Lecture 8: Robustness and Learning in SFM
Toolkit for Practical SFM

3D-3D 2D-2D Absolute Orientation

2D-3D Pose

2D-2D Relative Orientation

Triangulation

Bundle Adjustment

Robust Statistics
Five-Point Method for Relative Pose Estimation

- Camera 1: $K_1[I | 0]$, camera 2: $K_2[R | t]$

- Fundamental matrix: $F \equiv K_2^{-T}[t] \times RK_1^{-1}$

- For corresponding points $q$ and $q'$, we have $q'^T E q = 0$

- Five degrees of freedom for the essential matrix
- Conditions for a matrix $E$ to be an essential matrix:
  
  $$det(E) = 0$$

  $$EE^T E - \frac{1}{2} \text{trace}(EE^T) E = 0$$

- 10 cubic equations in entries of $E$ (not all independent)
- Reduce 5 epipolar and 10 cubic constraints to single univariate polynomial
Minimal Problems in Computer Vision

- There is a large zoo of minimal problems that have known solutions

5-point relative pose with known $K$

6-point relative pose with unknown $f$

6-point relative pose with unknown $r$

9-point relative pose with unknown and different $f$, $r$
Three-Point Absolute Pose Estimation

- Find camera pose from given 3D-2D correspondences
- Three points suffice to determine pose
- Given $p_1$, $p_2$, $p_3$ in world coordinates, find positions in camera coordinates

**Constraints: Law of cosines**

\[
\begin{align*}
    s_2^2 + s_3^2 - 2s_2s_3 \cos \alpha &= a^2 \\
    s_1^2 + s_3^2 - 2s_1s_3 \cos \beta &= b^2 \\
    s_1^2 + s_2^2 - 2s_1s_2 \cos \gamma &= c^2
\end{align*}
\]

All techniques yield fourth-degree polynomial

Finsterwalder’s technique recommended by [Haralick et al. 1994] as best numerically
Sequential Structure from Motion

- Initialize Motion (P₁, P₂ compatible with F)
- Initialize Structure (minimize reprojection error)
- Extend motion (compute pose through matches seen in 2 or more previous views)
- Extend structure (initialize new structure, refine existing structure)
Step 1: Initialization

- Two views initialization:
  - 5-Point algorithm (Minimal Solver)
  - 8-Point linear algorithm
  - 7-Point algorithm
Step 2: Generate 3D Points

• Triangulation: 3D Points
Step 3: Estimate Pose in Next Views

- Subsequent views: Perspective pose estimation
Real-Time SFM: Steady-State

- Usually absolute pose estimations rather than relative pose

Frame $k-N$  

Frame $k$  

Frame $k+1$

3D-2D correspondences

Frame $k-N$  

Frame $k$  

Frame $k+1$

2D-2D correspondences
Bundle Adjustment

- Refine a visual reconstruction to produce jointly optimal 3D structures $P$ and camera poses $C$.
- Minimize total reprojection errors $\Delta z$.

Cost Function:

$$\arg\min_X \sum_i \sum_j \frac{\left\| x_{ij} - \pi(P_j, C_i) \right\|^2_{W_{ij}}}{W_{ij}}$$

$\Delta z_{ij}$

$W_{ij}^{-1}$: Measurement error covariance

$X = [P, C]$
Bundle Adjustment: Nonlinear Least Squares

Levenberg-Marquardt

Regularized Gauss-Newton with damping factor $\lambda$.

$$\left(J^T W J + \lambda I\right)\delta = -J^T W \Delta Z$$

$H_{LM}$

$\lambda \rightarrow 0$: Gauss-Newton (when convergence is rapid)

$\lambda \rightarrow \infty$: Gradient descent (when convergence is slow)
Bundle Adjustment: Problem Structure

- Primary structure: Camera and 3D point blocks
Bundle Adjustment: Primary Structure

\[ H_{LM} \delta = -J^T W \Delta Z \]

\[
\begin{bmatrix}
H_S & H_{SC} \\
H_{SC}^T & H_C
\end{bmatrix}
\begin{bmatrix}
\delta_s \\
\delta_c
\end{bmatrix}
= 
\begin{bmatrix}
\varepsilon_s \\
\varepsilon_c
\end{bmatrix}
\]

Multiply both sides by:

\[
\begin{bmatrix}
I & 0 \\
-H_{SC}^T H_S^{-1} & I
\end{bmatrix}
\]

\[
\begin{bmatrix}
H_S & H_{SC} \\
0 & H_C - H_{SC}^T H_S^{-1} H_{SC}
\end{bmatrix}
\begin{bmatrix}
\delta_s \\
\delta_c
\end{bmatrix}
= 
\begin{bmatrix}
\varepsilon_s \\
\varepsilon_c - \varepsilon_s H_{SC}^T H_S^{-1}
\end{bmatrix}
\]

First solve for \( \delta_c \) from:

\[
(H_C - H_{SC}^T H_S^{-1} H_{SC}) \delta_c = \varepsilon_c - \varepsilon_s H_{SC}^T H_S^{-1}
\]

(Schur Complement
(Sparse and Symmetric Positive Definite Matrix)

Solve for \( \delta_s \) by backward substitution.
Bundle Adjustment: Robust Loss Functions

<table>
<thead>
<tr>
<th>Cost function</th>
<th>PDF</th>
<th>Attenuation factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Squared-error</td>
<td><img src="image1.png" alt="Graph" /></td>
<td><img src="image2.png" alt="Graph" /></td>
</tr>
<tr>
<td>$C(\delta) = \delta^2$</td>
<td><img src="image3.png" alt="Graph" /></td>
<td><img src="image4.png" alt="Graph" /></td>
</tr>
<tr>
<td>$L1$</td>
<td><img src="image5.png" alt="Graph" /></td>
<td><img src="image6.png" alt="Graph" /></td>
</tr>
<tr>
<td>$C(\delta) = 2b</td>
<td>\delta</td>
<td>$</td>
</tr>
<tr>
<td>Huber</td>
<td><img src="image9.png" alt="Graph" /></td>
<td><img src="image10.png" alt="Graph" /></td>
</tr>
<tr>
<td>$C(\delta) = \begin{cases} \delta^2 &amp; \text{for }</td>
<td>\delta</td>
<td>&lt; b \ 2b</td>
</tr>
</tbody>
</table>
Bundle Adjustment: Robust Loss Functions

- Goal in Levenberg-Marquardt: minimize squared-norm of residual vector
  \[ \sum_i \| x_i - \hat{x}_i \|^2 = \sum_i \| \delta_i \|^2 = \| \Delta \|^2 \]

- A robust cost function minimizes an outlier-tolerant function of residuals
  \[ \sum_i C'(||\delta_i||) \]

- Can plug robust norm into a solver by replacing the residuals with \( \delta'_i = w_i \delta_i \)
  \[ ||\delta'_i||^2 = w_i^2 ||\delta_i||^2 = C(||\delta_i||) \]
  \[ w_i = \frac{C(||\delta_i||)^{1/2}}{||\delta_i||} \]

- Attenuation weight large for inliers, decreases rapidly for outliers
Keyframes

- Provide a large enough baseline for triangulation

Wide baseline  Narrow baseline
Keyframes

- Provide a large enough baseline for triangulation
- Add new 3D points to point cloud at the keyframe
- Collect long tracks, triangulate and do nonlinear refinement
Keyframes

- Provide a large enough baseline for triangulation
- Add new 3D points to point cloud at the keyframe
- Collect long tracks, triangulate and do nonlinear refinement
- Do a global bundle adjustment over keyframes and associated 3D points
Keyframes

- Provide a large enough baseline for triangulation
- Add new 3D points to point cloud at the keyframe
- Collect long tracks, triangulate and do nonlinear refinement
- Do a global bundle adjustment over keyframes and associated 3D points

Criteria to add keyframes
- Camera has moved far enough: \[
\frac{\text{keyframe distance}}{\text{average-depth}} > \text{threshold}
\]
- Keyframe not added for long enough
- Number of remaining 3D points not large enough
Scale Drift Correction

Choose scale arbitrarily, for example, translation from frame 1 to 2

Frame 1  
Set scale as $t = 1$

Frame 2

Frame 3  
Estimate scaled $t = t_0$
Choose scale arbitrarily, for example, translation from frame 1 to 2.

Set scale as $t = 1$ and estimate scaled $t = t_0$.

Every bundle adjustment causes 3D points and cameras to change.

$t = 1 + \Delta$  \hspace{1cm}  $t = t_0 + \Delta'$
**Challenge**: We can compute 3D locations and camera motion, *up to unknown scale factor*. 

**Resolution**: compute height of ground plane.
Real-Time SFM Systems

- Use fundamental building blocks, with different system choices
Loop Closure
Indexing local features

• When we see close points in feature space, we have similar descriptors, which indicates similar local content.
Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images by frequencies of “visual words”
Learning the visual vocabulary

Visual vocabulary

Clustering

Slide credit: Josef Sivic
Comparing bags of words

• Rank frames by normalized scalar product between their (possibly weighted) occurrence counts---nearest neighbor search for similar images.

\[ \text{sim}(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|} \]

\[ = \frac{\sum_{i=1}^{V} d_j(i) * q(i)}{\sqrt{\sum_{i=1}^{V} d_j(i)^2} * \sqrt{\sum_{i=1}^{V} q(i)^2}} \]

for vocabulary of \(V\) words

[Kristen Grauman]
Some Recipes for SFM to Work

• Do everything you can to remove outliers

• Solve minimal problems to estimate geometric entities
  • Keeps RANSAC tractable
  • Typically, expect to spend less than 0.1ms

• Strategically consider what variables to optimize
  • Keyframe-based designs are successful
  • Try to robustly build long feature tracks
  • Do bundle adjustment whenever possible

• Drift is inevitable, so have a plan to address it
  • Local scale correction and global pose correction when possible
Learning Structure and Motion
Typical Way to Learn Structure and Motion

- Estimate depths (convert to 3D points given calibration) in frame $t$
- Estimate motion from frame $t$ to $t+1$ for background and objects
- Project 3D points to frame $t+1$ using the estimated motions
- Use a consistency condition to declare matches as good

[SFM-Net, ICCV 2017]
Estimate Depth in an Image

- Depth in frame \( t \) : \( d_t \), camera pose: \( \{ R^c_t, t^c_t \} \), K object motions: \( \{ R^k_t, t^k_t \} \)

- Standard encoder-decoder for depth estimation

- For pixel \((x^i_t, y^i_t)\), with camera intrinsics \((c_x, c_y, f)\), 3D points are

\[
X^i_t = \begin{bmatrix}
X^i_t \\
Y^i_t \\
Z^i_t
\end{bmatrix} = \frac{d^i_t}{f} \begin{bmatrix}
x^i_t - c_x \\
y^i_t - c_y \\
f
\end{bmatrix}
\]

[SFM-Net, ICCV 2017]
Predict Motion across Frames

- Motion network takes frames $t$ and $t+1$ as input
- Two fully-connected layers to predict camera and $K$ object motions
- Transposed convolution on same embedding to get motion masks
- Same pixel can belong to multiple motions (articulations), $m^k_t \in [0, 1]^{(h \times w)}$

[SFM-Net, ICCV 2017]
Estimate Displacement in Image

- Apply object transformations on 3D points in frame $t$:
  \[
  X'_t = X_t + \sum_{k=1}^{K} m^k_t(i)(R^k_t(X_t - p_k) + t^k_t - X_t)
  \]

- Apply camera transformations on 3D points:
  \[
  X''_t = R^c_t(X'_t - p^c_t) + t^c_t
  \]

- Project to frame $t+1$ and compute flow vector:
  \[
  \begin{bmatrix}
  x_{t+1}^i \\
  y_{t+1}^i \\
  z_{t+1}^i
  \end{bmatrix}
  = \frac{f}{Z''_t}
  \begin{bmatrix}
  X''_t \\
  Y''_t \\
  f
  \end{bmatrix}
  + \begin{bmatrix}
  c_x \\
  c_y
  \end{bmatrix}
  \\
  (U_t(i), V_t(i)) = (x_{t+1}^i - x_i^i, y_{t+1}^i - y_i^i)
  \]

[SFM-Net, ICCV 2017]
Types of Supervision: Photoconsistency

- Image intensity at a point remains the same
- Structure and motion estimates most consistent with above
- Minimize the error:

\[
\sum_{x,y} \| I_t(x,y) - I_{t+1}(x',y') \|_1
\]

\[
x' = x + U_t(x,y) \\
y' = y + V_t(x,y)
\]
Types of Supervision: Left-Right Consistency

- Depths of same point in two frames must be consistent with motion
- Estimate depths in frame \( t \)
- Apply estimated motion to 3D points
- Estimate depth in frame \( t+1 \) at flow-displaced location
- Minimize error:

\[
| (d_t(x, y) + W_t(x, y)) - d_{t+1}(x + U_t(x, y), y + V_t(x, y)) |
\]
Types of Supervision: Ground Truth

- In some cases, ground truth depth is known (LIDAR, Kinect) at some points
  \[
  \sum_{x,y} \text{dmask}_t^{GT}(x,y) \cdot \| d_t(x,y) - d_t^{GT}(x,y) \|_1
  \]

- Sometimes, ground truth motion is known (GPS, IMU)
  \[
  R_t^{err} = \text{inv}(R_t^c) R_t^{c-GT}, \quad t_t^{err} = \text{inv}(R_t^c)(t_t^{c-GT} - t_t^c)
  \]