Structure from Motion II

CS 4391 Introduction Computer Vision

Professor Yu Xiang

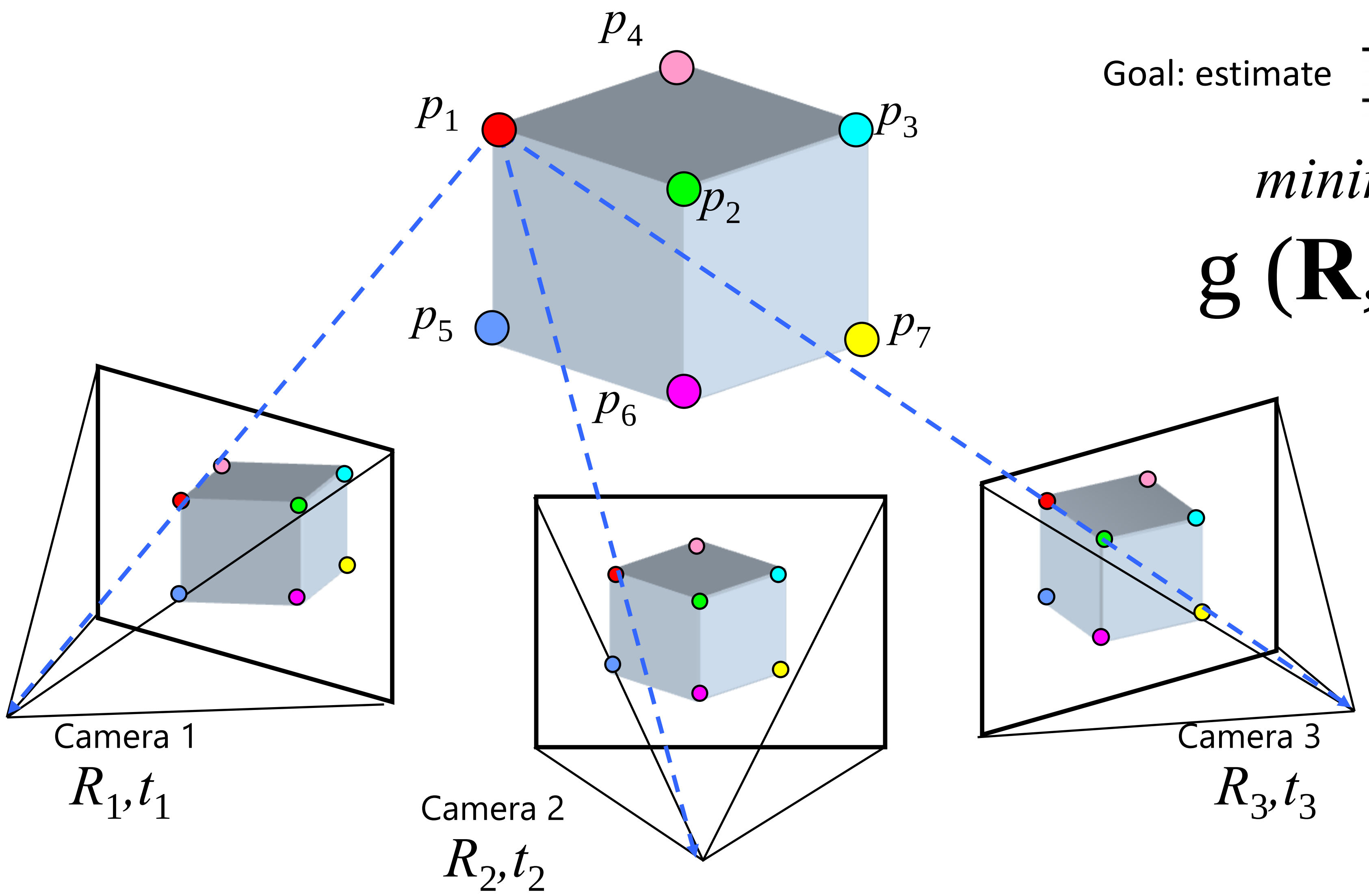
The University of Texas at Dallas

Structure from Motion

- Input
  - A set of images from different views

- Output
  - 3D Locations of all feature points in a world frame
  - Camera poses of the images

Structure from motion



Goal: estimate $\mathbf{R}, \mathbf{T}, \mathbf{P}$

minimize

$$g\,(\mathbf{R}, \mathbf{T}, \mathbf{P})$$

Structure from Motion

- Minimize sum of squared reprojection errors

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\textit{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\textit{observed} \\ \text{image location}}} \right\|^2$$

m points, n images

Indicator variable:
is point i visible in image j?

- A non-linear least squares problem
  - E.g. Levenberg-Marquardt algorithm

The Levenberg-Marquardt Algorithm

- Nonlinear least squares $\hat{\boldsymbol{\beta}} \in \operatorname{argmin}_{\boldsymbol{\beta}} S(\boldsymbol{\beta}) \equiv \operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{m} \left[y_i - f(x_i, \boldsymbol{\beta})\right]^2$

  $n \times 1$

- An iterative algorithm
  - Start with an initial guess β_0
  - For each iteration $\beta \leftarrow \beta + \delta$

Levenberg's contribution $\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I}\right)\boldsymbol{\delta} = \mathbf{J}^{\mathrm{T}}\left[\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\right]$ $\quad \mathbf{J}_i = \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ $\quad 1 \times n$

Wikipedia

Structure from Motion

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \underbrace{w_{ij}}_{\substack{\textit{indicator variable}: \\ \text{is point } i \text{ visible in image } j\text{ ?}}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\textit{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\textit{observed} \\ \text{image location}}} \right\|^2$$

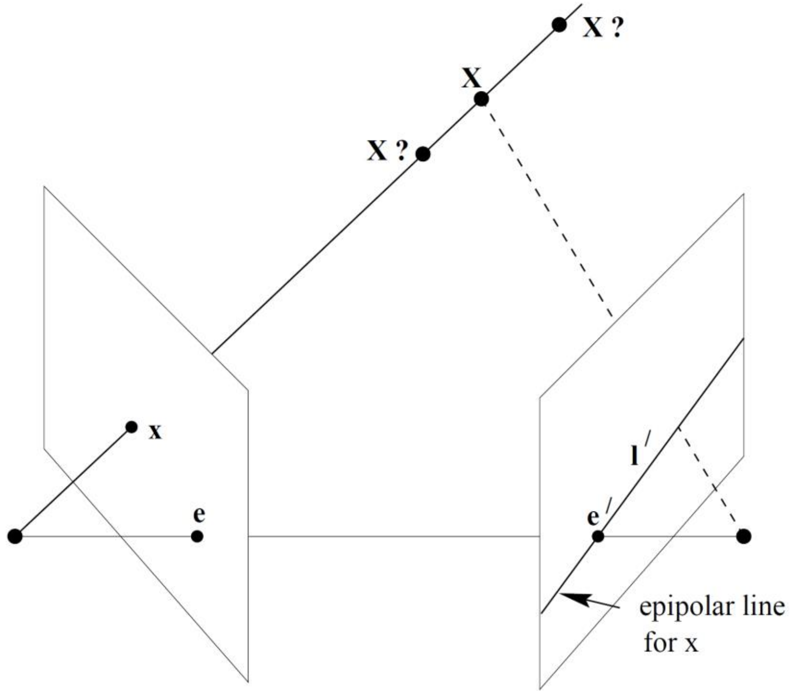
$$\beta = (\mathbf{X}, \mathbf{R}, \mathbf{T})$$

How to get the initial estimation β_0 ?

Random guess is not a good idea.

Matching Two Views

- Fundamental matrix



$\mathbf{x}'$ is on the epiploar line $\mathbf{l}' = F\mathbf{x}$

$$\mathbf{x}'^T F \mathbf{x} = 0$$

The 8-point algorithm

Matching Two Views

$$\mathbf{x}'^T F \mathbf{x} = 0$$

If we know camera intrinsics in SfM

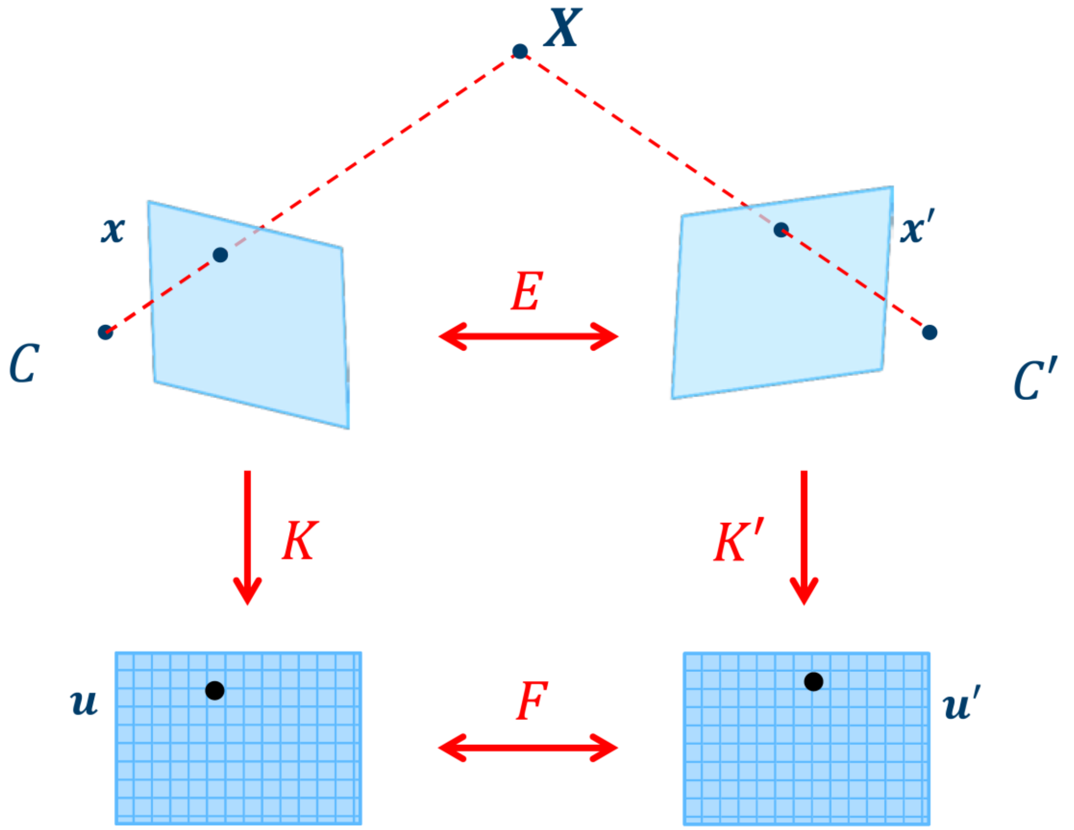
$$(K'^{-1}\mathbf{x}')^T E (K^{-1}\mathbf{x}) = 0$$

Normalized coordinates

$$F = K'^{-T} E K^{-1}$$

- Essential matrix E

$$E = K'^T F K$$



Credit: Thomas Opsahl

Matching Two Views

- Recover the relative pose R and $\boldsymbol{t}$ from the essential matrix E up to the scale of $\boldsymbol{t}$

$$\mathrm{F} = [\mathrm{e}']_{\times}\mathrm{K}'\mathrm{R}\mathrm{K}^{-1} = \mathrm{K}'^{-\mathsf{T}}[\mathrm{t}]_{\times}\mathrm{R}\mathrm{K}^{-1}$$

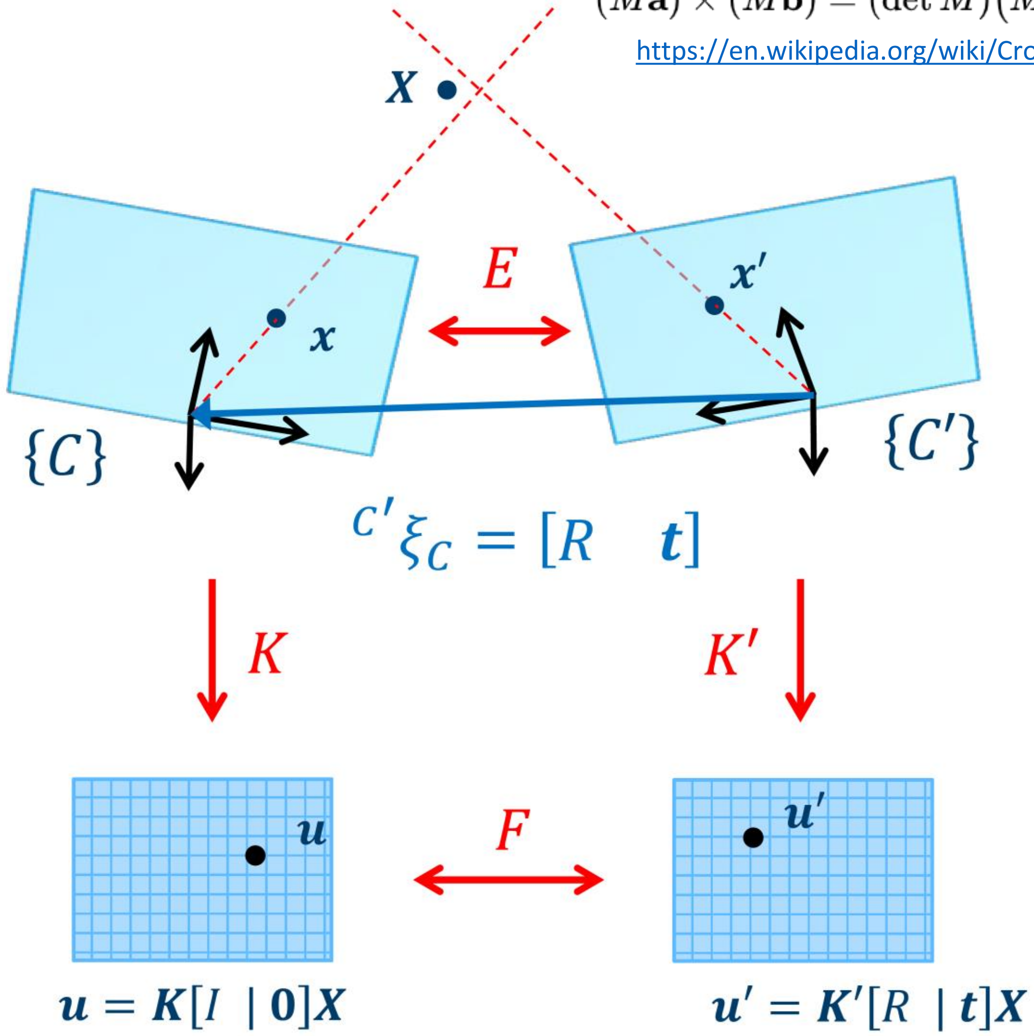
$$E = K'^T F K$$

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

$$(R\mathbf{a}) \times (R\mathbf{b}) = R(\mathbf{a} \times \mathbf{b})$$

$$(M\mathbf{a}) \times (M\mathbf{b}) = (\det M)\left(M^{-1}\right)^{\mathrm{T}}(\mathbf{a} \times \mathbf{b})$$

https://en.wikipedia.org/wiki/Cross_product



Credit: Thomas Opsahl

H. C Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections*, 1981

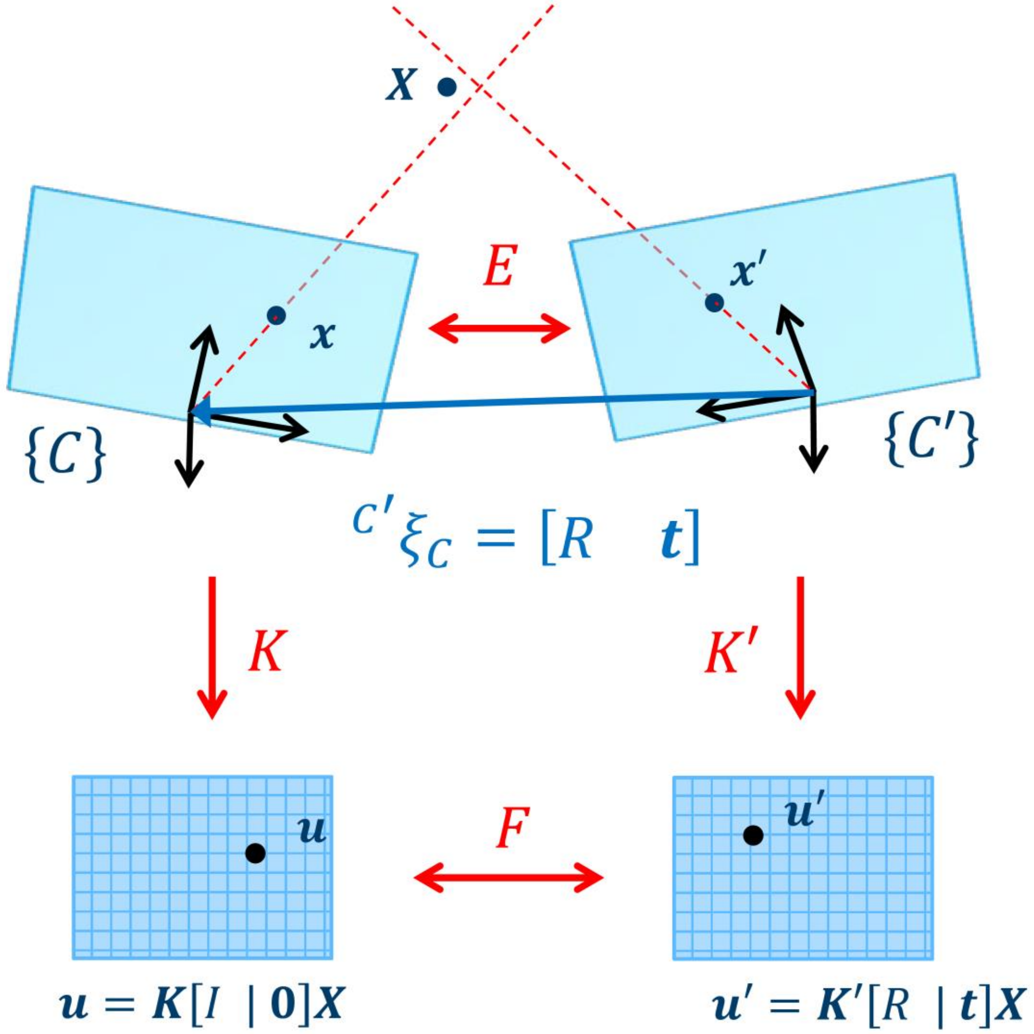
Matching Two Views

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

$$E \cdot \mathbf{t} = [\mathbf{t}]_\times R \cdot \mathbf{t}$$
$$= (\mathbf{t} \times R) \cdot \mathbf{t} = 0$$

Use SVD to solve for **t**

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



Credit: Thomas Opsahl

H. C Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections*, 1981

Matching Two Views

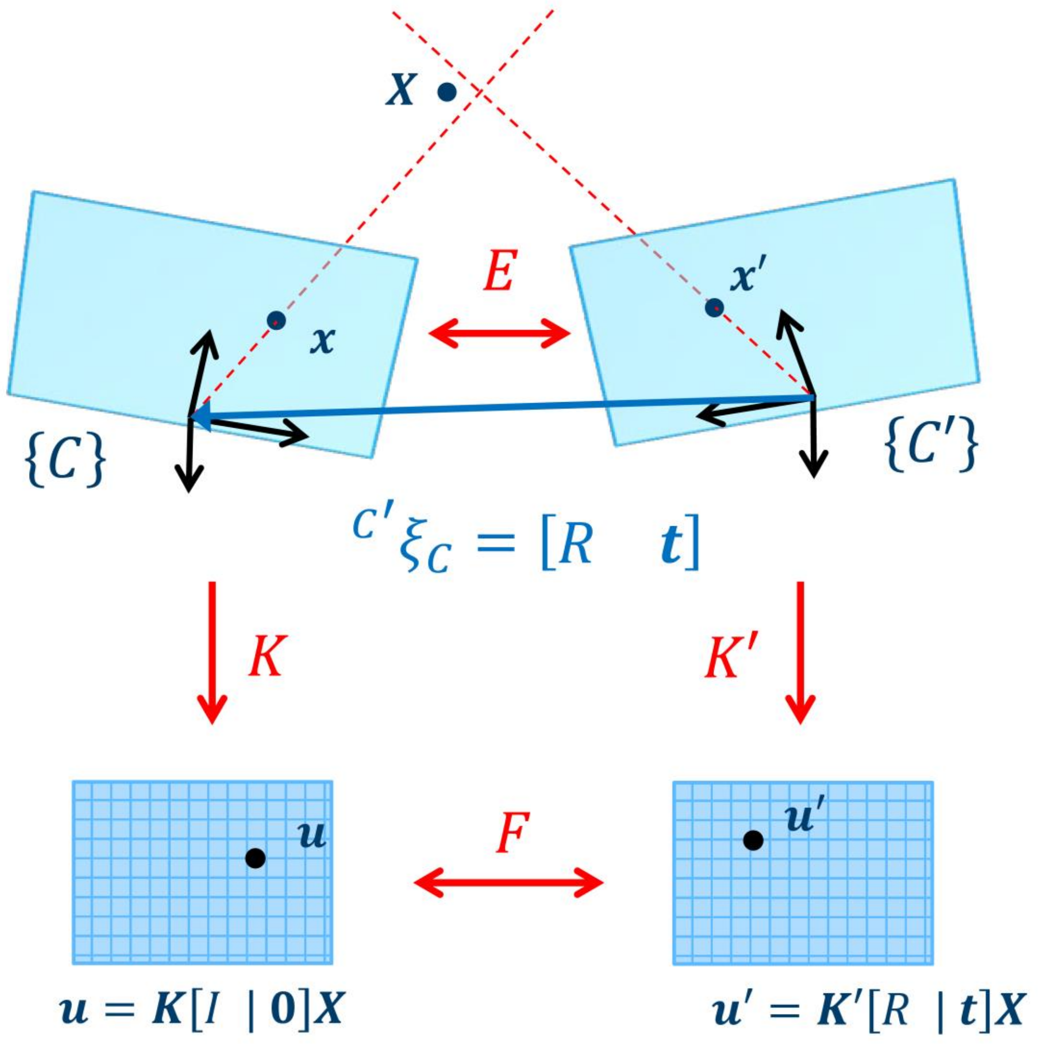
- If we do not know the camera intrinsics

- Work with projection matrix

$$P = [I|\mathbf{0}] \qquad P' = [A|\mathbf{b}]$$

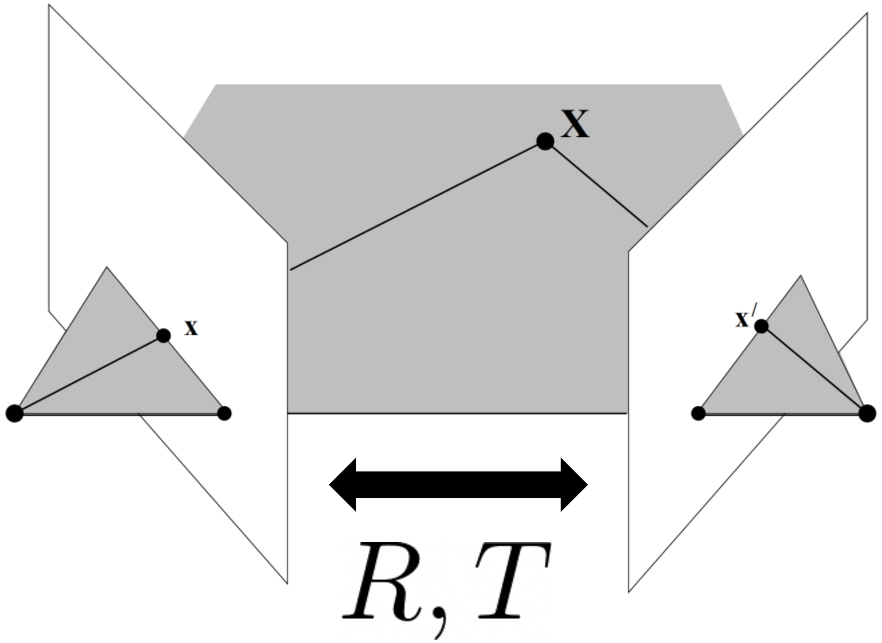
$$\mathbf{x}'^T F \mathbf{x} = 0$$

$$\mathrm{E} = [\mathbf{t}]_\times \mathrm{R} \qquad F = [\mathbf{b}]_\times A$$



Credit: Thomas Opsahl

Triangulation



Estimated from essential matrix E

Intersection of two backprojected lines

$$\mathbf{X} = \mathbf{l} \times \mathbf{l}'$$

How to get the initial estimation β_0 ?

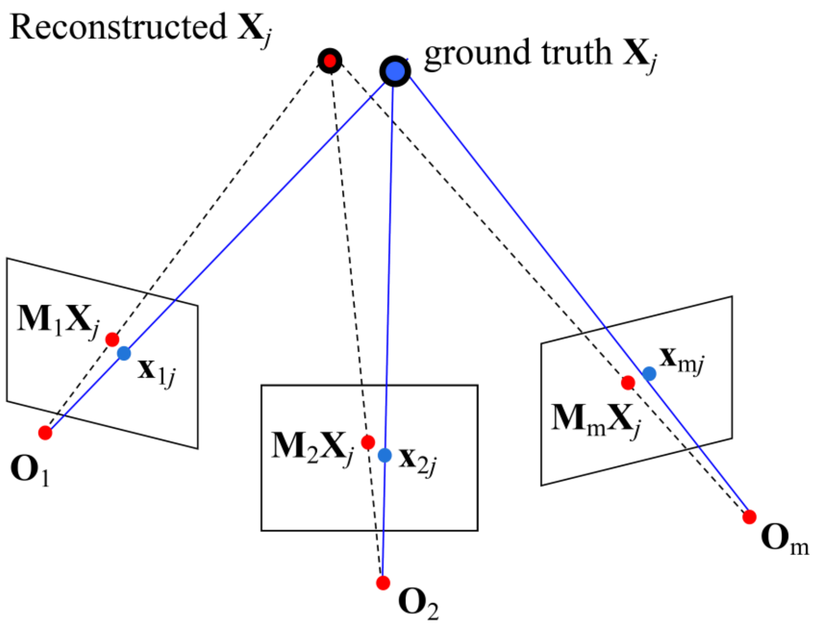
$$\beta = (\mathbf{X}, \mathbf{R}, \mathbf{T})$$

Structure from Motion

$$g(\mathbf{X},\mathbf{R},\mathbf{T}) = \sum_{i=1}^{m}\sum_{j=1}^{n} \underbrace{w_{ij}}_{\substack{\textit{indicator variable}: \\ \text{is point } i \text{ visible in image } j\text{ ?}}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i,\mathbf{R}_j,\mathbf{t}_j)}_{\substack{\textit{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\textit{observed} \\ \text{image location}}} \right\|^2$$

- Bundle adjustment
  - Iteratively refinement of structure (3D points) and motion (camera poses)
  - Levenberg-Marquardt algorithm

$$\beta \leftarrow \beta + \delta$$



Examples: http://vision.soic.indiana.edu/projects/disco/

Build Rome in One Day

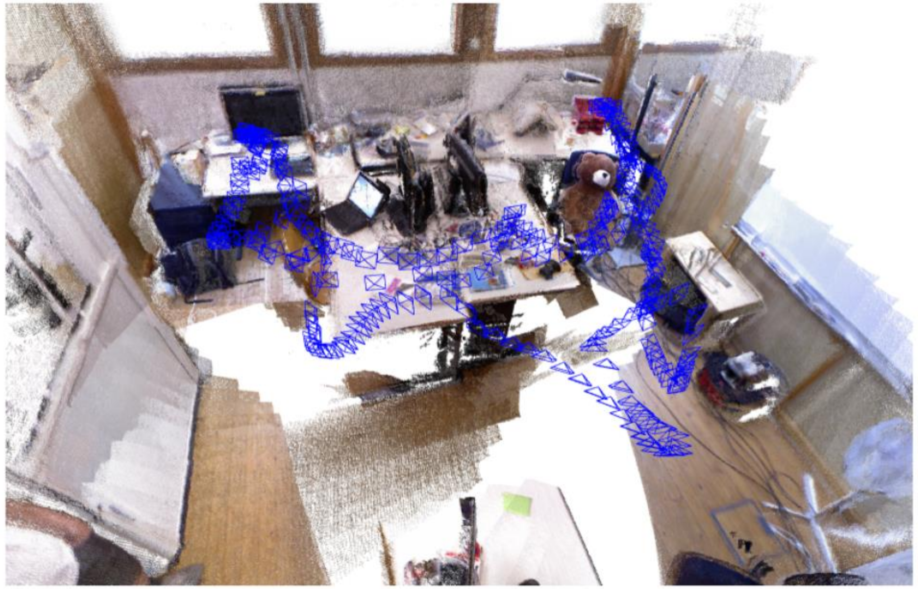
https://grail.cs.washington.edu/rome/

Structure-from-Motion Revisted

https://colmap.github.io/index.html

Simultaneous Localization and Mapping (SLAM)

- Localization: camera pose tracking
- Mapping: building a 2D or 3D representation of the environment
- The goal here is the same as structure from motion but with video input



ORB-SLAM2
- Point cloud and camera poses

Case Study: ORB-SLAM

- Oriented FAST and Rotated BRIEF (ORB)

- Tracking camera poses
  - Motion only Bundle Adjustment (BA)

- Mapping
  - Local BA around camera pose

- Loop closing
  - Loop detection

https://webdiis.unizar.es/~raulmur/orbslam/

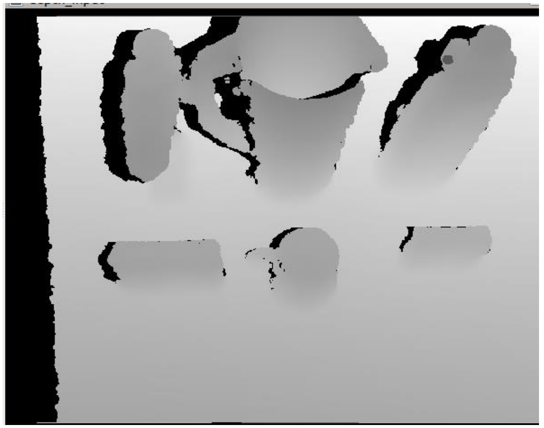
RGB-D SLAM

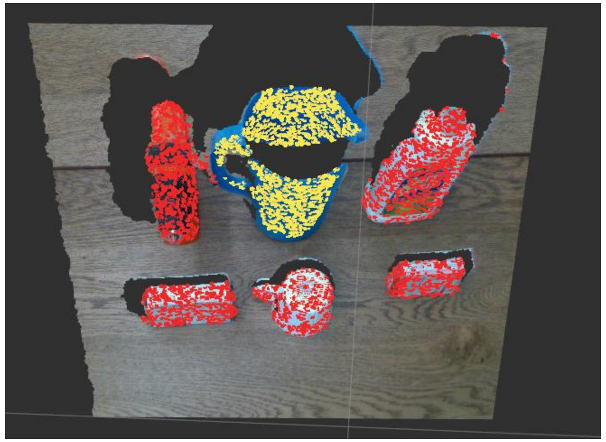
- RGB-D cameras

Microsoft Kinect

Intel RealSense

- Using depth images: 3D points in the camera frame





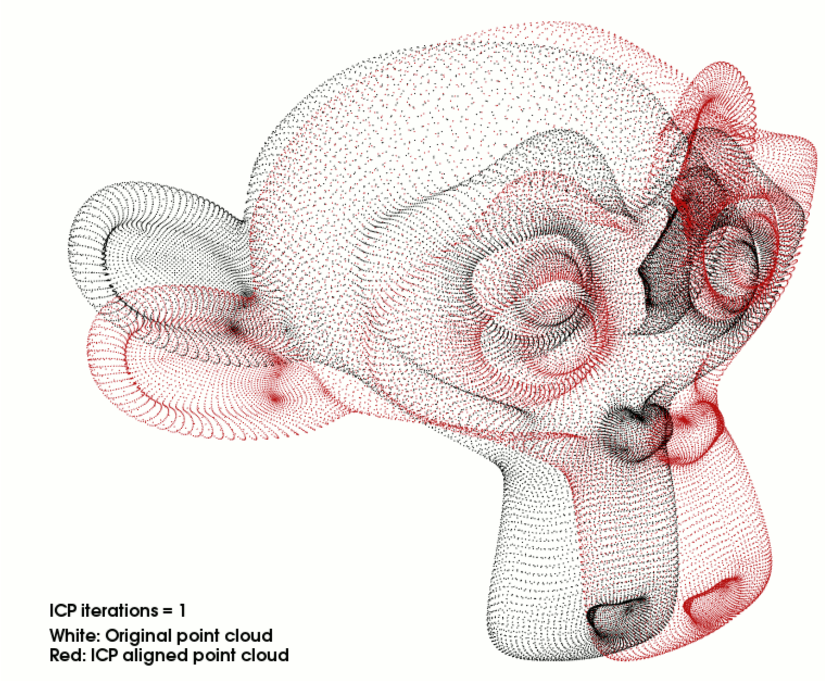
Point Cloud

RGB-D SLAM

- Camera pose tracking
  - Iterative closest point (ICP) algorithm

Input: source point cloud, target point cloud
Output: rigid transformation from source to target

- For i in range(N)
  - For each point in the source, find the closest point in the target (correspondences)
  - Estimation R and T using the correspondences
  - Transform the source points using R and T

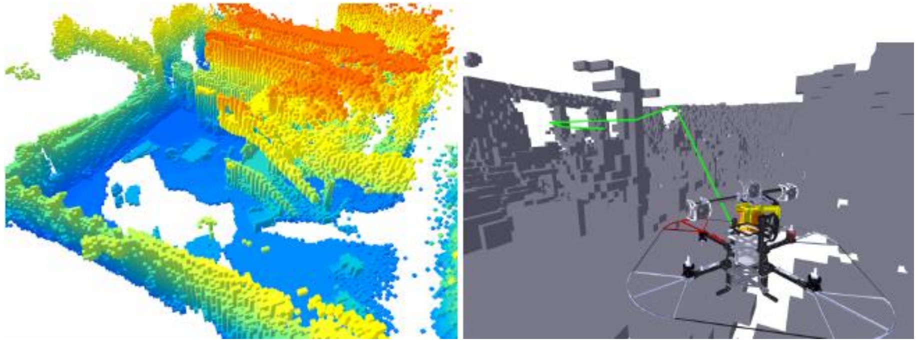


RGB-D SLAM

- Mapping: fuse point clouds into a global frame
- Map representation

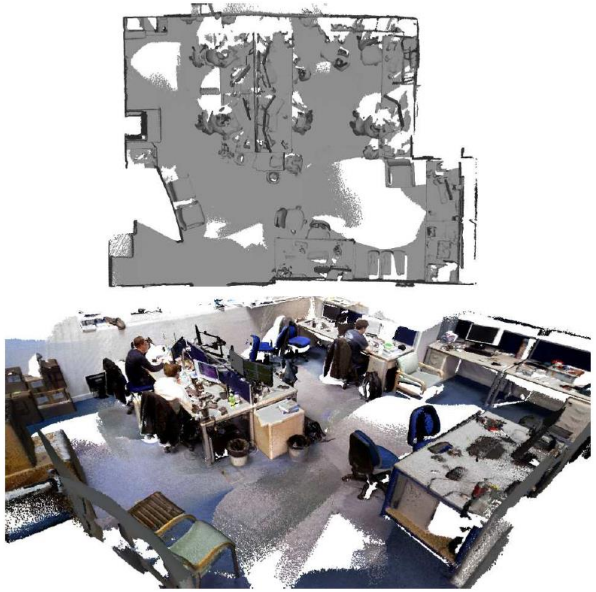
Point clouds

ORB-SLAM



Voxels

Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. Huang, et al. 2011



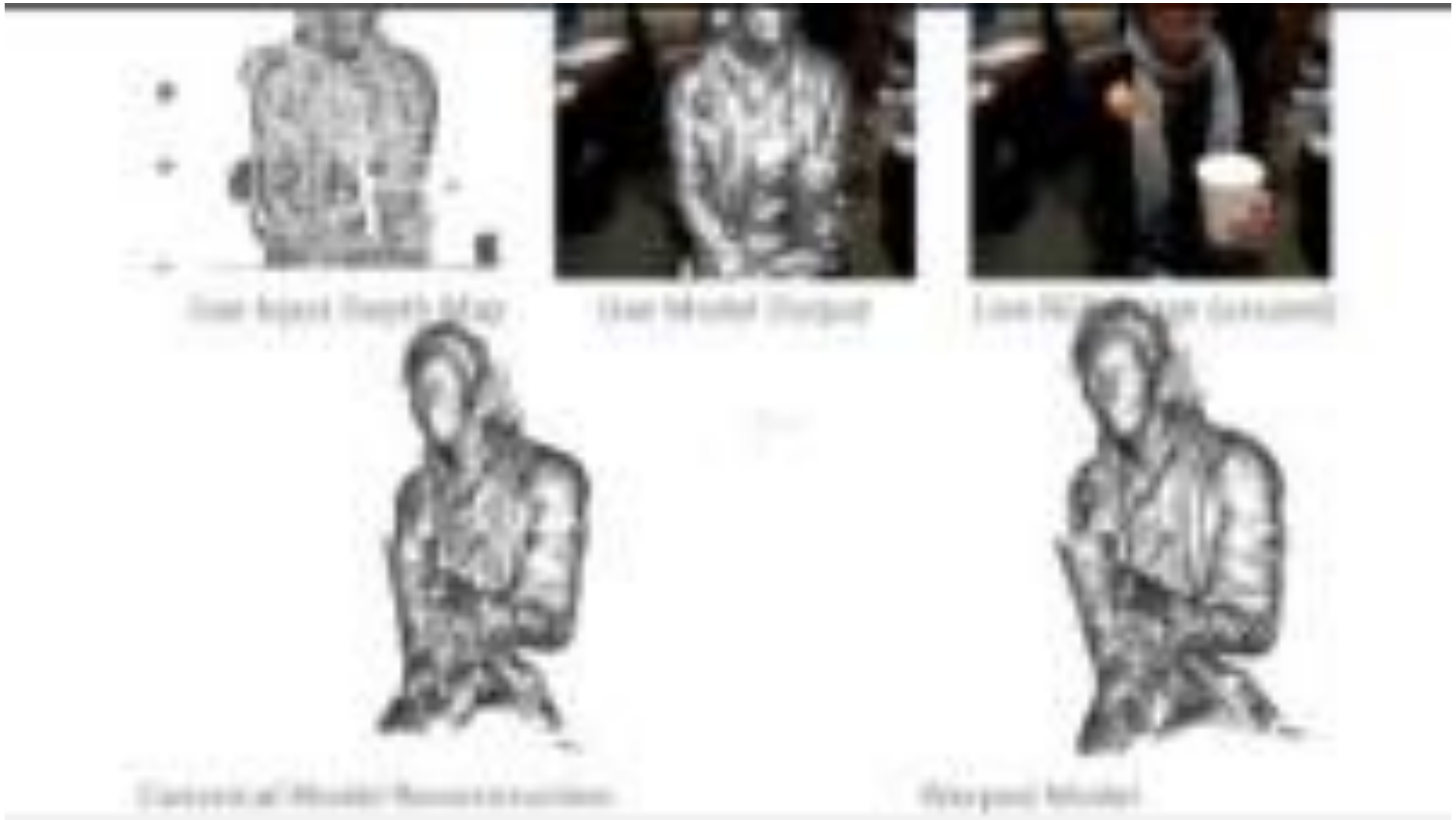
Surfels (small 3D surface)

ElasticFusion

KinectFusion

https://youtu.be/of6d7C_ZWwc

DynamicFusion



A volumetric flow field that transforms the state of the scene at each time instant into a fixed, canonical frame.

https://youtu.be/i1eZekcc_lM

DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time. Newcombe, Fox, Seitz, CVPR’15.

Further Reading

- Chapter 11, Computer Vision, Richard Szeliski
- KinectFusion: Real-Time Dense Surface Mapping and Tracking. Newcombe et al., ISMAR’11
- ORB-SLAM https://webdiis.unizar.es/~raulmur/orbslam/