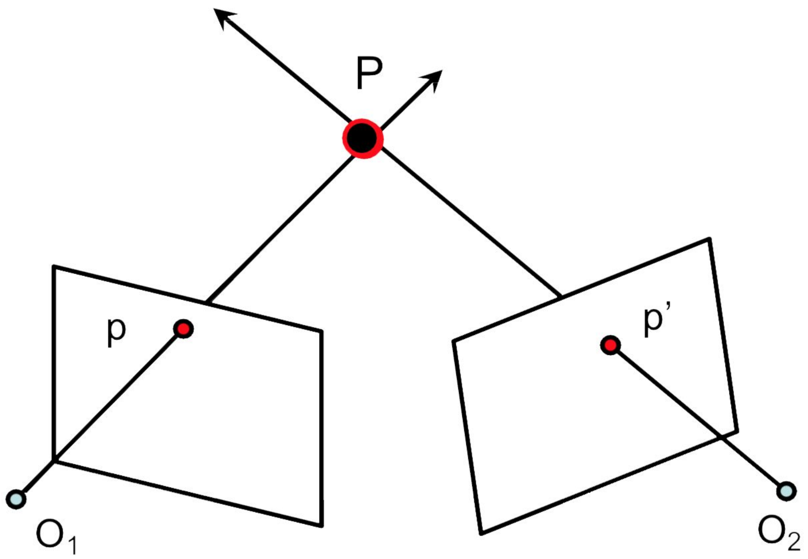
CSE 152 Section 7
HW3: Epipolar Geometry

November 16, 2018

Owen Jow

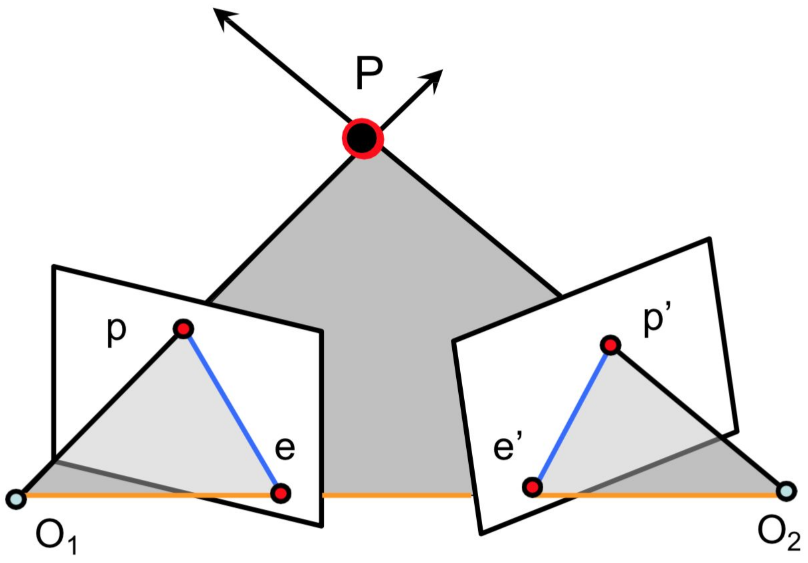
Epipolar Geometry



P	arbitrary 3D point
p	projection of **P** onto image 1
p'	projection of **P** onto image 2
O_1	pinhole (center of projection) of camera 1
O_2	pinhole (center of projection) of camera 2

image source: Hata, Savarese

Epipolar Geometry

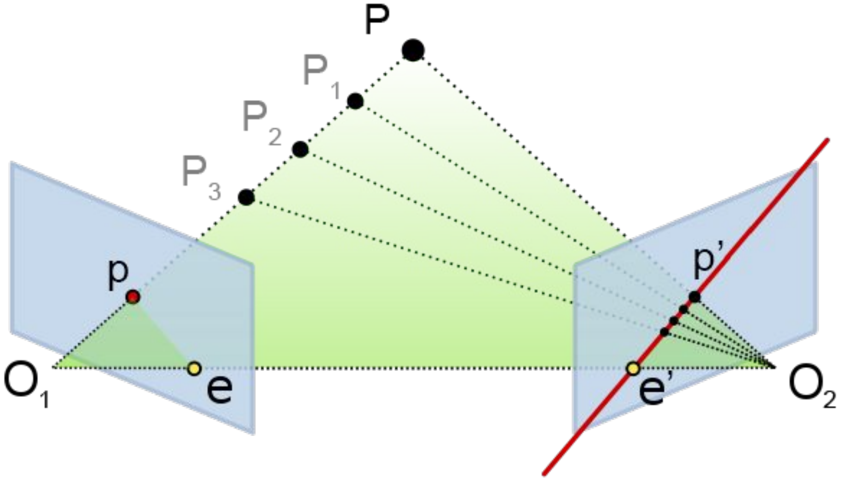


P	arbitrary 3D point
p	projection of **P** onto image 1
p’	projection of **P** onto image 2
O_1	pinhole (center of projection) of camera 1
O_2	pinhole (center of projection) of camera 2
e	epipole 1 (projection of **O_2** onto image 1)
e’	epipole 2 (projection of **O_1** onto image 2)

gray plane	epipolar plane (defined by **P**, **O_1**, **O_2**)
orange line	baseline (defined by **O_1**, **O_2**)
blue line	epipolar line (intersection of epipolar plane with image)

image source: Hata, Savarese

Epipolar Geometry



Epipolar constraint:
the point **p’** which corresponds to **p**
must lie on the epipolar line for image 2

an alternative interpretation of this epipolar line:
the projection of the line O_1 - P onto image 2

3D → 2D

Recall from the calibration lecture:

p = M P = K [R t] P

for **K** the 3x3 intrinsic (camera projection) matrix, **[R t]** the 3x4 extrinsic (camera pose) matrix

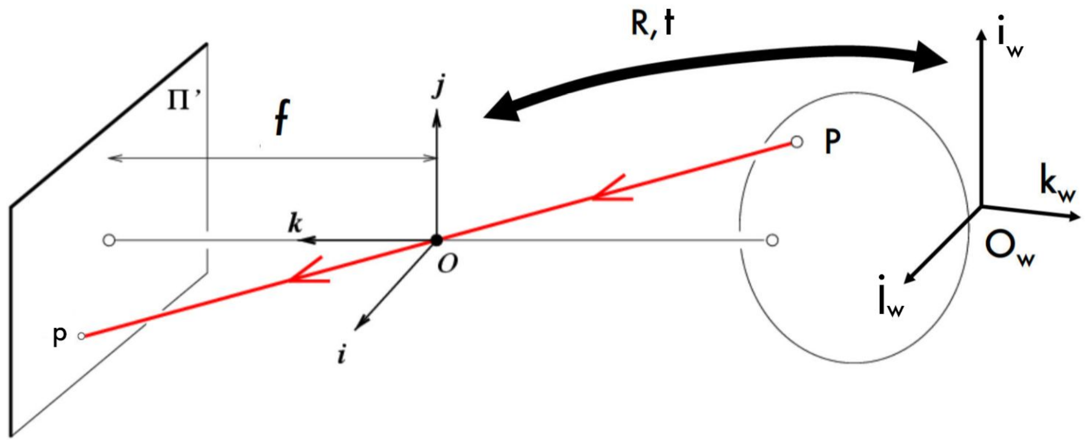


image source: Savarese

$3D \rightarrow 2D$

Let

$$\mathbf{M} \text{ (mapping for camera 1)} = \mathbf{K}\ [\mathbf{I}\ \ \mathbf{0}]$$
$$\mathbf{M'} \text{ (mapping for camera 2)} = \mathbf{K'}\ [\mathbf{R}\ \ \mathbf{t}]$$

i.e.

- world coordinates = camera 1 coordinates
- the transformation from a point $\mathbf{p_1}$ in camera 1 coords to a point $\mathbf{p_2}$ in camera 2 coords is $\mathbf{p_2 = Rp_1 + t}$ [$\rightarrow \mathbf{p_1 = R^T(p_2 - t)}$]
- $\mathbf{p'}$ in camera 1 coordinates is $\mathbf{R^T[(K')^{-1}p' - t]}$

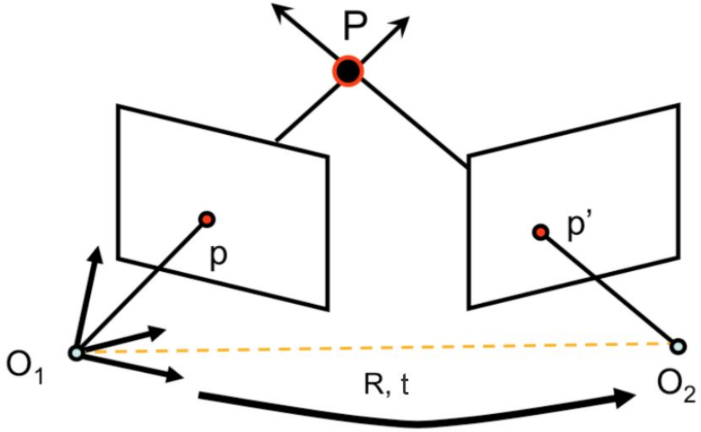


image source: Hata, Savarese

Fundamental Matrix

Then the fundamental matrix is

$$F = (K')^{-T}\, T_x\, R\, K^{-1}$$

where $\mathbf{T}_x$ is a skew-symmetric matrix corresponding to the translation vector $\mathbf{t}$: $\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$

- 3x3, rank 2, seven degrees of freedom
- relates corresponding points **p, p'** according to $\mathbf{(p')^T F p = 0}$ (epipolar constraint)
    - note that **p** is in homogeneous image 1 coords, **p'** is in homogeneous image 2 coords
- gives epipolar line in image 1 as $\mathbf{\{x : (F^T p')^T x = 0\}}$
- gives epipolar line in image 2 as $\mathbf{\{x : (Fp)^T x = 0\}}$
- $\mathbf{F^T e' = 0,\ Fe = 0}$
    - [epipolar point is on every epipolar line, so $\mathbf{(p')^T Fe = 0}$ for all **p'** and $\mathbf{(e')^T Fp = 0}$ for all **p**]

Eight-Point Algorithm

We can estimate the fundamental matrix using the **eight-point algorithm.**

Input: 8+ pairs of corresponding points $\mathbf{p}_i = (u_i, v_i, 1)$, $\mathbf{p}_i' = (u_i', v_i', 1)$
Output: fundamental matrix **F**

each correspondence is 1 equation $(\mathbf{p}_i')^T\mathbf{F}\mathbf{p}_i = \mathbf{0}$

$$\begin{bmatrix} u_i' & v_i' & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = 0$$

$$u_i u_i' F_{11} + v_i u_i' F_{12} + u_i' F_{13} + u_i v_i' F_{21} + v_i v_i' F_{22} + v_i' F_{23} + u_i F_{31} + v_i F_{32} + F_{33} = 0$$

$$\begin{bmatrix} u_i u_i' & v_i u_i' & u_i' & u_i v_i' & v_i v_i' & v_i' & u_i & v_i & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

Eight-Point Algorithm

We use 8+ equations to solve for the 8 independent entries in **F** (the ninth entry is a scaling factor).

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ u_3u_3' & v_3u_3' & u_3' & u_3v_3' & v_3v_3' & v_3' & u_3 & v_3 & 1 \\ u_4u_4' & v_4u_4' & u_4' & u_4v_4' & v_4v_4' & v_4' & u_4 & v_4 & 1 \\ u_5u_5' & v_5u_5' & u_5' & u_5v_5' & v_5v_5' & v_5' & u_5 & v_5 & 1 \\ u_6u_6' & v_6u_6' & u_6' & u_6v_6' & v_6v_6' & v_6' & u_6 & v_6 & 1 \\ u_7u_7' & v_7u_7' & u_7' & u_7v_7' & v_7v_7' & v_7' & u_7 & v_7 & 1 \\ u_8u_8' & v_8u_8' & u_8' & u_8v_8' & v_8v_8' & v_8' & u_8 & v_8 & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

$$W\mathbf{f} = 0$$

Eight-Point Algorithm

Approach: find a least-squares solution to this system of equations. **Can use SVD for this!**
Might also want to normalize each $\mathbf{p}_i$ and $\mathbf{p}_i'$ for better results (must de-normalize resulting **F** as well!).

1. Normalize points in each image according to **T** and **T'**, use normalized points to construct **W**.
2. Compute the SVD of **W**, reshape right singular vector into initial estimate of **F**.
    a. As reference, see sections 3 and 4 of this document.
3. Enforce rank = 2 by taking another SVD, this time of **F**, and zeroing out the last singular value.

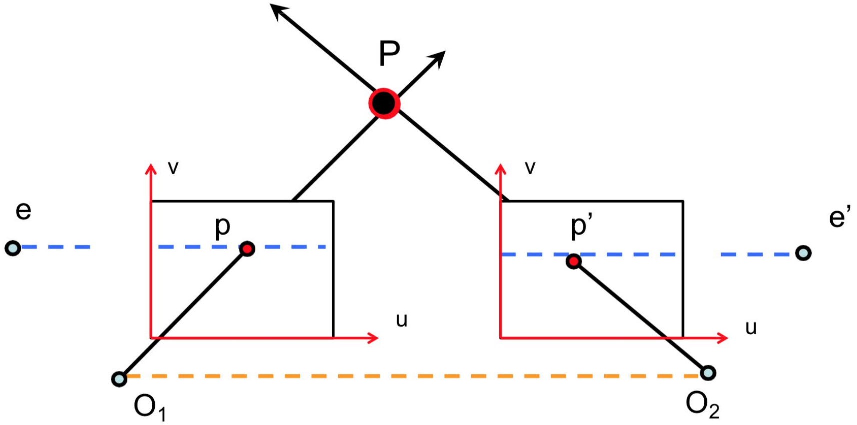
$$F = U \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

4. De-normalize **F**.
    a. Currently, $\mathbf{(T'p')^T F(Tp) = 0} \rightarrow \mathbf{(p')^T (T')^T F T p = 0}$, so $\mathbf{(T')^T F T}$ is the true fundamental matrix.

:)

courtesy Daniel Wedge

Question 1.1



Notes:

- no rotation, only translation

Hints:

[option 1] argue geometrically that all of the epipolar lines are parallel to the baseline

[option 2] compute the essential matrix $\mathbf{E} = \mathbf{T}_\times \mathbf{R}$
(difference is that **p, p'** are now normalized image coordinates)

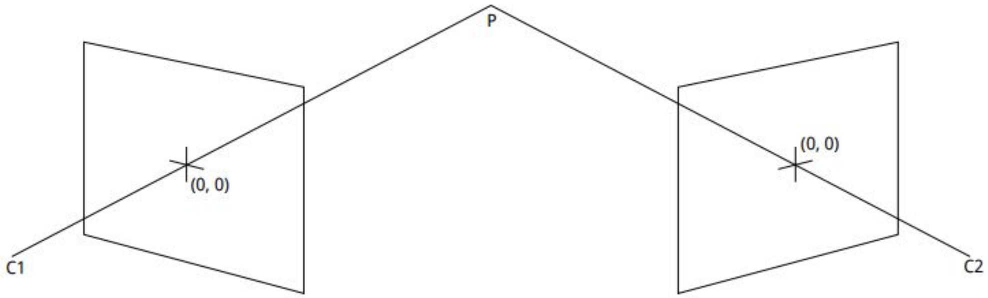
- what is the rotation matrix?
- what is the translation vector?
- what is the direction of each epipolar line?
  - **Ep** and $\mathbf{E}^T\mathbf{p'}$ give normals to lines
- what does $\mathbf{(p')^T E p = 0}$ tell us? (expand it)

image source: Hata, Savarese

Question 1.2

Hints:

- Under this setup,
  - what is the **p** corresponding to **P**?
  - what is the **p'** corresponding to **P**?
- What is the relationship between **p**, **p'**, and the fundamental matrix?



Question 2

1. **Eight-point algorithm**
   Estimate the fundamental matrix given point correspondences.
   for when you haven't done camera calibration and don't have the intrinsics/extrinsics

2. **Metric reconstruction**
   Estimate the camera matrices, triangulate and visualize the 3D points.

3. **3D correspondence**
   Estimate corresponding points given the fundamental matrix.

2.1. Eight-Point Algorithm

Notes:

- See eight-point algorithm slides for outline.
- **W** is an **n** x **9** matrix, where **n** is the number of correspondences.
- As an alternative to using SVD,
  you can define the initial **F** estimate as the eigenvector of $\mathbf{W}^T\mathbf{W}$ with the smallest eigenvalue.

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ & & & & \vdots & & & & \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix}$$

2.1. Eight-Point Algorithm

Normalization

Notes:

- Recall that we would like to precondition **W** before SVD.
  - To do so, normalize the pixel point coordinates through scaling and/or translation.
  - Then construct **W** from the normalized coordinates.
- In this homework, we suggest scaling by 1 / (the largest image dimension).
  - Although you shouldn't need to, you are free to do something else if you'd like.
  - For example, you can subtract the mean and divide by the standard deviation.
- **Don't forget to de-normalize the fundamental matrix at the end!**

2.2. Metric Reconstruction

1. **Load K1 and K2.**
   a. Load the intrinsic matrices **K1** and **K2** from **`temple/intrinsics.mat`**.
   b. Documentation: https://www.mathworks.com/help/matlab/ref/load.html

2.2. Metric Reconstruction

2. **Find M2 and M1.**
   a. Recover camera 2's extrinsic matrix **[R t]** using **`camera2`**.
      i. To obtain the full 3D → 2D matrix **M2**, multiply by the intrinsic matrix **K2**.
   b. Define camera 1's frame to be the world coordinate frame.
      i. Don't forget to multiply by **K1**!

Notes:

- The spec says that **`camera2`** returns **M2**, but it only returns the extrinsic matrix.

2.2. Metric Reconstruction

3. **Load the correspondences for 3D visualization.**
    a. Load the correspondences **`x1`, `y1`, `x2`, `y2`** from **`many_corresp.mat`**.
    b. Documentation: https://www.mathworks.com/help/matlab/ref/load.html

2.2. Metric Reconstruction

4. **Get 3D points given 2D point correspondences.**
   a. Use the [`triangulate`] function (provided).

```
P = triangulate(M1, pts1, M2, pts2)
```

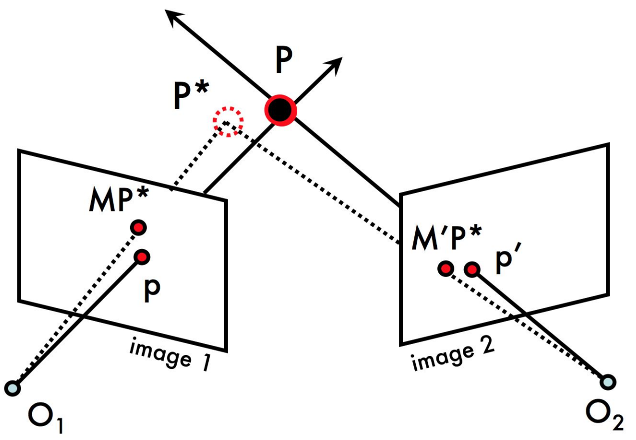


image source: Savarese

2.2. Metric Reconstruction

5. **Plot 3D points.**
    a. Use the `scatter3` function.
    b. Documentation: https://www.mathworks.com/help/matlab/ref/scatter3.html

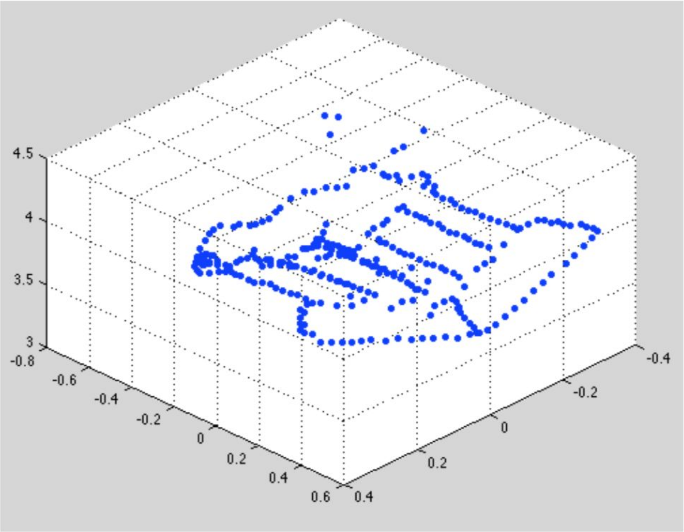


image source: Meng Song

2.3. 3D Correspondence

Notes:

- In this problem, we take advantage of the epipolar constraint to search for corresponding points.
- We are given **p = (x1, y1)** in image 1, and we would like to find **p'** in image 2.

Compare **the window around (x1, y1) in image 1**
to **the window around each point on the epipolar line in image 2**.

The point in image 2 with the minimum window distance is our match.

- We can weight the window according to a 2D Gaussian when computing the difference.
- To speed things up, we can look **only** at points along the line which are close to **(x1, y1)**.
    - for this data, we know the images are not that different

2.3. 3D Correspondence

Computing the epipolar line:

- The epipolar line associated with **p** is ℓ = **Fp**.
- The equation of the line is $\ell^T \mathbf{x} = \mathbf{0}$.
    - i.e. if $\ell = [\ell_1, \ell_2, \ell_3]^T$ and $\mathbf{x} = [u, v, 1]$, then the equation of the line is $\ell_1 u + \ell_2 v + \ell_3 = 0$
    - in $\mathbf{y = mx + b}$ form, the equation of the line is **v = ...** (you can figure this out)

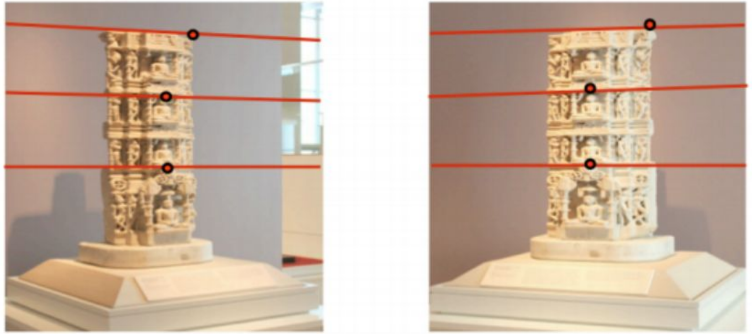


image source: Hata, Savarese

Additional Readings

- CS 231A course notes
- How to use SVD to solve homogeneous linear least-squares