Virtual classrooms

• Virtual lectures on Zoom
  – Only host shares the screen
  – Keep video off and microphone muted
  – But please do speak up (remember to unmute!)

• Virtual interactions on Zoom
  – Ask and answer plenty of questions
  – “Raise hand” feature on Zoom when you wish to speak
  – Post questions on chat window
  – TA will help keep track of raised hands and chat window

• Lectures recorded and upload on Kaltura
  – Available under “Media Gallery” on Canvas
Course details

• Class webpage:
  – http://cseweb.ucsd.edu/~mkchandraker/classes/CSE152B/Spring2020/

• Instructor email:
  – mkchandraker@eng.ucsd.edu

• TA: Rui Zhu
  – Emails: rzhu@eng.ucsd.edu

• Aim is to learn together, discuss and have fun!

• Assignment 1 has been released (see the class webpage)
Recap
Unordered or Ordered Images
Feature detection

Several images observe a scene from different viewpoints
Feature detection

Detect features using, for example, SIFT [Lowe, IJCV 2004]
Feature matching

Match features between each pair of images
Structure from motion

\[
\Pi_1 X_1 \sim p_{11}
\]

minimize \( g(R, T, X) \)

non-linear least squares

\[
\begin{align*}
R_1, t_1 & \quad \text{Camera 1} \\
R_2, t_2 & \quad \text{Camera 2} \\
R_3, t_3 & \quad \text{Camera 3}
\end{align*}
\]

CSE 152B, SP20: Manmohan Chandraker
Toolkit for Practical SFM

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

2D-3D
Pose

2D-2D
Relative Orientation

Triangulation

Bundle Adjustment

Robust Statistics

CSE 152B, SP20: Manmohan Chandraker
Projection matrix

\[
\Pi = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} I_{3 \times 3} & -c \\ 0 & 0 & 0 & 1 \end{bmatrix} = K \begin{bmatrix} R & -Rc \end{bmatrix}
\]

Denote this by \( t \)
Fundamental Matrix

For any two corresponding points across two images:

\[ x_1 \leftrightarrow x_2 \]

\[ x_1^T F x_2 = 0 \]
Direct Linear Transform Method

Given $n$ point correspondences, set up a system of equations:

$$
\begin{pmatrix}
    u_1 u_1' & v_1 u_1' & u_1' & u_1 v_1' & v_1 v_1' & v_1' & u_1 & v_1 & 1 \\
    u_2 u_2' & v_2 u_2' & u_2' & u_2 v_2' & v_2 v_2' & v_2' & u_2 & v_2 & 1 \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    u_n u_n' & v_n u_n' & u_n' & u_n v_n' & v_n v_n' & v_n' & u_n & v_n & 1 \\
\end{pmatrix} \begin{pmatrix}
    f_{11} \\
    f_{12} \\
    f_{13} \\
    f_{21} \\
    f_{22} \\
    f_{23} \\
    f_{31} \\
    f_{32} \\
    f_{33}
\end{pmatrix} = 0
$$

- Minimize $\|A f\|$, using SVD
- Enforce rank of $F$ to be 2, by setting third singular value to 0.
Motion from correspondences

• Use 8-point algorithm to estimate $F$

• Get $E$ from $F$:

\[
F = K_2^{-\top} E K_1^{-1}
\]
\[
E = K_2 \top F K_1
\]

• Decompose $E$ into skew-symmetric and rotation matrices:

\[
E = [t] \times R
\]
Four Possible Solutions

1. **Baseline reversal**
2. **Rotate camera B by 180 degrees**
Triangulation
Goal: ensure correspondences used for estimating F do not contain outliers.
RANSAC: Counting Inliers
RANSAC: Counting Inliers

Inliers: 3

CSE 152B, SP20: Manmohan Chandraker
RANSAC: Counting Inliers

Inliers: 20

CSE 152B, SP20: Manmohan Chandraker
How do we find the best line?

• Unlike least-squares, no simple closed-form solution

• Hypothesize-and-test
  – Try out many lines, keep the best one
  – RANSAC: Random Sample Consensus

• Number of samples depends on
  – Outlier ratio
  – Probability of correct answer
  – Model size
RANSAC

• General version:
  1. Randomly choose $s$ samples
     (Typically $s =$ minimum sample size to fit a model)
  2. Fit a model to those samples
  3. Count number of inliers consistent with the model
  4. Repeat $N$ times
  5. Choose the model with the largest set of inliers
Fundamental Matrix

$x_1 \leftrightarrow x_2$

\[ x_1^T F x_2 = 0 \]
Fundamental Matrix

\[ x_1 \leftrightarrow x_2 \]

\[ x_1^T F x_2 = 0 \]

Degrees of freedom for \( F \): 7

So, 7 points suffice to find \( F \), but use 8 for linear method.
RANSAC to Estimate Fundamental Matrix

• For $N$ times
  – Randomly pick 8 correspondences across two images
  – Compute $\mathbf{F}$ using these 8 correspondences
  – Among all correspondences, count number of inliers with $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2$ close to 0 (less than a threshold)

• Pick the $\mathbf{F}$ with the largest number of inliers
RANSAC: Number of Iterations

- Adaptively determine number of iterations based on outlier proportion

\[ N = \frac{\log(1 - p)}{\log(1 - (1 - \epsilon)^s)} \]

<table>
<thead>
<tr>
<th>Sample size</th>
<th>Proportion of outliers $\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>5%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

Values of $N$ for $p = 0.99$
Bundle adjustment

• Minimize sum of squared reprojection errors:

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

- Cost function defined over all cameras and 3D points
- Optimized with non-linear least squares
- Levenberg-Marquardt is a popular choice

• Practical challenges?
  - Initialization: triangulate \( x \), using \((R, t)\) from decomposition of \( F \)
  - Outliers: use RANSAC for estimation of \( F \)
Some Actually Practical Steps
Calibrated Relative Pose Estimation

- Camera 1: \( K_1[I \mid 0] \), camera 2: \( K_2[R \mid t] \)
- Fundamental matrix: \( F \equiv K_2^{-1}R K_1^{-1} \)

- For corresponding points \( q \) and \( q' \), we have \( q'^\top F q = 0 \)
- Condition for a matrix \( F \) to be a fundamental matrix: \( \det(F) = 0 \)
Calibrated Relative Pose Estimation

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

- Fundamental matrix: $F \equiv K_2^{-T}[t]_x R K_1^{-1}$

- Essential matrix, $E$

- For corresponding points $q$ and $q'$, we have $q'\top Fq = 0$

- Condition for a matrix $F$ to be a fundamental matrix: $det(F) = 0$
Five-Point Method for Relative Pose Estimation

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

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

- For corresponding points $q$ and $q'$, we have $q^{\top}Fq = 0$

- Condition for a matrix $F$ to be a fundamental matrix: $det(F) = 0$

- Five degrees of freedom for the essential matrix, $E$
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 $q'^T F q = 0$

- Condition for a matrix $F$ to be a fundamental matrix: $det(F) = 0$

- Five degrees of freedom for the essential matrix, $E$
- Estimating $E$ needs 5 correspondences, better for RANSAC ($F$ needs 7 or 8)
- Goal: should be able to estimate $E$ efficiently from 5 correspondences
Five-Point Method for Relative Pose Estimation

• Camera 1: $K_1[I \mid 0]$, camera 2: $K_2[R \mid t]$

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

• For corresponding points $q$ and $q'$, we have $q'^{T}Fq = 0$

• Condition for a matrix $F$ to be a fundamental matrix: $det(F) = 0$

• Five degrees of freedom for the essential matrix, $E$

• Estimating $E$ needs 5 correspondences, better for RANSAC ($F$ needs 7 or 8)

• Goal: should be able to estimate $E$ efficiently from 5 correspondences

• Conditions for a matrix $E$ to be an essential matrix:

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

10 cubic equations in entries of $E$
(not all independent)
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 $f$, $r$
Minimal Problems in Computer Vision

- There is a large zoo of minimal problems that have known solutions
Three-Point Absolute Pose Estimation

• Find camera pose from given 3D-2D correspondences
• Minimal case is 3 points: 6 degrees of freedom, 2 constraints per point (x, y)
• Given $p_1$, $p_2$, $p_3$ in world coordinates, find positions in camera coordinates

Determine how to move camera, such that corresponding 2D-3D points align
Three-Point Absolute Pose Estimation

- Find camera pose from given 3D-2D correspondences
- Minimal case is 3 points: 6 degrees of freedom, 2 constraints per point \((x, y)\)
- Given \(p_1, p_2, p_3\) in world coordinates, find positions in camera coordinates

\[
\begin{align*}
    a &= \|p_2 - p_3\| \\
    b &= \|p_1 - p_3\| \\
    c &= \|p_1 - p_2\|.
\end{align*}
\]

\[
\begin{align*}
    u_i &= f \frac{x_i}{z_i} \\
    v_i &= f \frac{y_i}{z_i}.
\end{align*}
\]

Knowns

Distances between points  Image points  Angle between image rays \(j_i\)

\[
\begin{align*}
    \cos \alpha &= j_2 \cdot j_3 \\
    \cos \beta &= j_1 \cdot j_3 \\
    \cos \gamma &= j_1 \cdot j_2
\end{align*}
\]

Unknowns

Distance to 3D points from camera center, \(s_i\)

CSE 152B, SP20: Manmohan Chandraker
Three-Point Absolute Pose Estimation

- Find camera pose from given 3D-2D correspondences
- Minimal case is 3 points: 6 degrees of freedom, 2 constraints per point \((x, y)\)
- Given \(p_1, p_2, p_3\) in world coordinates, find positions in camera coordinates

**Constraints: Law of cosines**

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

**Knowns**

\[
\begin{align*}
    a &= \|p_2 - p_3\| \\
    b &= \|p_1 - p_3\| \\
    c &= \|p_1 - p_2\|.
\end{align*}
\]

\[
\begin{align*}
    u_i &= f \frac{x_i}{z_i} \\
    v_i &= f \frac{y_i}{z_i}.
\end{align*}
\]

Distance between points \(\cos \alpha = j_2 \cdot j_3\)

Image points \(\cos \beta = j_1 \cdot j_3\)

Angle between image rays \(\cos \gamma = j_1 \cdot j_2\)

**Unknowns**

Distance to 3D points from camera center, \(s_i\)

All techniques yield fourth-degree polynomial
Three-Point Absolute Pose Estimation

- Find camera pose from given 3D-2D correspondences
- Minimal case is 3 points: 6 degrees of freedom, 2 constraints per point \((x, y)\)
- 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**
  - $(P_1, P_2)$ 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
Real-Time SFM Systems
Real-Time SFM Systems

- Use fundamental building blocks, with different system choices
Real-Time SFM Systems

- Use fundamental building blocks, with different system choices

[Diagram showing Real-Time SFM Systems]

- Feature detection and matching
- Map Initialization
- PnP absolute pose

[CSE 152B, SP20: Manmohan Chandraker]

[Mur-Artal et al., ORB-SLAM]
Real-Time SFM Systems

- Use fundamental building blocks, with different system choices

[CSE 152B, SP20: Manmohan Chandraker]
Real-Time SFM: Steady-State

- Point cloud and camera poses available at frame k, find pose in frame k+1

[Song et al., Parallel Real-Time Monocular Visual Odometry]
Real-Time SFM: Steady-State

- Usually absolute pose estimations rather than relative pose
Real-Time SFM: Steady-State

- Detect features in frame k+1
Real-Time SFM: Steady-State

- Project 3D points to frame $k+1$ using motion model based on previous poses

Assume constant velocity to guess pose in frame $k+1$
Real-Time SFM: Steady-State

- Match projected and detected features efficiently in a small window
Real-Time SFM: Steady-State

- Retain matched points as 3D-2D correspondences
Step 1: Initialization

- **Two views initialization:**
  - 5-Point algorithm (Minimal Solver)
  - 8-Point linear algorithm
  - 7-Point algorithm

\[ E \rightarrow (R,t) \]
Step 2: Generate 3D Points

- Triangulation: 3D Points

\[ E \rightarrow (R,t) \]
Step 3: Estimate Pose in Next Views

- Subsequent views: Perspective pose estimation
Intermediate Step: 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:

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

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

$$X = [P, C]$$

$$\arg\min_X \sum_i \sum_j \Delta z_{ij}^T W_{ij} \Delta z_{ij}$$

$f(X)$

CSE 152B, SP20: Manmohan Chandraker
Choose scale arbitrarily, for example, translation from frame 1 to 2

Set scale as \( t = 1 \)

Estimate scaled \( t = t_0 \)
Choose scale arbitrarily, for example, translation from frame 1 to 2.

Every bundle adjustment causes 3D points and cameras to change.
Challenge: We can compute 3D locations and camera motion, *up to unknown scale factor*.

Resolution: Compute height of ground plane.
Practical SFM Systems

- Use fundamental building blocks, with different system choices
Global Scale Correction: 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

SIFT descriptors for patches in all images of dataset

Clustering

Visual vocabulary

Most “representative” descriptors

CSE 152B, SP20: Manmohan Chandraker

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.

\[
\text{sim}(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|} = \frac{\sum_{i=1}^{V} d_j(i) \cdot q(i)}{\sqrt{\sum_{i=1}^{V} d_j(i)^2} \cdot \sqrt{\sum_{i=1}^{V} q(i)^2}}
\]

for vocabulary of \( V \) words
Inverted file index

- Database images are loaded into the index mapping words to image numbers

CSE 152B, SP20: Manmohan Chandraker
Inverted file index and bags of words similarity

1. Extract words in query
2. Inverted file index to find relevant frames
3. Compare word counts
Hierarchical clustering for large vocabularies

- Tree construction:
Vocabulary tree

- Training: Filling the tree

[Nistér and Stewenius, Scalable Recognition with a Vocabulary Tree]
Vocabulary tree

- Training: Filling the tree

[CSE 152B, SP20: Manmohan Chandraker
[Nistér and Stewenius, Scalable Recognition with a Vocabulary Tree]
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 0.01ms

• 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
General Recipe 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 estimated 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: $\{R_t^c, t_t^c\}$, K object motions: $\{R_t^k, t_t^k\}$
- Standard encoder-decoder for depth estimation
- For pixel $(x_t^i, y_t^i)$, with camera intrinsics $(c_x, c_y, f)$, 3D points are

$$X_t^i = \begin{bmatrix} X_t^i \\ Y_t^i \\ Z_t^i \end{bmatrix} = \frac{d_t}{f} \begin{bmatrix} x_t^i - c_x \\ y_t^i - c_y \\ f \end{bmatrix}$$
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:
  \[ X'_t = X_t + \sum_{k=1}^{K} m_k(i) (R^k_t (X_t - p_k) + t^k_t - X_t) \]

- 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:
  \[
  \begin{bmatrix}
  x_{t+1}^i \\ y_{t+1}^i \\ z_{t+1}^i
  \end{bmatrix}
  = \frac{f}{Z''_t} \begin{bmatrix}
  X''_t \\ Y''_t \\ f
  \end{bmatrix} + \begin{bmatrix}
  c_x \\ c_y
  \end{bmatrix}
  \]

  \((U_t(i), V_t(i)) = (x_{t+1}^i - x_i^t, y_{t+1}^i - y_i^t)\)
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, $W_t$, 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

\[
\sum_{x,y} d_{\text{mask}_t}^{\text{GT}}(x, y) \cdot \| d_t(x, y) - d_t^{\text{GT}}(x, y) \|_1
\]

- Sometimes, ground truth motion is known (GPS, IMU)

\[
R_t^{\text{err}} = \text{inv}(R_t^c) R_t^{c-\text{GT}} , \quad t_t^{\text{err}} = \text{inv}(R_t^c)(t_t^{c-\text{GT}} - t_t^c)
\]