Structure from Motion and SLAM

CS 6384 Computer Vision
Professor Yu Xiang
The University of Texas at Dallas
How to Recover the 3D World from Images?

• Structure from Motion (SfM)
  • Structure: the geometry of the 3D world
  • Motion: camera motion
  • Input: a set of images (no need to be videos)
  • From computer vision

• Simultaneous Localization and Mapping (SLAM)
  • Localization: camera pose
  • Mapping: build the geometry of the 3D world
  • Input: video sequences
  • From robotics

Point cloud captured on an Ouster OS1-128 digital lidar sensor
Triangulation

• Idea: using images from different views and feature matching

• Triangulation from pixel correspondences to compute 3D location

Given \( X \leftarrow \rightarrow X' \)

Intersection of two backprojected lines

\[
X = 1 \times 1'
\]

What if unknown camera pose?
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

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

m points, n images

Indicator variable: is point \( i \) visible in image \( j \)?

Projection

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = Rx + t
\]

\[
\begin{align*}
   u' &= f_x \frac{x'}{z'} + p_x \\
v' &= f_y \frac{y'}{z'} + p_y
\end{align*}
\]

\[
\begin{bmatrix}
u' \\
v'
\end{bmatrix} = P(x, R, t)
\]
Structure from Motion

• How to minimize

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

• A non-linear least squares problem (why?)
  • E.g. Levenberg-Marquardt
The Levenberg-Marquardt Algorithm

• Nonlinear least squares
  \[ \hat{\beta} \in \arg\min_{\beta} S(\beta) = \arg\min_{\beta} \sum_{i=1}^{m} [y_i - f(x_i, \beta)]^2 \]
  \[ n \times 1 \]

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

• How to get \( \delta \)?
  • Linear approximation
    \[ f(x_i, \beta + \delta) \approx f(x_i, \beta) + J_i \delta \]
    \[ J_i = \frac{\partial f(x_i, \beta)}{\partial \beta} \]
    \[ 1 \times n \]
  • Find \( \delta \) to minimize the objective
    \[ S(\beta + \delta) \approx \sum_{i=1}^{m} [y_i - f(x_i, \beta) - J_i \delta]^2 \]

Best to minimize the objective
The Levenberg-Marquardt Algorithm

- Vector notation for
  \[ S(\beta + \delta) \approx \sum_{i=1}^{m} [y_i - f(x_i, \beta) - J_i \delta]^2 \]

\[
S(\beta + \delta) \approx \|y - f(\beta) - J\delta\|^2 \\
= [y - f(\beta) - J\delta]^T [y - f(\beta) - J\delta] \\
= [y - f(\beta)]^T [y - f(\beta)] - [y - f(\beta)]^T J\delta - (J\delta)^T [y - f(\beta)] + \delta^T J^T J\delta \\
= [y - f(\beta)]^T [y - f(\beta)] - 2[y - f(\beta)]^T J\delta + \delta^T J^T J\delta.
\]

Take derivation with respect to \( \delta \) and set to zero
\[
(J^T J) \delta = J^T [y - f(\beta)]
\]

Levenberg's contribution
\[
(J^T J + \lambda I) \delta = J^T [y - f(\beta)] \quad \text{damped version}
\]

\( \beta \leftarrow \beta + \delta \)  

Structure from Motion

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

\[ \text{indicator variable: is point } i \text{ visible in image } j? \]

$$\beta = (X, R, T)$$

How to get the initial estimation $\beta_0$?

Random guess is not a good idea.
Matching Two Views

- Fundamental matrix

\[ x' \text{ is on the epiploar line } \quad l' = Fx \]

\[ x'^T Fx = 0 \]

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

\[ E = K'^T F K \]

Credit: Thomas Opsahl
Matching Two Views

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

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

$$E = K'^T FK$$

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

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 \( t \)

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

Matching Two Views

• If we do not know the camera intrinsics

• Work with projection matrix

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

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

\[ F = [b] \times A \]

Credit: Thomas Opsahl
Triangulation

Intersection of two backprojected lines

\[ X = 1 \times 1' \]

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

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

Estimated from essential matrix E
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/
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/
Case Study: ORB-SLAM

• Feature descriptors: Oriented FAST and Rotated BRIEF (ORB)
  • Similar matching performance as SIFT
  • Real-time computation without GPUs

ORB: an efficient alternative to SIFT or SURF. Rublee et al. ICCV’11.
Case Study: ORB-SLAM

- Tracking camera poses
  - Motion only Bundle Adjustment (BA)
    - Huber cost function and covariance matrix associated to the scale of the keypoint

\[
\{ R, t \} = \arg\min_{R, t} \sum_{i \in \mathcal{X}} \rho \left( \left\| x^i_{(\cdot)} - \pi_{(\cdot)} \left(R X^i + t\right) \right\|_\Sigma^2 \right)
\]

- Camera pose
- Detected Keypoint
- 3D point in the map (world coordinates)

Levenberg–Marquardt method

\[
L_\delta(a) = \begin{cases} 
\frac{1}{2} a^2 & \text{for } |a| \leq \delta, \\
\delta(|a| - \frac{1}{2} \delta), & \text{otherwise.}
\end{cases}
\]

Huber loss function
Case Study: ORB-SLAM

- Mapping
  - Local BA around the estimated camera pose
  - Refine 3D point locations

\[
\{X^i, R_l, t_l \mid i \in \mathcal{P}_L, l \in \mathcal{K}_L\} = \arg\min_{X^i, R_l, t_l} \sum_{k \in \mathcal{K}_L \cup \mathcal{K}_F} \sum_{j \in \mathcal{X}_k} \rho(E_{k,j})
\]

\[
E_{k,j} = \left\| x^j_{(.)} - \pi_{(.)} \left( R_k X^j + t_k \right) \right\|_{\Sigma}^2
\]
Case Study: ORB-SLAM

• Loop closing and full BA

Edges from the covisibility graph with high covisibility
Case Study: ORB-SLAM
RGB-D SLAM

- RGB-D cameras

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

Microsoft Kinect

Intel RealSense

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

Surfels (small 3D surface)
ElasticFusion
KinectFusion

https://youtu.be/of6d7C_ZWwc
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
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/