Structure from Motion

Computer Vision
CS 143, Brown

James Hays

Many slides adapted from Derek Hoiem, Lana Lazebnik, Silvio Saverese, Steve Seitz, and Martial Hebert
This class: structure from motion

• Recap of epipolar geometry
  – Depth from two views

• Affine structure from motion
Recap: Epipoles

- Point $x$ in left image corresponds to **epipolar line** $l'$ in right image.
- Epipolar line passes through the epipole (the intersection of the cameras’ baseline with the image plane).
Recap: Fundamental Matrix

- Fundamental matrix maps from a point in one image to a line in the other

\[ l' = Fx \quad l = F^\top x' \]

- If \( x \) and \( x' \) correspond to the same 3d point \( X \):

\[ x'^\top Fx = 0 \]
Structure from motion

• Given a set of corresponding points in two or more images, compute the camera parameters and the 3D point coordinates.
Structure from motion ambiguity

• If we scale the entire scene by some factor $k$ and, at the same time, scale the camera matrices by the factor of $1/k$, the projections of the scene points in the image remain exactly the same:

$$ x = PX = \left( \frac{1}{k} P \right) (k X) $$

It is impossible to recover the absolute scale of the scene!
How do we know the scale of image content?
Structure from motion ambiguity

• If we scale the entire scene by some factor \( k \) and, at the same time, scale the camera matrices by the factor of \( 1/k \), the projections of the scene points in the image remain exactly the same.

• More generally: if we transform the scene using a transformation \( Q \) and apply the inverse transformation to the camera matrices, then the images do not change.

\[
x = PX = \left( PQ^{-1} \right)(QX)
\]
Projective structure from motion

• Given: \( m \) images of \( n \) fixed 3D points
  • \( x_{ij} = P_i X_j, \ i = 1, \ldots, m, \ j = 1, \ldots, n \)

• Problem: estimate \( m \) projection matrices \( P_i \) and \( n \) 3D points \( X_j \) from the \( mn \) corresponding points \( x_{ij} \)
Projective structure from motion

• Given: $m$ images of $n$ fixed 3D points
  • $x_{ij} = P_i X_j$, $i = 1, \ldots, m$, $j = 1, \ldots, n$

• Problem: estimate $m$ projection matrices $P_i$ and $n$ 3D points $X_j$ from the $mn$ corresponding points $x_{ij}$

• With no calibration info, cameras and points can only be recovered up to a 4x4 projective transformation $Q$:
  • $X \rightarrow QX$, $P \rightarrow PQ^{-1}$

• We can solve for structure and motion when
  • $2mn \geq 11m + 3n - 15$

• For two cameras, at least 7 points are needed
Types of ambiguity

- **Projective**
  - 15dof
  - \[
  \begin{bmatrix}
  A & t \\
  v^T & v
  \end{bmatrix}
  \]
  - Preserves intersection and tangency

- **Affine**
  - 12dof
  - \[
  \begin{bmatrix}
  A & t \\
  0^T & 1
  \end{bmatrix}
  \]
  - Preserves parallelism, volume ratios

- **Similarity**
  - 7dof
  - \[
  \begin{bmatrix}
  sR & t \\
  0^T & 1
  \end{bmatrix}
  \]
  - Preserves angles, ratios of length

- **Euclidean**
  - 6dof
  - \[
  \begin{bmatrix}
  R & t \\
  0^T & 1
  \end{bmatrix}
  \]
  - Preserves angles, lengths

- With no constraints on the camera calibration matrix or on the scene, we get a *projective* reconstruction
- Need additional information to *upgrade* the reconstruction to affine, similarity, or Euclidean
Projective ambiguity

\[ x = PX = \left( PQ_P^{-1} \right) \left( Q_P X \right) \]
Projective ambiguity
Affine ambiguity

\[ X = PX = (PQ_A^{-1})(Q_A X) \]

\[ Q_A = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} \]
Affine ambiguity
Similarity ambiguity

\[ Q_s = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \]

\[ x = PX = \left( PQ_s^{-1} \right) (Q_s X) \]
Similarity ambiguity
Bundle adjustment

- Non-linear method for refining structure and motion
- Minimizing reprojection error

\[ E(P, X) = \sum_{i=1}^{m} \sum_{j=1}^{n} D(x_{ij}, P_i X_j)^2 \]
Photo synth


http://photosynth.net/
Structure from motion

- Let’s start with *affine cameras* (the math is easier)
Affine structure from motion

- Affine projection is a linear mapping + translation in inhomogeneous coordinates

\[ \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \mathbf{A} \mathbf{X} + \mathbf{t} \]

1. We are given corresponding 2D points (\( \mathbf{x} \)) in several frames
2. We want to estimate the 3D points (\( \mathbf{X} \)) and the affine parameters of each camera (\( \mathbf{A} \))
Affine structure from motion

• Centering: subtract the centroid of the image points

\[
\hat{x}_{ij} = x_{ij} - \frac{1}{n} \sum_{k=1}^{n} x_{ik} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i - \frac{1}{n} \sum_{k=1}^{n} (\mathbf{A}_i \mathbf{X}_k + \mathbf{b}_i)
\]

\[
= \mathbf{A}_i \left( \mathbf{X}_j - \frac{1}{n} \sum_{k=1}^{n} \mathbf{X}_k \right) = \mathbf{A}_i \hat{\mathbf{X}}_j
\]

• For simplicity, assume that the origin of the world coordinate system is at the centroid of the 3D points

• After centering, each normalized point \( x_{ij} \) is related to the 3D point \( \mathbf{X}_i \) by

\[
\hat{\mathbf{x}}_{ij} = \mathbf{A}_i \mathbf{X}_j
\]
Suppose we know 3D points and affine camera parameters ... then, we can compute the observed 2d positions of each point.

\[
\begin{bmatrix}
A_1 \\
A_2 \\
\vdots \\
A_m
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
\vdots \\
X_n
\end{bmatrix}
\]

3D Points (3xn)

Camera Parameters (2mx3)
What if we instead observe corresponding 2d image points?

Can we recover the camera parameters and 3d points?

\[
\begin{bmatrix}
\hat{x}_{11} & \hat{x}_{12} & \cdots & \hat{x}_{1n} \\
\hat{x}_{21} & \hat{x}_{22} & \cdots & \hat{x}_{2n} \\
\vdots & \ddots & \ddots & \vdots \\
\hat{x}_{m1} & \hat{x}_{m2} & \cdots & \hat{x}_{mn}
\end{bmatrix}
\]

\[
[\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix} [X_1 \ X_2 \ \cdots \ X_n]
\]

What rank is the matrix of 2D points?
Factorizing the measurement matrix

\[ D = AX \]
Factorizing the measurement matrix

- Singular value decomposition of $D$:

$$
\begin{align*}
\begin{array}{cc}
\begin{array}{c}
\text{D} \\
\end{array} & = \\
\begin{array}{c}
\text{U} \\
\end{array}
\end{array}
\times
\begin{array}{c}
\text{W} \\
\end{array}
\times
\begin{array}{c}
\text{V}_T \\
\end{array}
\end{align*}

\begin{array}{c}
\text{D} \\
\end{array} = \\
\begin{array}{c}
\text{U}_3 \\
\end{array}
\times
\begin{array}{c}
\text{W}_3 \\
\end{array}
\times
\begin{array}{c}
\text{V}_3^T \\
\end{array}
\end{align*}
$$

Source: M. Hebert
Factorizing the measurement matrix

- Singular value decomposition of $D$:

\[
D = U W V^T
\]

To reduce to rank 3, we just need to set all the singular values to 0 except for the first 3.
Factorizing the measurement matrix

- Obtaining a factorization from SVD:

\[
2m \quad D = U_3 \times 3 \quad W_3 \times V_3^T
\]
Factorizing the measurement matrix

• Obtaining a factorization from SVD:

\[ D = U_3 W_3^{1/2} S = W_3^{1/2} V_3^T \]

This decomposition minimizes \(|D-MS|^2\)

Source: M. Hebert
Affine ambiguity

- The decomposition is not unique. We get the same $\mathbf{D}$ by using any $3 \times 3$ matrix $\mathbf{C}$ and applying the transformations $\mathbf{A} \rightarrow \mathbf{AC}$, $\mathbf{X} \rightarrow \mathbf{C}^{-1}\mathbf{X}$
- That is because we have only an affine transformation and we have not enforced any Euclidean constraints (like forcing the image axes to be perpendicular, for example)
Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and scale is 1

\[ \mathbf{a}_1 \cdot \mathbf{a}_2 = 0 \]
\[ |\mathbf{a}_1|^2 = |\mathbf{a}_2|^2 = 1 \]

- This translates into \( 3m \) equations in \( \mathbf{L} = \mathbf{CC}^T \):

\[ \mathbf{A}_i \mathbf{L} \mathbf{A}_i^T = \mathbf{Id}, \quad i = 1, \ldots, m \]

- Solve for \( \mathbf{L} \)
- Recover \( \mathbf{C} \) from \( \mathbf{L} \) by Cholesky decomposition: \( \mathbf{L} = \mathbf{CC}^T \)
- Update \( \mathbf{M} \) and \( \mathbf{S} \): \( \mathbf{M} = \mathbf{MC} \), \( \mathbf{S} = \mathbf{C}^{-1} \mathbf{S} \)

Source: M. Hebert
Algorithm summary

- Given: \( m \) images and \( n \) tracked features \( x_{ij} \)
- For each image \( i \), center the feature coordinates
- Construct a \( 2m \times n \) measurement matrix \( D \):
  - Column \( j \) contains the projection of point \( j \) in all views
  - Row \( i \) contains one coordinate of the projections of all the \( n \) points in image \( i \)
- Factorize \( D \):
  - Compute SVD: \( D = U W V^T \)
  - Create \( U_3 \) by taking the first 3 columns of \( U \)
  - Create \( V_3 \) by taking the first 3 columns of \( V \)
  - Create \( W_3 \) by taking the upper left \( 3 \times 3 \) block of \( W \)
- Create the motion (affine) and shape (3D) matrices:
  \[ A = U_3 W_3^{\frac{1}{2}} \text{ and } X = W_3^{\frac{1}{2}} V_3^T \]
- Eliminate affine ambiguity

Source: M. Hebert
Dealing with missing data

• So far, we have assumed that all points are visible in all views

• In reality, the measurement matrix typically looks something like this:

One solution:

– solve using a dense submatrix of visible points
– Iteratively add new cameras
A nice short explanation

- Class notes from Lischinksi and Gruber