Structure from Motion

Jia-Bin Huang, Virginia Tech

Many slides from S. Seitz, N Snavely, and D. Hoiem
Administrative stuffs

• HW 3 due 11:59 PM, Oct 25 (Wed)

• Submit your alignment results! [Link]

• HW 2 grades are out
  • Average: 113.7, median: 126
  • Excellent reports: Diya Li, Vikram Mohanty, Yihan Pang, Chang Sun, Yecheng Zhao
Perspective and 3D Geometry

- **Projective geometry and camera models**
  - Vanishing points/lines
  - \( \mathbf{x} = \mathbf{K}[\mathbf{R} \; \mathbf{t}]\mathbf{X} \)

- **Single-view metrology and camera calibration**
  - Calibration using known 3D object or vanishing points
  - Measuring size using perspective cues

- **Photo stitching**
  - Homography relates rotating cameras \( \mathbf{x}' = \mathbf{H}\mathbf{x} \)
  - Recover homography using RANSAC + normalized DLT

- **Epipolar Geometry and Stereo Vision**
  - Fundamental/essential matrix relates two cameras \( \mathbf{x}'\mathbf{F}\mathbf{x} = \mathbf{0} \)
  - Recover \( \mathbf{F} \) using RANSAC + normalized 8-point algorithm, enforce rank 2 using SVD

- **Structure from motion** (this class)
  - How can we recover 3D points from multiple images?
Recap: Epipoles

- Point $x$ in the left image corresponds to \textit{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 \( X \):

\[ x'^\top Fx = 0 \]
Recap: Automatic Estimation of F

Assume we have matched points \( x \leftrightarrow x' \) with outliers

**8-Point Algorithm for Recovering F**

- Correspondence Relation
  \[ x'^T F x = 0 \]

1. Normalize image coordinates
   \[ \tilde{x} = T x \quad \tilde{x}' = T' x' \]

2. RANSAC with 8 points
   - Randomly sample 8 points
   - Compute \( F \) via least squares
   - Enforce \( \det(\tilde{F}) = 0 \) by SVD
   - Repeat and choose \( F \) with most inliers

3. De-normalize:
   \[ F = T'^T \tilde{F} T \]
This class: Structure from Motion

• Projective structure from motion

• Affine structure from motion

• HW 3
  • Fundamental matrix
  • Affine structure from motion

• Multi-view stereo (optional)
Structure ['strək(t)SHər]:
3D Point Cloud of the Scene

Motion ['mōSH(ə)n]:
Camera Location and Orientation

Structure from Motion (SfM)
Get the Point Cloud from Moving Cameras
SfM Applications – 3D Modeling
SfM Applications – Surveying cultural heritage structure analysis

Guidi et al. High-accuracy 3D modeling of cultural heritage, 2004
SfM Applications – Robot navigation and mapmaking

https://www.youtube.com/watch?v=1HhOmF22oYA
SfM Applications – Visual effect (matchmove)

https://www.youtube.com/watch?v=bK6vCPCFkfk
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

$+$

Images $\rightarrow$ Models: Image-based Modeling

Slide credit: J. Xiao
<table>
<thead>
<tr>
<th>Step</th>
<th>Method</th>
<th>Step</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images → Points:</td>
<td>Structure from Motion</td>
<td>Points → More points:</td>
<td>Multiple View Stereo</td>
</tr>
<tr>
<td>Points → Meshes:</td>
<td>Model Fitting</td>
<td>Meshes → Models:</td>
<td>Texture Mapping</td>
</tr>
<tr>
<td>Images → Models:</td>
<td>Image-based Modeling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

$+$

Images $\rightarrow$ Models: Image-based Modeling

Slide credit: J. Xiao
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

$+$

= Images $\rightarrow$ Models: Image-based Modeling
Steps

Images $\rightarrow$ Points: Structure from Motion
Points $\rightarrow$ More points: Multiple View Stereo
Points $\rightarrow$ Meshes: Model Fitting
Meshes $\rightarrow$ Models: Texture Mapping

Images $\rightarrow$ Models: Image-based Modeling
Steps

Images $\rightarrow$ Points: Structure from Motion
Points $\rightarrow$ More points: Multiple View Stereo
Points $\rightarrow$ Meshes: Model Fitting
Meshes $\rightarrow$ Models: Texture Mapping

+$=$

Images $\rightarrow$ Models: Image-based Modeling

Example: https://photosynth.net/
Triangulation: Linear Solution

• Generally, rays $C \rightarrow x$ and $C' \rightarrow x'$ will not exactly intersect

• Solve via SVD:
  A least squares solution to a system of equations

$$AX = 0 \quad A = \begin{bmatrix}
up_3^T - p_1^T \\
v p_3^T - p_2^T \\
u'p_3^{rT} - p_1^{rT} \\
v'p_3^{rT} - p_2^{rT}
\end{bmatrix}$$

Further reading: HZ p. 312-313
Triangulation: Linear Solution

\[ x = w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} \]

\[ x' = w \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \quad \mathbf{P}' = \begin{bmatrix} p'_1^T \\ p'_2^T \\ p'_3^T \end{bmatrix} \]

\[ x = \mathbf{P}X \quad x' = \mathbf{P}'X \]

\[ w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{u}p_3^TX = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix}X = \begin{bmatrix} p_1^TX \\ p_2^TX \\ p_3^TX \end{bmatrix} \]

\[ u'p'_3X - p'_1X = [u'p'_3 - p'_1]X = 0 \]
\[ v'p'_3X - p'_2X = [v'p'_3 - p'_2]X = 0 \]
\[ u'p'_3X - p'_1X = [u'p'_3 - p'_1]X = 0 \]
\[ v'p'_3X - p'_2X = [v'p'_3 - p'_2]X = 0 \]
Triangulation: Linear Solution

Given $P, P', x, x'$

1. Precondition points and projection matrices
2. Create matrix $A$
3. $[U, S, V] = \text{svd}(A)$
4. $X = V(:, \text{end})$

Pros and Cons
- Works for any number of corresponding images
- Not projectively invariant

$$
\begin{align*}
\mathbf{x} &= \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \\
\mathbf{x}' &= \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}
\end{align*}
$$

$$
\begin{align*}
P &= \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} \\
P' &= \begin{bmatrix} p_1'^T \\ p_2'^T \\ p_3'^T \end{bmatrix}
\end{align*}
$$

$$
\begin{align*}
A &= \begin{bmatrix} up_3^T - p_1^T \\ vp_3^T - p_2^T \\ up_1'^T - p_1'^T \\ vp_1'^T - p_2'^T \end{bmatrix}
\end{align*}
$$

Code: [http://www.robots.ox.ac.uk/~vgg/hzbook/code/vgg_multiview/vgg_X_from_xP_lin.m](http://www.robots.ox.ac.uk/~vgg/hzbook/code/vgg_multiview/vgg_X_from_xP_lin.m)
Triangulation: Non-linear Solution

• Minimize projected error while satisfying

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

\[
\text{cost}(X) = \text{dist}(x, \hat{x})^2 + \text{dist}(x', \hat{x}')^2
\]

Figure source: Robertson and Cipolla (Chpt 13 of Practical Image Processing and Computer Vision)
Triangulation: Non-linear Solution

• Minimize projected error while satisfying

$$\hat{x}'^T F \hat{x} = 0$$

$$cost(X) = \text{dist}(x, \hat{x})^2 + \text{dist}(x', \hat{x}')^2$$

• Solution is a 6-degree polynomial of $t$, minimizing

$$d(x, l(t))^2 + d(x', l'(t))^2$$

Further reading: HZ p. 318
Projective structure from motion

• Given: $m$ images of $n$ fixed 3D points

$$x_{ij} = P_i X_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

• Problem: estimate $m$ projection matrices $P_i$ and $n$ 3D points $X_j$ from the $mn$ corresponding 2D points $x_{ij}$
Projective structure from motion

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

• Problem:
  
  • Estimate unknown $m$ projection matrices $P_i$ and $n$ 3D points $X_j$ from the known $mn$ corresponding points $x_{ij}$
  
  • With no calibration info, cameras and points can only be recovered up to a 4x4 projective transformation $Q$:
    
    \[ X \rightarrow QX, \quad P \rightarrow PQ^{-1} \]

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

  - Up to 4x4 projective tform $Q$

• For two cameras, at least 7 points are needed
Sequential structure from motion

• Initialize motion (calibration) from two images using fundamental matrix

• Initialize structure by triangulation

• For each additional view:
  • Determine projection matrix of new camera using all the known 3D points that are visible in its image – calibration/resectioning
Sequential structure from motion

- Initialize motion from two images using fundamental matrix

- Initialize structure by triangulation

- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
  
  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*
Sequential structure from motion

• Initialize motion from two images using fundamental matrix

• Initialize structure by triangulation

• For each additional view:
  • Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*

  • Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*

• Refine structure and motion: bundle adjustment
Bundle adjustment

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

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

- Theory: The Levenberg–Marquardt algorithm
- Practice: The Ceres-Solver from Google
Auto-calibration

• Auto-calibration: determining intrinsic camera parameters directly from uncalibrated images

• For example, we can use the constraint that a moving camera has a fixed intrinsic matrix
  • Compute initial projective reconstruction and find 3D projective transformation matrix $Q$ such that all camera matrices are in the form $P_i = K [R_i | t_i]$

• Can use constraints on the form of the calibration matrix, such as zero skew
Summary so far

• From two images, we can:
  • Recover fundamental matrix $F$
  • Recover canonical camera projection matrix $P$ and $P'$ from $F$
  • Estimate 3D positions (if $K$ is known) that correspond to each pixel

• For a moving camera, we can:
  • Initialize by computing $F$, $P$, $X$ for two images
  • Sequentially add new images, computing new $P$, refining $X$, and adding points
  • Auto-calibrate assuming fixed calibration matrix to upgrade to similarity transform
Recent work in SfM

• Reconstruct from many images by efficiently finding subgraphs

• Improving efficiency of bundle adjustment or

(best method with software available; also has good overview of recent methods)

[Reconstruction of Cornell](http://vision.soic.indiana.edu/projects/disco/) (Crandall et al. ECCV 2011)
3D from multiple images

Building Rome in a Day: Agarwal et al. 2009
Structure from motion under orthographic projection

3D Reconstruction of a Rotating Ping-Pong Ball

• Reasonable choice when
  • Change in depth of points in scene is much smaller than distance to camera
  • Cameras do not move towards or away from the scene

Orthographic Projection - Examples
Orthographic projection for rotated/translated camera

\[
\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} R'_{f} \\ X_{p} \\ Y_{p} \\ Z_{p} \end{bmatrix} + t_{f}
\]

\[
R_{f} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R'_{f} \quad \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = R_{f} \begin{bmatrix} X_{p} \\ Y_{p} \\ Z_{p} \end{bmatrix} + t_{f}
\]
Affine structure from motion

• Affine projection is a linear mapping + translation in homogeneous coordinates

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

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 (\(\mathbf{A}\))
Step 1: Simplify by getting rid of $t$: shift to centroid of points for each camera

\[ x_i = A_i X + t_i \]
\[ \hat{x}_{ij} = x_{ij} - \frac{1}{n} \sum_{k=1}^{n} x_{ik} \]

\[ x_{ij} - \frac{1}{n} \sum_{k=1}^{n} x_{ik} = A_i X \_ j + t_i - \frac{1}{n} \sum_{k=1}^{n} (A_i X \_ k + t_i) = A_i \left( X \_ j - \frac{1}{n} \sum_{k=1}^{n} X \_ k \right) = A_i \hat{X} \_ j \]

\[ \hat{x}_{ij} = A_i \hat{X} \_ j \]

2d normalized point (observed)

3d normalized point

Linear (affine) mapping
Suppose we know 3D points and affine camera parameters...

then, we can compute the observed 2d positions of each point

$$\begin{bmatrix}
A_1 \\
A_2 \\
\vdots \\
A_m
\end{bmatrix}
\begin{bmatrix}
X_1 & X_2 & \cdots & X_n
\end{bmatrix}$$

3D Points (3xn)

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

Can we recover the camera parameters and 3d points?

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

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

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

\[ \text{Measurements} = \text{Motion} \times \text{Shape} \]

\[ D = AX \]
Factorizing the measurement matrix

- Singular value decomposition of D:

$$\begin{align*}
    D &= U \times W \times V^T \\
    D &= U_3 \times W_3 \times V_3^T
\end{align*}$$

Source: M. Hebert
Factorizing the measurement matrix

- Singular value decomposition of D:

\[ D = U W V_T^T \]

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:

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

- Obtaining a factorization from SVD:

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

Possible decomposition:

\[ M = U_3 W_3^{1/2} \quad S = W_3^{1/2} V_3^T \]
**Affine ambiguity**

- The decomposition is not unique. We get the same $D$ by using any $3 \times 3$ matrix $C$ and applying the transformations $A \rightarrow AC$, $X \rightarrow C^{-1}X$.

- Why? We have only an affine transformation and we have not enforced any Euclidean constraints (e.g., perpendicular image axes).

Source: M. Hebert
Eliminating the affine ambiguity

• Orthographic: image axes are perpendicular and of unit length

\[ \mathbf{a}_1 \cdot \mathbf{a}_2 = 0 \]

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

Source: M. Hebert
Solve for orthographic constraints

Three equations for each image i

\[ \tilde{\mathbf{a}}_{i1}^T \mathbf{C} \mathbf{C}^T \tilde{\mathbf{a}}_{i1} = 1 \]
\[ \tilde{\mathbf{a}}_{i2}^T \mathbf{C} \mathbf{C}^T \tilde{\mathbf{a}}_{i2} = 1 \]
\[ \tilde{\mathbf{a}}_{i1}^T \mathbf{C} \mathbf{C}^T \tilde{\mathbf{a}}_{i2} = 0 \]

- Solve for \( \mathbf{L} = \mathbf{C} \mathbf{C}^T \)
- Recover \( \mathbf{C} \) from \( \mathbf{L} \) by Cholesky decomposition: \( \mathbf{L} = \mathbf{C} \mathbf{C}^T \)
- Update \( \mathbf{A} \) and \( \mathbf{X} \): \( \mathbf{A} = \tilde{\mathbf{A}} \mathbf{C} \), \( \mathbf{X} = \mathbf{C}^{-1} \tilde{\mathbf{X}} \)
How to solve $\mathbf{L} = \mathbf{CC}^T$?

\[
\begin{bmatrix}
a & b & c \\
L_{11} & L_{21} & L_{31} \\
L_{12} & L_{22} & L_{32} \\
L_{13} & L_{23} & L_{33} \\
\end{bmatrix}
\begin{bmatrix}
d \\
e \\
f \\
\end{bmatrix} = k
\]
How to solve $L = CC^T$?

$$\begin{bmatrix} a & b & c \\ L_{12} & L_{22} & L_{32} \\ L_{13} & L_{23} & L_{33} \end{bmatrix} \begin{bmatrix} L_{11} & L_{21} & L_{31} \\ d \\ e \\ f \end{bmatrix} = k$$

Reshape: $\text{reshape}([a b c]'*[d e f], [1, 9])$
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:
  \[ A = U_3 W_3^{\frac{1}{2}} \text{ and } S = W_3^{\frac{1}{2}} V_3^T \]
- Eliminate affine ambiguity
  - Solve \( L = CC^T \) using metric constraints
  - Solve \( C \) using Cholesky decomposition
  - Update \( A \) and \( X \): \( A = AC, S = C^{-1}S \)

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
Further reading

• Short explanation of Affine SfM: class notes from Lischinksii and Gruber

• Clear explanation of epipolar geometry and projective SfM
Review of Affine SfM from Interest Points

1. Detect interest points (e.g., Harris)

\[
\mu(\sigma_I, \sigma_D) = g(\sigma_I)^* \begin{bmatrix}
    I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\
    I_x I_y(\sigma_D) & I_y^2(\sigma_D)
\end{bmatrix}
\]

1. Image derivatives

\[\det M = \lambda_1 \lambda_2\]
\[\text{trace } M = \lambda_1 + \lambda_2\]

2. Square of derivatives

3. Gaussian filter \(g(\sigma_I)\)

4. Cornerness function – both eigenvalues are strong

\[\text{har} = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] =
\]
\[g(I_x^2) g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2\]

5. Non-maxima suppression
Review of Affine SfM from Interest Points

2. Correspondence via Lucas-Kanade tracking

a) Initialize \((x', y') = (x, y)\)

b) Compute \((u, v)\) by

\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} = -
\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}
\]

- 2\textsuperscript{nd} moment matrix for feature patch in first image
- displacement

\[
I_t = I(x', y', t+1) - I(x, y, t)
\]

Original \((x, y)\) position

\[
x' = x' + u; \quad y' = y' + v;
\]

c) Shift window by \((u, v)\)
d) Recalculate \(I_t\)
e) Repeat steps 2-4 until small change
   - Use interpolation for subpixel values
Review of Affine SfM from Interest Points

3. Get Affine camera matrix and 3D points using

\[
\begin{align*}
\mathbf{D} & = \mathbf{U} \\
\mathbf{W} & \times \\
\mathbf{V}^T & \times \\
\mathbf{U}_3 & = \mathbf{W}_3 \\
\mathbf{V}_3^T & \\
\end{align*}
\]

Solve for orthographic constraints
Problem: recover F from matches with outliers

load matches.mat

\([c1, r1] - 477 \times 2\)
\([c2, r2] - 500 \times 2\)
\(\text{matches} - 252 \times 2\)

matches(:,1): matched point in im1
matches(:,2): matched point in im2

Write-up:

- Describe what test you used for deciding inlier vs. outlier.
- Display the estimated fundamental matrix \(F\) after normalizing to unit length
- Plot the outlier keypoints with green dots on top of the first image `plot(x, y, '.g');`
- Plot the corresponding epipolar lines

8-Point Algorithm for Recovering \(F\)

- Correspondence Relation
  \[ x'\top F x = 0 \]

1. Normalize image coordinates
   \[ \tilde{x} = T x \quad \tilde{x}' = T' x' \]

2. RANSAC with 8 points
   - Randomly sample 8 points
   - Compute \(F\) via least squares
   - Enforce \(\det(\tilde{F}) = 0\) by SVD
   - Repeat and choose \(F\) with most inliers

3. De-normalize: \(F = T'\top \tilde{F} T\)
Distance of point to epipolar line

\[ l = Fx = [a \ b \ c] \]

\[ x' = [u \ v \ 1] \]

\[ d(l, x') = \frac{|au + bv + c|}{\sqrt{a^2 + b^2}} \]
HW 3 – Part 2 Affine SfM

Problem: recover motion and structure

Affine Structure from motion

- 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:
  $A = U_3 W_3^{\frac{1}{2}}$ and $S = W_3^{\frac{1}{2}} V_3^T$
- Eliminate affine ambiguity
  - Solve $L = CC^T$ using metric constraints
  - Solve $C$ using Cholesky decomposition
  - Update $A$ and $X$: $A = AC$, $S = C^{-1}S$

load tracks.mat

track_x - [500 x 51]
track_y - [500 x 51]

Use plotSfM($A$, $S$) to display motion and shape

$A$ - [2m x 3] motion matrix
$S$ - [3 x n]
HW 3 – Part 2 Affine SfM

• Eliminate affine ambiguity

\[ \tilde{a}_{i1}^T CC^T \tilde{a}_{i1} = 1 \]
\[ \tilde{a}_{i2}^T CC^T \tilde{a}_{i2} = 1 \] where \[ \tilde{A}_i = \begin{bmatrix} \tilde{a}_{i1}^T \\ \tilde{a}_{i2}^T \end{bmatrix} \]
\[ \tilde{a}_{i1}^T CC^T \tilde{a}_{i2} = 0 \]

• Solve for \( L = CC^T \)
  • \( L = \text{reshape}(A \backslash b, [3,3]); \) \% A - 3m x 9, b – 3m x 1

• Recover \( C \) from \( L \) by Cholesky decomposition: \( L = CC^T \)

• Update \( A \) and \( X \): \( A = AC, X = C^{-1}X \)
Assume Sign = 1.65m

Question: What’s the heights of

- Building
- Tractor
- Camera
HW 3 – Graduate credits
Automatic vanishing point detection

Input:
• lines: a matrix of size [NumLines x 5] where each row represents a line segment with (x1, y1, x2, y2, lineLength)

Output:
• VP: [2 x 3] each column corresponds to a vanishing point in the order of X, Y, Z
• lineLabel: [NumLine x 3] each column is a logical vector indicating which line segments correspond to the vanishing point.
Try “un-normalized” 8-point algorithm.

Report and compare the accuracy with the normalized version.
HW 3 – Graduate credits  
**Affine structure from motion**

• Missing track completion.

• Some keypoints will fall out of frame, or come into frame throughout the sequence.

• Fill in the missing data and visualize the predicted positions of points that aren't visible in a particular frame.
Multi-view stereo
Multi-view stereo

• Generic problem formulation: given several images of the same object or scene, compute a representation of its 3D shape

• “Images of the same object or scene”
  • Arbitrary number of images (from two to thousands)
  • Arbitrary camera positions (special rig, camera network or video sequence)
  • Calibration may be known or unknown

• “Representation of 3D shape”
  • Depth maps
  • Meshes
  • Point clouds
  • Patch clouds
  • Volumetric models
  • ....
Multi-view stereo: Basic idea

Source: Y. Furukawa
Multi-view stereo: Basic idea

Source: Y. Furukawa
Multi-view stereo: Basic idea

Source: Y. Furukawa
Multi-view stereo: Basic idea

Source: Y. Furukawa
Plane Sweep Stereo

- Sweep family of planes at different depths w.r.t. a reference camera
- For each depth, project each input image onto that plane
- This is equivalent to a homography warping each input image into the reference view
- What can we say about the scene points that are at the right depth?

Plane Sweep Stereo

- For each depth plane
  - For each pixel in the composite image stack, compute the variance
- For each pixel, select the depth that gives the lowest variance

- Can be accelerated using graphics hardware

Merging depth maps

• Given a group of images, choose each one as reference and compute a depth map w.r.t. that view using a multi-baseline approach

• Merge multiple depth maps to a volume or a mesh (see, e.g., Curless and Levoy 96)
Stereo from community photo collections

• Need *structure from motion* to recover unknown camera parameters
• Need *view selection* to find good groups of images on which to run dense stereo
Local view selection

- Automatically select neighboring views for each point in the image
- Desiderata: good matches AND good baselines
Local view selection

- Automatically select neighboring views for each **point** in the image
- DeSiderate: good matches **AND** good baselines
Local view selection

- Automatically select neighboring views for each **point** in the image
- Desiderata: good matches AND good baselines
Towards Internet-Scale Multi-View Stereo

• **YouTube video, high-quality video**

The Visual Turing Test for Scene Reconstruction

The Reading List

• “A computer algorithm for reconstructing a scene from two images”, Longuet-Higgins, Nature 1981

• “Shape and motion from image streams under orthography: A factorization method.” C. Tomasi and T. Kanade, IJCV, 9(2):137-154, November 1992

• “In defense of the eight-point algorithm”, Hartley, PAMI 1997

• “An efficient solution to the five-point relative pose problem”, Nister, PAMI 2004

• “Accurate, dense, and robust multiview stereopsis”, Furukawa and Ponce, CVPR 2007

• “Photo tourism: exploring image collections in 3d”, ACM SIGGRAPH 2006

• “Building Rome in a day”, Agarwal et al., ICCV 2009

• https://www.youtube.com/watch?v=kylzMr917Rc, 3D Computer Vision: Past, Present, and Future
Next class

• Grouping and Segmentation