CSE 559A: Computer Vision

Fall 2018: T-R: 11:30-1pm @ Lopata 101

Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Course Staff: Zhihao Xia, Charlie Wu, Han Liu

http://www.cse.wustl.edu/~ayan/courses/cse559a/

Oct 4, 2018

GENERAL

- Printed Solution Keys for Problem Set 1
  - Take only one.
  - Note academic integrity warning on top.
- Recitation Tomorrow: 10:30AM-Noon, Jolley 309.
- PSET 2 Due Tuesday Oct 9, 11:59 PM

PROJECT

- **Option 1**: Read and analyze a computer vision paper.
  - Recent or classic from ICCV, ECCV, CVPR. (Suggestions posted)
  - Either implement, or if implementation available, modify / analyze.
  - Key is to demonstrate you understood method, and why it was needed.
- **Option 2**: Apply what you've learned in class to a problem you care about.
  - Read up on most relevant related work.
  - Implement adapted method for your problem.
  - Analyze results. Did it work ? If so, how well. If not, why not.
- Should be roughly 2x the effort of a problem set. (Let's say, problem set 2)
- Avoid anything that requires training neural networks !

PROJECT

Read through the project section on the course website

Grading

- 25 points Report
  - Abstract: 3 pts (One paragraph succinct summary)
  - Introduction/Motivation: 5 pts
    Why is this problem important, what is the vision task, prelude to rest of the report.
  - Related work: 4 pts
    How have other people solved it ? What are other similar problems ? Read, describe.
  - Description / Experiments / Technical Correctness: 10 pts
  - Conclusion: 3 pts

2-3 Paragraph Proposal is due 11:59pm Oct 18th.

Submit as a text file through git (repo will be created shortly).

3D HOMOGENEOUS CO-ORDINATES

- Four dimensional vector defined upto scale: $p = [\alpha x, \alpha y, \alpha z, \alpha]^T$
- If l is a four-dimensional vector, what does $l^T p = 0$ represent ? A plane.
- How do we represent a line ? $L^T p = 0$ where L is a 4×2 matrix.
- Interpret line as intersection of two planes.

3D TRANSFORMATIONS

Represented by 4×4 matrices.

- Translation

$$p' = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} p$$

3D TRANSFORMATIONS

Represented by 4×4 matrices.

- Rotation

$$p' = \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix} p$$

Where R is now a 3×3 matrix with $R^T R = I$.

Also covers reflection. For rotation only, $R = R_x(\theta_1) R_y(\theta_2) R_z(\theta_3)$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \quad R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Corresponds to rotation around each axis. Not commutative.

3D TRANSFORMATIONS

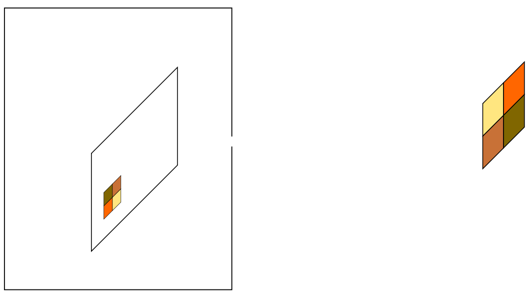
General Euclidean Transformation

$$p' = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} p$$

Now R is a 3×3 rotation matrix, and t is a 3×1 translation vector.

CAMERA PROJECTION

Sensor to Image Locations

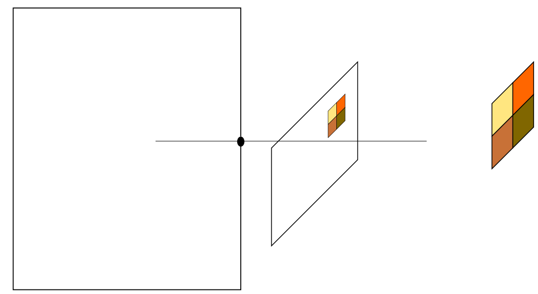


But then, the image formed on the sensor is flipped before we see it as an array.

$$p = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p'$$

CAMERA PROJECTION

Sensor to Image Locations



But then, the image formed on the sensor is flipped before we see it as an array.

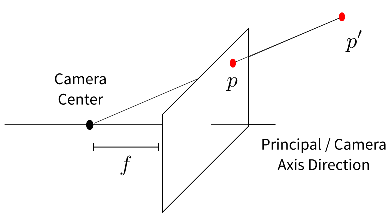
So factoring that in (and assuming the y-coordinate increases from bottom to top).

$$p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p'$$

(You'll see both versions in different textbooks/papers)

CAMERA PROJECTION

Sensor to Image Locations



Think of the sensor plane as being *in front* of the pinhole, and the image is what you "frame" on that plane.

But then, the image formed on the sensor is flipped before we see it as an array.

So factoring that in (and assuming the y-coordinate increases from bottom to top).

$$p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p'$$

(You'll see both versions in different textbooks/papers)

CAMERA PROJECTION

Sensor to Image Locations

$$p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p'$$

- This still assumes that p' and p share the same co-ordinate system.
- What units is p in ? What units is f in ?

Meters to Pixels

- Location on sensor plane in meters: $x'_m = f\frac{x}{z}$
- Let's say each sensor pixel is s meters wide.
- Location in 'pixels' is $x_p = x'_m/s = \frac{f}{s}\frac{x}{z}$
- Or can just assume f is focal length in pixels.

CAMERA PROJECTION

$$p = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p'$$

More General:

- $f_x \neq f_y$ handles the case where pixels aren't square (so you have f in meters divided by sensor width and sensor height separately)
- $s \neq 0$ implies the pixels are skewed (almost never happens).
- c_x and c_y just picks the location of origin on the image plane.

Often, ok to assume $s = 0, f_x = f_y = f, c_x = W/2, c_y = H/2$.

CAMERA PROJECTION

$$p = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p'$$

Still assumes that p' is with respect to an "aligned" co-ordinate system:

- Camera center (pinhole) is at origin
- x and y axes aligned with sensor plane
- z axis is viewing direction

CAMERA PROJECTION

$$p = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} p' = [K \ \ \mathbf{0}] \ p'$$

Still assumes that p' is with respect to an "aligned" co-ordinate system:

- Camera center (pinhole) is at origin
- x and y axes aligned with sensor plane
- z axis is viewing direction

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The 3×3 matrix K is called the intrinsic camera matrix.

CAMERA PROJECTION

- But what if p' is in some other co-ordinate system ?
  - Calibration target (trying to estimate camera parameters)
  - Multi-view Scenario
- Define p'' and p' are 3D homogeneous co-ordinates:
  - p'' is in camera aligned axes, p' is in world axes
  - Both are related by a euclidean / 'rigid' transformation (rotation + translation)

$$p'' = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} p'$$

Where R is 3×3 3-D rotation matrix, and t is 3×1 translation vector.

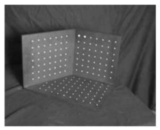
$$p = [K \ \ 0] \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} p' = K \ \ [R|t] \, p' = Pp'$$

The projection matrix P can be factorized into the upper triangular matrix 3×3 intrinsic matrix K, and the 3×4 **extrinsic matrix** $[R|t]$ that represents camera "pose".

CAMERA CALIBRATION

$$p = [K \ \ 0] \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} p' = K \ [R|t] \, p' = Pp'$$

- P defined upto scale.
- Get a bunch of 3D-2D correspondences (p_i', p_i)
- Solve for $p_i \times (Pp_i') = 0$ like for Homographies
- Except that now P is a 3×4 matrix instead of 3×3
- Need six linearly independent points (three if K is known).



- Once you have P, can decompose into K and $[R|t]$ using QR factorization
- Restricted versions possible if you assume no skew, square pixels, etc.

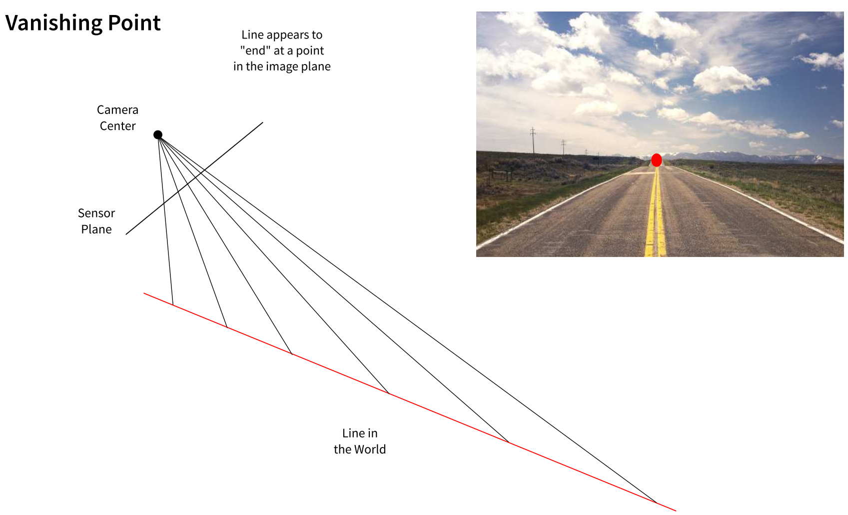
CAMERA CALIBRATION

- $P = K[R|t]$ describes projection from calibration object's co-ordinate system
- But really, most of the time we just want to estimate K.
- Assume square pixels, no skew, optical center at center of image.

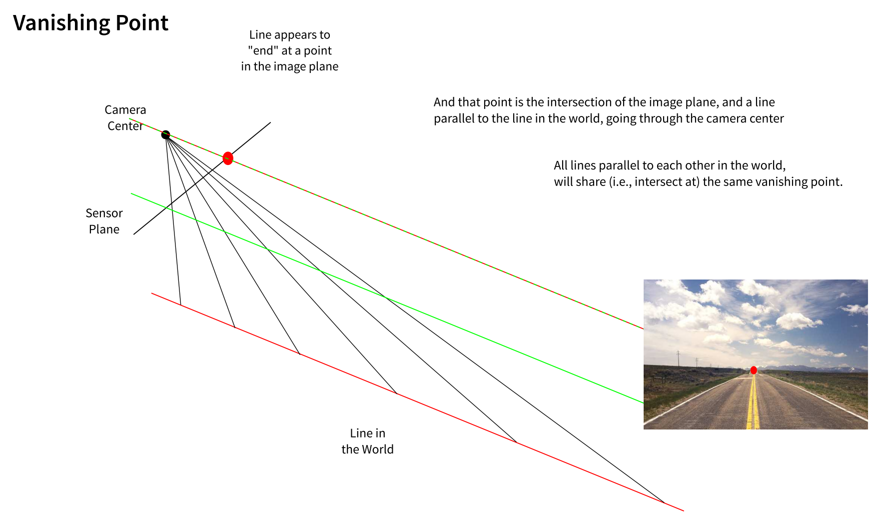
$$K = \begin{bmatrix} f & 0 & W/2 \\ 0 & f & H/2 \\ 0 & 0 & 1 \end{bmatrix}$$

Is there a simpler way to get f ?

CAMERA CALIBRATION



CAMERA CALIBRATION



CAMERA CALIBRATION

Vanishing Point

Alternate equation of line in 3D:

In 3-D cartesian, all points r that satisfy for some scalar λ: $\quad r = r_0 + \lambda d$, where

- r_0 is a 3-vector representing the cartesian co-ordinate of a point,
- d is a 3-vector representing "direction" of line,
- Same r_0 and different scaled versions of d represent same line.
- Two lines with different r_0 but same d (upto scale) are parallel to each other.
- $r = \lambda d$ represents parallel line passing through origin.

In homogeneous co-ordinates, $p = [(r_0 + \lambda d)^T, 1] = [(\frac{1}{\lambda} r_0 + d)^T, \frac{1}{\lambda}]$

Projection $\tilde{p}$ of p (assuming camera-aligned co-ordinate system):

$$\tilde{p} \sim [K \;\; 0]p = Kr_0 + \lambda Kd \sim \frac{1}{\lambda} Kr_0 + Kd$$

CAMERA CALIBRATION

Vanishing Point

Projection $\tilde{p}$ of p (assuming camera-aligned co-ordinate system):

$$\tilde{p} \sim [K \;\; 0]p = Kr_0 + \lambda Kd \sim \frac{1}{\lambda} Kr_0 + Kd$$

- As $\lambda \to \infty, \tilde{p} \sim Kd$
- Kd is the 2D homogeneous co-ordinate of the projection of the point on the given line at infinity. It is the projection of all points on the line parallel to the given line and passing through origin / camera center (same $d, r_0 = 0$).
- d represents a ray in $\mathbb{R}^3$. All points in parallel line through origin have co-ordinate $[d^T, 1/\lambda]$ for some λ, and all project to Kd.
- Note that vanishing point will be at infinity if z-component of d is 0, i.e., if line in 3D space is perpendicular to camera axis.

CAMERA CALIBRATION

Vanishing Point

- $p \sim Kd \Rightarrow K^{-1}p \sim d$
- If p's cartesian co-ordinate is (x, y), for simple K:

$$d \sim K^{-1}p \sim \begin{bmatrix} (x - W/2) \\ (y - H/2) \\ f \end{bmatrix}$$

So I can write an equation relating d to the co-ordinate of it's vanishing point and unknown focal length f.

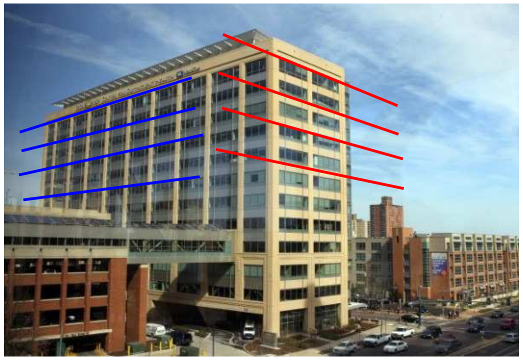
But I don't know d.

But what if I knew that d_1 and d_2 were perpendicular (in the real world) ?

CAMERA CALIBRATION

Vanishing Point

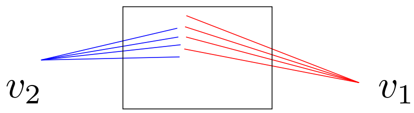
Find two sets of lines where
- All lines in each set are parallel to each other
- Lines in different sets are perpendicular to each other



CAMERA CALIBRATION

Vanishing Point

Find two sets of lines where
- All lines in each set are parallel to each other
- Lines in different sets are perpendicular to each other

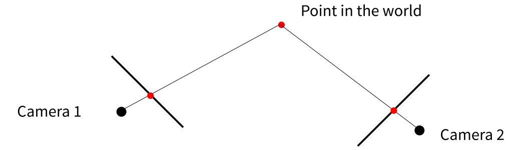


Find vanishing point for each set by finding intersection of lines (intersection might be outside image)

$$d_1 \sim K^{-1} v_1$$
$$d_2 \sim K^{-1} v_2$$
$$d_1^T d_2 = 0$$

Solve for focal length.

TWO-VIEW GEOMETRY



$$\tilde{p}_1 \sim K_1[R_1|t_1]p$$
$$\tilde{p}_2 \sim K_2[R_2|t_2]p$$

What can we say about the relationship between $\tilde{p}_1$ and $\tilde{p}_2$, and what does it say about p?

TWO-VIEW GEOMETRY

$$\tilde{p}_1 \sim K_1[R_1|t_1]p, \qquad \tilde{p}_2 \sim K_2[R_2|t_2]p$$

Let's just assume $K_1 = K_2 = K$,

and the co-ordinate system is aligned with the first camera: $R_1 = I, t_1 = 0$.

TWO-VIEW GEOMETRY

$$\tilde{p}_1 \sim K[I|0]p, \qquad \tilde{p}_2 \sim K[R|t]p$$

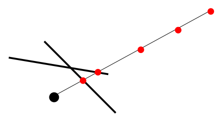
What if $t = 0$? Second image is from just rotating the camera, but not moving it's center.

Let $p = [x, y, z, 1]$. We're going to deal with $\sim$ by saying equal to some scalar factor $\lambda_1, \lambda_2, \ldots$

$$\tilde{p}_1 = \lambda_1 K[I|0]p = \lambda_1 K[x, y, z]^T, \quad \text{for some } \lambda_1$$
$$\tilde{p}_2 = \lambda_2 K[R|0]p = \lambda_2 KR[x, y, z]^T, \quad \text{for some } \lambda_2$$
$$\tilde{p}_2 = \frac{\lambda_2}{\lambda_1} KRK^{-1}\tilde{p}_1 \sim KRK^{-1}\tilde{p}_1$$

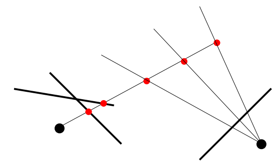
So if there's only rotation, points in two images can be related by a Homography $= KRK^{-1}$.

TWO-VIEW GEOMETRY



Mapping doesn't depend on depth if only rotation.

TWO-VIEW GEOMETRY



Will depend with translation.

Mapping doesn't depend on depth if only rotation.