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 $R, T, P$

minimize $g(R, T, P)$
Structure from Motion

• Minimize sum of squared reprojection errors

\[ g(X, R, T) = \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} \cdot \left\| P(x_i, R_j, t_j) - \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} \right\|^2 \]

- m points, n images
- \( P(x_i, R_j, t_j) \) predicted image location
- \( \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} \) observed image location
- 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
  \[ \hat{\beta} \in \text{argmin}_{\beta} S(\beta) \equiv \text{argmin}_{\beta} \sum_{i=1}^{m} [y_i - f(x_i, \beta)]^2 \]
  \[ n \times 1 \]

• An iterative algorithm
  • Start with an initial guess \( \beta_0 \)
  • For each iteration \( \beta \leftarrow \beta + \delta \)

Levenberg's contribution

\[ (J^T J + \lambda I) \delta = J^T [y - f(\beta)] \]

\[ J_i = \frac{\partial f(x_i, \beta)}{\partial \beta} \]  \[ 1 \times n \]
Structure from Motion

\[ g(X, R, T) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| P(x_{i}, R_{j}, t_{j}) - \left[ u_{i,j} \right] \right\|^2 \]

*indicator variable:*
- is point \( i \) visible in image \( j \)?

\[ \beta = (X, R, T) \]

How to get the initial estimation \( \beta_0 \)?

Random guess is not a good idea.
Matching Two Views

- Fundamental matrix

\[ \mathbf{x}' \text{ is on the epipolar line} \quad \mathbf{1}' = \mathbf{F}\mathbf{x} \]

\[ \mathbf{x}'^T \mathbf{F}\mathbf{x} = 0 \]

The 8-point algorithm
Matching Two Views

\[ x'^{T} F x = 0 \]

If we know camera intrinsics in SfM

\[
(K'^{-1} x')^{T} E (K^{-1} x) = 0
\]

Normalized coordinates

\[ F = K'^{-T} E K^{-1} \]

- Essential matrix E

\[ E = K'^{T} F K \]
Matching Two Views

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

$$F = [e'] \times K'RK^{-1} = K'^{-T}[t] \times RK^{-1}$$

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

$$E = [t] \times R$$


Credit: Thomas Opsahl

$$\begin{align*}
(Ra) \times (Rb) &= R(a \times b) \\
(Ma) \times (Mb) &= (\det M)(M^{-1})^T (a \times b)
\end{align*}$$

Matching Two Views

\[ E = [t] \times R \]

\[ E \cdot t = [t] \times R \cdot t \]

\[ = (t \times R) \cdot t = 0 \]

Use SVD to solve for \( t \)

\[ R = -[t] \times E \]

---


Credit: Thomas Opsahl
Matching Two Views

• If we do not know the camera intrinsics

• Work with projection matrix

\[
P = [I | 0] \quad P' = [A | b]
\]

\[
x'^T F x = 0
\]

\[
E = [t] \times R \quad F = [b] \times A
\]

Credit: Thomas Opsahl
Triangulation

Intersection of two backprojected lines

\[ X = 1 \times 1' \]

How to get the initial estimation \( \beta_0 \)?

\[ \beta = (X, R, T) \]
Structure from Motion

• 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/](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

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

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_IM
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/