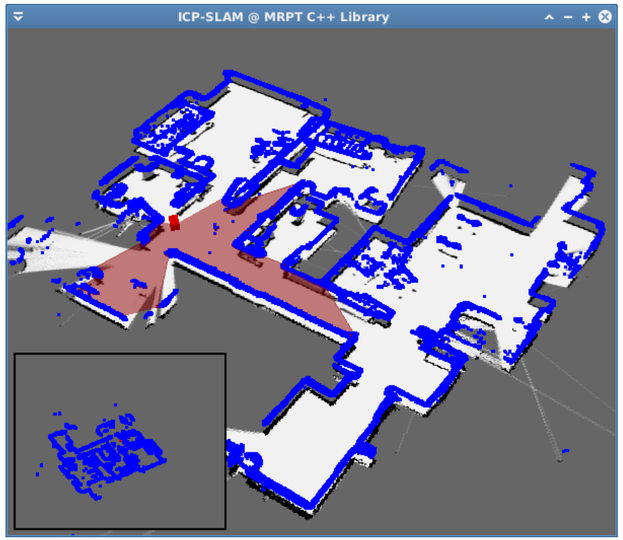
Putting It All Together

- Start with multiple views of a scene
- Find interesting points and construct descriptors
- Match descriptors and reject ambiguous matches
- Simultaneously solve for good matches and **F** using RANSAC
- Incorporate additional knowledge (camera internal calibration, known angles, planes, etc.) to upgrade uncalibrated **F** to **E**
- Decompose **E** to find **R** and **t**
- Use **R** and **t**, along with point correspondences, to triangulate the real-world 3D positions of the sparse feature points
- Bundle adjust poses and 3D points
- If a dense depth map is desired, use known epipolar constraints to constrain correspondence and use stereo (per-point triangulation)



Related Problems/Variants

- Video: *differential* structure from motion
- SLAM: Simultaneous Localization and Mapping
  - Problem in robotics:
    - you don’t know the layout around you (map)
    - you don’t know your own motion
      (may not be what you told it to do)
  - Use ideas similar to SfM to construct both as you go
- Visual odometry:
  - Can you track your own motion solely by what you’re seeing?
  - Can be fused with sensor data
  - Useful in GPS-denied areas



Fun Application: PhotoTourism