CS4501: Introduction to Computer Vision

Epipolar Geometry: Essential Matrix

Various slides from previous courses by:
D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL), S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook). J. M. Frahm (UNC), V. Ordonez (UVA), Steve Seitz (UW).
Last Class

• Stereo Vision – Dense Stereo
• More on Epipolar Geometry
Today’s Class

• More on Epipolar Geometry
  • Essential Matrix
  • Fundamental Matrix
Estimating depth with stereo

• **Stereo**: shape from “motion” between two views
• We’ll need to consider:
  • Info on camera pose (“calibration”)
  • Image point correspondences
Key idea: Epipolar constraint

Potential matches for $x$ have to lie on the corresponding line $l'$.

Potential matches for $x'$ have to lie on the corresponding line $l$. 
Epipolar geometry: notation

- **Baseline** – line connecting the two camera centers
- **Epipoles**
  - intersections of baseline with image planes
  - projections of the other camera center
- **Epipolar Plane** – plane containing baseline (1D family)
Epipolar geometry: notation

- **Baseline** – line connecting the two camera centers
- **Epipoles**
  - intersections of baseline with image planes
  - projections of the other camera center
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)
Epipolar Geometry: Another example

Credit: William Hoff, Colorado School of Mines
Example: Converging cameras
Geometry for a simple stereo system

• First, assuming parallel optical axes, known camera parameters (i.e., calibrated cameras):
Simplest Case: Parallel images

- Image planes of cameras are parallel to each other and to the baseline
- Camera centers are at same height
- Focal lengths are the same
- Then epipolar lines fall along the horizontal scan lines of the images
Depth from disparity

image \( I(x,y) \)  

Disparity map \( D(x,y) \)  

image \( I'(x',y') \)

\[(x',y') = (x + D(x,y), y)\]

So if we could find the corresponding points in two images, we could estimate relative depth...
Correspondence search

- Slide a window along the right scanline and compare contents of that window with the reference window in the left image
- Matching cost: SSD or normalized correlation
Correspondence search

\[ \text{SSD} = \sum \sum (I_{left} - I_{right})^2 \]
Correspondence search

\[ NC = \frac{\sum \sum (I_{left} \cdot I_{right})}{\sqrt{\sum \sum \sum I_{left} \cdot I_{right}}} \]

Norm. corr
Basic stereo matching algorithm

- If necessary, rectify the two stereo images to transform epipolar lines into scanlines.
- For each pixel $x$ in the first image:
  - Find corresponding epipolar scanline in the right image.
  - Examine all pixels on the scanline and pick the best match $x'$.
  - Compute disparity $x - x'$ and set $\text{depth}(x) = B \cdot f / (x - x')$. 
Failures of correspondence search

- Textureless surfaces
- Occlusions, repetition
- Non-Lambertian surfaces, specularities
Active stereo with structured light

- Project “structured” light patterns onto the object
  - Simplifies the correspondence problem
  - Allows us to use only one camera

Kinect and Iphone X Solution

• Add Texture!
Kinect: Structured infrared light

iPhone X
Basic stereo matching algorithm

• If necessary, rectify the two stereo images to transform epipolar lines into scanlines

• For each pixel $x$ in the first image
  • Find corresponding epipolar scanline in the right image
  • Examine all pixels on the scanline and pick the best match $x'$
  • Compute disparity $x - x'$ and set $\text{depth}(x) = \frac{B\cdot f}{x - x'}$
Effect of window size

- Smaller window
  + More detail
  + More noise

- Larger window
  + Smoother disparity maps
  + Less detail
Results with window search

Data

Window-based matching

Ground truth
Better methods exist...


For the latest and greatest: [http://www.middlebury.edu/stereo/](http://www.middlebury.edu/stereo/)
When cameras are not aligned: Stereo image rectification

- Reproject image planes onto a common plane parallel to the line between optical centers
- Pixel motion is horizontal after this transformation
- Two homographies (3x3 transform), one for each input image reprojection

Rectification example
Back to the General Problem
The optical centers of the two cameras, a point $P$, and the image points $p_0$ and $p_1$ of $P$ all lie in the same plane (epipolar plane).

These vectors are co-planar: $\overrightarrow{C_0p_0}$, $\overrightarrow{C_1p_1}$, $\overrightarrow{C_0C_1}$
Another way to write the fact they are co-planar is

\[ \overrightarrow{C_0p_0} \cdot \left( \overrightarrow{C_0C_1} \times \overrightarrow{C_1p_1} \right) = 0 \]
Back to the General Problem

• Now, instead of treating $p_0$ as a point, treat it as a 3D direction vector:

$$p_0 = \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

We assume "normalized" image coordinates; i.e., effective focal length = 1.

This is defined with respect to the coordinate frame of camera 0.

• Another way to write the fact they are co-planar is

$$\overrightarrow{C_0P_0} \cdot (\overrightarrow{C_0C_1} \times \overrightarrow{C_1P_1}) = 0$$

Credit: William Hoff, Colorado School of Mines
Back to the General Problem

- Now, instead of treating $p_0$ as a point, treat it as a 3D direction vector. We assume "normalized" image coordinates; i.e., effective focal length = 1.

\[
p_0 = \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \quad \text{This is defined with respect to the coordinate frame of camera 0}
\]

- $p_1$ is also a direction vector.

\[
p_1 = \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \quad \text{This is defined with respect to the coordinate frame of camera 1}
\]

- Another way to write the fact they are co-planar is

\[
\overrightarrow{c_0p_0} \cdot (\overrightarrow{c_0c_1} \times \overrightarrow{c_1p_1}) = 0
\]
Back to the General Problem

- Another way to write the fact they are co-planar is
  \[ \overrightarrow{C_0P_0} \cdot (\overrightarrow{C_0C_1} \times \overrightarrow{C_1P_1}) = 0 \]

- Now, instead of treating \( \mathbf{p}_0 \) as a point, treat it as a 3D direction vector*
  \[ \mathbf{p}_0 = \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \]
  This is defined with respect to the coordinate frame of camera 0

- \( \mathbf{p}_1 \) is also a direction vector
  \[ \mathbf{p}_1 = \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \]
  This is defined with respect to the coordinate frame of camera 1

- The direction of \( \mathbf{p}_1 \) in camera 0 coordinates is
  \[ \frac{C_0}{C_1} R \mathbf{p}_1 \]
  Namely, we apply the rotation matrix from the camera 1 to camera 0 pose

*A direction vector is a vector whose starting point (tail) doesn’t matter, just its direction.
Back to the General Problem

\[ \mathbf{C_0p_0} \cdot (\mathbf{C_0C_1} \times \mathbf{C_1p_1}) = 0 \]

So we can write the coplanar constraint as

\[ \mathbf{p_0} \cdot (\mathbf{t} \times \mathbf{Rp_1}) = 0 \]
Brief Digression: Cross Product as Matrix Multiplication

• Dot Product as matrix multiplication: Easy
  \[ [a_1 \ a_2 \ a_3] \cdot [b_1 \ b_2 \ b_3] = a_1 b_1 + a_2 b_2 + a_3 b_3 \]
  \[ [a_1 \ a_2 \ a_3][b_1 \ b_2 \ b_3]^T = a_1 b_1 + a_2 b_2 + a_3 b_3 \]

• Cross Product as matrix multiplication:

\[
\begin{pmatrix}
  a_x \\
  a_y \\
  a_z
\end{pmatrix} \times
\begin{pmatrix}
  b_x \\
  b_y \\
  b_z
\end{pmatrix} =
\begin{pmatrix}
  a_y b_z - b_y a_z \\
  a_z b_x - b_z a_x \\
  a_x b_y - b_x a_y
\end{pmatrix}
\]

\[
a \times b = [a] \times b =
\begin{bmatrix}
  0 & -a_3 & a_2 \\
  a_3 & 0 & -a_1 \\
  -a_2 & a_1 & 0
\end{bmatrix}
\begin{pmatrix}
  b_1 \\
  b_2 \\
  b_3
\end{pmatrix}
\]
Back to the General Problem

So we can write the coplanar constraint as

\[ \mathbf{p}_0 \cdot \left( \mathbf{t} \times \mathbf{R}\mathbf{p}_1 \right) = 0 \]

Or

\[ \mathbf{p}_0^T \begin{bmatrix} \mathbf{t} \end{bmatrix}_x \mathbf{R}\mathbf{p}_1 = 0 \]
The Essential Matrix

\[ \mathbf{p}_0^T [t] \times \mathbf{R} \mathbf{p}_1 = 0 \]

Let \( \mathbf{E} = [t] \times \mathbf{R} \) (which is a 3x3 matrix)

\[ \mathbf{p}_0^T \mathbf{E} \mathbf{p}_1 = 0 \]

\[
\begin{pmatrix}
    x_0 & y_0 & 1
\end{pmatrix}
\begin{pmatrix}
    E_{11} & E_{12} & E_{13} \\
    E_{21} & E_{22} & E_{23} \\
    E_{31} & E_{32} & E_{33}
\end{pmatrix}
\begin{pmatrix}
    x_1 \\
    y_1 \\
    1
\end{pmatrix} = 0
\]
The Essential Matrix

Is the matrix $E$, that relates the image of a point in one camera to its image in the other camera, given a translation and rotation

$$p_0^T E p_1 = 0$$

where

$$E = [t]_x R$$

**Essential Matrix**

(Longuet-Higgins, 1981)
Epipolar constraint: Calibrated case

- Intrinsic and extrinsic parameters of the cameras are known, world coordinate system is set to that of the first camera.
- Then the projection matrices are given by $K[I \mid 0]$ and $K'[R \mid t]$.
- We can multiply the projection matrices (and the image points) by the inverse of the calibration matrices to get normalized image coordinates:
  
  \[ x_{\text{norm}} = K^{-1} x_{\text{pixel}} = [I \ 0] X, \quad x'_{\text{norm}} = K'^{-1} x'_{\text{pixel}} = [R \ t] X \]
Epipolar constraint: Calibrated case

The vectors $Rx$, $t$, and $x'$ are coplanar
Epipolar constraint: Calibrated case

\[ x' \cdot [t \times (Rx)] = 0 \quad \Rightarrow \quad x'^T [t_x] Rx = 0 \]

Recall: \( \mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a} \times] \mathbf{b} \)

The vectors \( Rx, t, \) and \( x' \) are coplanar
Epipolar constraint: Calibrated case

The vectors $Rx$, $t$, and $x'$ are coplanar

\[ x' \cdot [t \times (Rx)] = 0 \rightarrow x'^T [t_x] Rx = 0 \rightarrow x'^T E x = 0 \]

Essential Matrix
(Longuet-Higgins, 1981)
Epipolar constraint: Calibrated case

- \( E \mathbf{x} \) is the epipolar line associated with \( \mathbf{x} \) (\( l' = E \mathbf{x} \))
- Recall: a line is given by \( ax + by + c = 0 \) or
  \[
  \mathbf{l}^T \mathbf{x} = 0 \quad \text{where} \quad \mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
  \]
Epipolar constraint: Calibrated case

- $Ex$ is the epipolar line associated with $x$ ($l' = Ex$)
- $E^T x'$ is the epipolar line associated with $x'$ ($l = E^T x'$)
- $Ee = 0$ and $E^T e' = 0$
- $E$ is singular (rank two)
- $E$ has five degrees of freedom

$$x'^T Ex = 0$$
Epipolar constraint: Uncalibrated case

• The calibration matrices $K$ and $K'$ of the two cameras are unknown

• We can write the epipolar constraint in terms of unknown normalized coordinates:

$$\hat{x}'^T E \hat{x} = 0 \quad \hat{x} = K^{-1} x, \quad \hat{x}' = K'^{-1} x'$$
Epipolar constraint: Uncalibrated case

\[ \hat{x}'^T E \hat{x} = 0 \quad \Rightarrow \quad x'^T F x = 0 \quad \text{with} \quad F = K'^{-T} E K^{-1} \]

\[ \hat{x} = K^{-1} x \]

\[ \hat{x}' = K'^{-1} x' \]

**Fundamental Matrix**
(Faugeras and Luong, 1992)
Epipolar constraint: Uncalibrated case

\[ \hat{x}'^T E \hat{x} = 0 \quad \Rightarrow \quad x'^T F x = 0 \quad \text{with} \quad F = K'^{-T} E K^{-1} \]

- \( F x \) is the epipolar line associated with \( x \) (\( l' = F x \))
- \( F^T x' \) is the epipolar line associated with \( x' \) (\( l = F^T x' \))
- \( F e = 0 \) and \( F^T e' = 0 \)
- \( F \) is singular (rank two)
- \( F \) has seven degrees of freedom
Estimating the Fundamental Matrix

• 8-point algorithm
  • Least squares solution using SVD on equations from 8 pairs of correspondences
  • Enforce $\det(F)=0$ constraint using SVD on $F$

• 7-point algorithm
  • Use least squares to solve for null space (two vectors) using SVD and 7 pairs of correspondences
  • Solve for linear combination of null space vectors that satisfies $\det(F)=0$

• Minimize reprojection error
  • Non-linear least squares

Note: estimation of $F$ (or $E$) is degenerate for a planar scene.
8-point algorithm

1. Solve a system of homogeneous linear equations
   a. Write down the system of equations
      \[ x^T F x' = 0 \]
      \[ uu'f_{11} + uv'f_{12} + uf_{13} + vu'f_{21} + vv'f_{22} + vf_{23} + u'f_{31} + v'f_{32} + f_{33} = 0 \]

      \[ Af = \begin{bmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & u_nv_n' & u_n & v_nv_n' & v_n & u_n' & v_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ \vdots \\ f_{33} \end{bmatrix} = 0 \]
8-point algorithm

1. Solve a system of homogeneous linear equations
   a. Write down the system of equations
   b. Solve $f$ from $Af=0$ using SVD

Matlab:

```matlab
[U, S, V] = svd(A);
f = V(:, end);
F = reshape(f, [3 3])';
```

`numpy.linalg.svd`

Singular Value Decomposition.

When $a$ is a 2D array, it is factorized as $u @ np.diag(s) @ vh = (u * s) @ vh$, where $u$ and $vh$ are 2D unitary arrays and $s$ is a 1D array of $a$'s singular values. When $a$ is higher-dimensional, SVD is applied in stacked mode as explained below.
SVD – as a way to obtain Eigen Values and Eigen Vectors of a Matrix $M^T M$

In any singular value decomposition

$$M = U \Sigma V^*$$

the diagonal entries of $\Sigma$ are equal to the singular values of $M$. The first $p = \min(m, n)$ columns of $U$ and $V$ are, respectively, left- and right-singular vectors for the corresponding singular values. Consequently, the

Questions?

https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf