# Estimating focal length with only CV

Jordan Richards

October 22, 2021

## 1 fSpy

`https://github.com/stuffmatic/fSpy`

Attempts to find vanishing points and focal length based on user inputted perpendicular lines in the image, but doesn't do the best job for almost parallel lines. Can we achieve better accuracy if we cut user input of perpendicular lines?

## 2 Vanishing Points

`https://www.cs.princeton.edu/courses/archive/fall13/cos429/lectures/11-epipolar.pdf`

- if we have accurate vanishing points, we can get the focal length
- we need accurate parallel lines to estimate the vanishing point, so need some way to find those so maybe impossible for our project (where the only dependable straight lines are probably the pyramids)

### 2.1 Hough Transform

`http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09.pdf`

- Detects probably straight lines
- Doesn't necessarily find probably parallel lines, but I guess we can do some clustering
- Maybe ML solution to finding good lines in our image

### 2.1.1 algorithm

- sobel or canny to find edge points
- map points to hough transform (additive)
- find possible centers corresponding to valid lines
- assuming lines fit into perpendicular groups, valid vanishing points should be at the intersection of some of these lines
- (alt) find vanishing points directly by fitting sine curves to hough transform
  - `https://link.springer.com/content/pdf/10.1007/3-540-48311-X_137.pdf`

### 2.1.2 caveats

- NOT ALL SCENES HAVE APPROPRIATE STRUCTURE FOR THE DETECTION OF ORTHOGONAL VANISHING POINTS

## 2.2 Motion vectors

`http://www.itspy.cz/wp-content/uploads/2014/11/acmspy2014_submission_25.pdf#page=64&zoom=100,130,908`

- Even if our scene might lack good target lines, we have 2 cameras
- In the paper, motion vectors can be drawn between consecutive frames where a car moves relative to the camera
- In our case, the entire scene has moved relative to the camera

### 2.2.1 algortithm

- extract good features from both camera frames
- (maybe need to align the rotation of the 2 frames using the features)
- map the features from each frame to produce "motion" lines
- use the lines/vectors to find vanishing point(s)

#### 2.2.2 caveats

- THIS DOES ASSUME BOTH CAMERAS HAVE THE SAME FOCAL LENGTH

## 3 Focal Length from Vanishing Points

- `https://stackoverflow.com/questions/53873082/focal-length-computation-from-vanish`
- `http://www.itspy.cz/wp-content/uploads/2014/11/acmspy2014_submission_25.pdf#page=64&zoom=100,130,908`

```
f = sqrt(-(U - P) * (V - P))
```

- f: focal length
- P: principal point in image space (probably the center, assuming the image is unedited)
- U, V: vanishing points in image space

## 4 Homogenous Coordinates

`https://web.stanford.edu/class/cs231a/course_notes/02-single-view-metrology.pdf`

### 4.1 normalization

- Points of the form <x, y, 1> when normalized.
- We can recover our 2d image coordinates (x, y) from any homogenous vector by dividing by the third parameter: <x, y, z> -> <x/z, y/z, 0>

### 4.2 points at infinity

- <x, y, 0> is a point at infinity in the direction of (x,y)

### 4.3 lines

- Lines described using vectors <a, b, c>, where
  - -a/b is the slope
  - -c/b is the y intercept
- A point, p, is on a line, l, when p  = 0
- The intersection of 2 lines, $\alpha$ and $\beta$ is $\alpha \times \beta$
  - the intersection of parallel lines is a point at infinity in the direction of the lines <b, -a, 0>

### 4.4 transformations

- Matrices on homogeneous coordinates are useful for describing transformations
- Can map between normal points and points and infinity

### 4.5 3D

- Follows from 2D:
  - Point <x, y, z, 1>
  - Plane <a, b, c, d>, Point on Plane lane = 0
- Projection mapping, H: mapping from 3d point to camera space

### 4.6 Vanishing points and vanishing lines

- Vanishing point: given a 3d direction <a, b, c> we can apply the camera intrinsic matrix K to get a point in image space
  - Importantly this process is reversible, so if we can find the vanishing point in 2D, we can find the direction in 3D
- Given a coplanar set of pairs of parallel lines, we can find the intersection point for each pair (a set of points and infinity) and this set of points will be co-linear on a line at infinity
  - Applying our projective transformation to this line at infinity, we get a vanishing line or horizon line

    - The normal vector for our plane can be computed directly using this vanishing line and K

- We can use these relationships to calculate the angles between vectors or planes in our image:
    - vanishing points -> 3D vectors -> angle
    - vanishing lines -> normal of 3D planes -> angle
    - with 3 vanishing points from 3 pair-wise perpendicular planes, we can use this relationship to solve for K

## 5 TODO extra sources

### 5.1 `http://www.fit.vutbr.cz/research/groups/graph/pclines/papers/2013-BMVC-Dubska-VanishingPointsDetection.pdf`

Diamond space ???

### 5.2 `https://ieeexplore.ieee.org/document/5456535`

### 5.3 `https://ieeexplore.ieee.org/document/6671929`

### 5.4 `https://dl.acm.org/doi/10.1145/3402597.3402602`

Not sure if I can get access to this one