Lecture 3 Homework

TechX Academy Computer Vision 2019

1 Projective Line

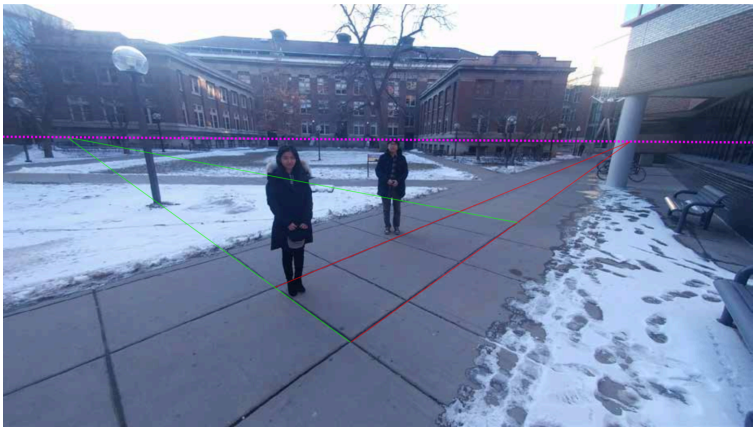

Figure 1: Figure 2. (a) You will use your cellphone camera image to compute camera poses with respect to the ground plane and measure the height of your friend given the height of an object. (b) You will paste the OpenCV logo to the ground plane

Write-up:

1. Derive and compute two vanishing points and a vanishing line, and visualise them on your image similar to Figure 2(a).
2. Compute camera rotation matrix, $\boldsymbol{R}$, and visualise 3D camera axes with respect to the ground plane axes using Python *Axes3D.plot* in *matplotlib* (You are free to use other functions/modules if you prefer). Give a geometric interpretation of the computed rotation matrix.
3. Measure the heights of at least 3D object given your friend's height using the cross ratio. Verify the height measurements.
4. Project OpenCV logo onto the ground plane or any planar surface. You are also free to choose **different** logo or image.

2 Image Formation and Rigid Body Transformations

In this problem we will practice rigid body transformations and image formations through the projective camera model. The goal will be to photograph the following four points $\boldsymbol{X}_1 = [-1 \ -0.5 \ 2]^T, \boldsymbol{X}_2 = [1 \ -0.5 \ 2]^T, \boldsymbol{X}_3 = [1 \ 0.5 \ 2]^T, \boldsymbol{X}_4 = [-1 \ 0.5 \ 2]^T$ in the world coordinate frame.
The formula for rigid body transformation is

$$\widetilde{\boldsymbol{X}}_{cam} = R\widetilde{\boldsymbol{X}} + \boldsymbol{t} \tag{1}$$

where $\widetilde{\boldsymbol{X}}_{cam}$ is the point coordinate in the camera coordinate system. $\widetilde{\boldsymbol{X}}$ is a point in the world coordinate frame, and R and $\boldsymbol{t}$ are the rotation and translation that transform points from the world coordinate frame to the camera coordinate frame. Together, R and $\boldsymbol{t}$ are the *extrinsic* camera parameters. Once transformed to the camera coordinate frame, the points can be photographed using the 3×3 camera calibration matrix $\boldsymbol{K}$, which embodies the *intrinsic* camera parameters, and the canonical projection matrix $[\boldsymbol{I}|\boldsymbol{0}]$. Given $\boldsymbol{K}, R$, and $\boldsymbol{t}$, the image of a point $\widetilde{\boldsymbol{X}}$ is $\boldsymbol{x} = \boldsymbol{K}[\boldsymbol{I}|\boldsymbol{0}]\boldsymbol{X}_{Cam} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\boldsymbol{X}$, where the homogeneous points $\boldsymbol{X}_{Cam} = (\widetilde{\boldsymbol{X}}_{Cam}^T, 1)^T$ and $\boldsymbol{X} = (\widetilde{\boldsymbol{X}}^T, 1)^T$. We will consider four different of focal length, viewing angles and camera positions below:

(a) [No rigid body transformation] Focal length = 1. The optical axis of the camera is aligned with the z-axis.

(b) [Translation] $\boldsymbol{t} = [0 \ 0 \ 1]^T$. The optical axis of the camera is aligned with the z-axis.

(c) [Translation and Rotation] Focal length = 1. R encodes a 30° around the z-axis and then 60° around the y-axis. $\boldsymbol{t} = [0 \ 0 \ 1]^T$.

(d) [Translation and Rotation, long distance] Focal length = 5. R encodes a 30° around the z-axis and then 60° around the y-axis. $\boldsymbol{t} = [0 \ 0 \ 13]^T$.

For each of the cases above, please calculate the required information below, and integrate them into the code template provided (see attachment *plot_ points.py*).

1. The extrinsic transformation matrix,
2. Intrinsic camera matrix under the perspective camera assumption
3. Calculate the image of the four vertices and plot using the supplied **plot_points** function (see e.g. output in figure below).

We will not use a full intrinsic matrix (e.g. that maps centimeters to pixels, and defines the coordinates of the center of the image), but only parameterise this with f, the focal length. In other words: the only parameter in the intrinsic camera matrix under the perspective assumption is f.

For all the four cases, include a image like above. Note that the axis are the same for each row, to facilitate comparison between the two camera models. Note: the angles and offsets used to generate these plots may be different from those in the problem statement, it is just to illustrate how to report your results.

Also, Explain why you observe any distortions in the projection, if any, under this model.

3 Hints

3.1 Projective Line

Vanishing Line — Hough Technique breakdown by stages

1. Use Hough transform to accumulate votes for likely vanishing point candidates: use pairs of detected line segments to form candidate vanishing point locations.

   Additional materials:

   - About the expression for a straight line: (en-US) Hough Transform Equation
   - About the understanding of Hough Transformation: (zh-CN) How to interpret that all collinear points have same (r, θ) pair under Hough Transformation?
   - Another article about Hough Transformation: (zh-CN) OpenCV notes: Hough Transformation
   - For edge detection, you may use function *cv2.Canny* from OpenCV libraries.

   Hough transformation step by step: (translated from the second link above, at the end of section 1)

   (a) Set up a 2 dimensional array M with desired range of (r, θ) (Scaling is probably required since a typical range of θ which represents an angle is $[0, 2\pi)$ yet only integers are permitted to be used as the indices). The array M would be used as an accumulator later on.

   (b) Transform each point on the edge detected in the image into the parametric space (i.e., use a pair of (r, θ) to represent it). To be specific, calculate using the following expression for each point (x, y) in the image:

   $$\rho = x \cos\theta + y \sin\theta \text{ with each arbitrary } \theta \in [0, 2\pi) \text{ you selected in step (a)} \tag{2}$$

   Increase the accumulator $M(r, \theta)$ with corresponding (ρ, θ) for each pair of values you calculated.

   (c) Scan the array M with an arbitrary threshold value T. A straight line with parameters (r, θ) it thought to exist if $M(r, \theta) > T$. You may try different values of T until you obtain a satisfactory result.

   (d) Calculate the straight lines with (r, θ) obtained in previous step.

Reference

1. Problem 3 in CSCI 5980 Assignment #2
2. Problem 2 in CSE 152 Assignment 2