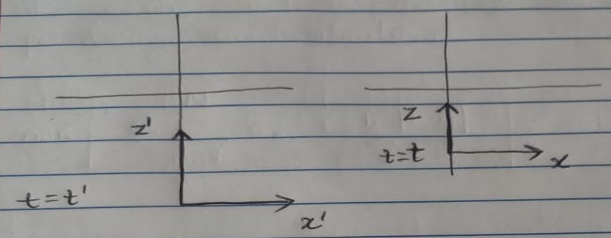
ECE 4554 / 5554: Computer Vision: Homework 4

Name: *Ankit Parekh*
VT Username: *ankitparekh*
Program: *MS CPE (Non-Thesis)*

Problem 1)



a) Essential matrix (E) is given by: $E = [t]_x R$

For pose 1, $t = t'$: Assuming position of camera (x_1, y_1, z_1) and orientation is R_1

For pose 2, $t = t$: Position of camera $(x_2, y_2, z_2) = (x_1+f, y, z_1+f)$ is given, and there is no rotation so $\underline{R_2 = R_1}$

Hence, $[t]_x = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} = \begin{bmatrix} 0 & -f & 0 \\ f & 0 & -f \\ 0 & f & 0 \end{bmatrix}$

Since, orientation has not changed b/w poses

$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$\therefore E = \begin{bmatrix} 0 & -f & 0 \\ f & 0 & -f \\ 0 & f & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -f & 0 \\ f & 0 & -f \\ 0 & f & 0 \end{bmatrix}$ //

b) Let e' be the epipole in the image plane at time t', e be the epipole at time t.

To calculate e we can use: $Ee = 0$ Let $e = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

$$\begin{bmatrix} 0 & -f & 0 \\ f & 0 & -f \\ 0 & f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -yf \\ fx - fz \\ yf \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$\therefore y = 0, \quad x = z$

$\therefore$ For $z = 1$, $\underline{\underline{e = (1, 0, 1)}}$

Problem 2)

a) Given: $t = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$ $R = \begin{bmatrix} \cos(45°) & -\sin(45°) & 0 \\ \sin(45°) & \cos(45°) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Essential Matrix $E = [t]_x R$

$[t]_x = \begin{bmatrix} 0 & -1 & 2 \\ 1 & 0 & -3 \\ -2 & 3 & 0 \end{bmatrix}$ $R = \begin{bmatrix} 0.707 & -0.707 & 0 \\ 0.707 & 0.707 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$E = [t]_x R = \begin{bmatrix} -0.707 & -0.707 & 2 \\ 0.707 & -0.707 & -3 \\ 0.707 & 3.535 & 0 \end{bmatrix}$

b)

```
# Singular-value decomposition
from numpy import array
from scipy.linalg import svd
U, s, VT = svd(E)
print("E : {}".format(E))
print("U : {}".format(U))
print("s : {}".format(s))
print("V : {}".format(VT.T))
```

```
E : [[-0.70710678 -0.70710678  2.        ]
 [ 0.70710678 -0.70710678 -3.        ]
 [ 0.70710678  3.53553391  0.        ]]
U : [[ 0.57735027  0.15430335  0.80178373]
 [-0.57735027 -0.6172134   0.53452248]
 [-0.57735027  0.77151675  0.26726124]]
s : [3.74165739e+00 3.74165739e+00 1.71464245e-16]
V : [[-0.32732684  0.          0.94491118]
 [-0.54554473  0.81649658 -0.18898224]
 [ 0.77151675  0.57735027  0.26726124]]
```

```
# Reconstructing E
# create m x n Sigma matrix
Sigma = np.zeros((E.shape[0], E.shape[1]))
# populate Sigma with n x n diagonal matrix
Sigma[:E.shape[1], :E.shape[1]] = np.diag(s)
print("s = ", Sigma)
# reconstruct matrix
B = U.dot(Sigma.dot(VT))
print("Original Matrix reconstruction : ", B)
```

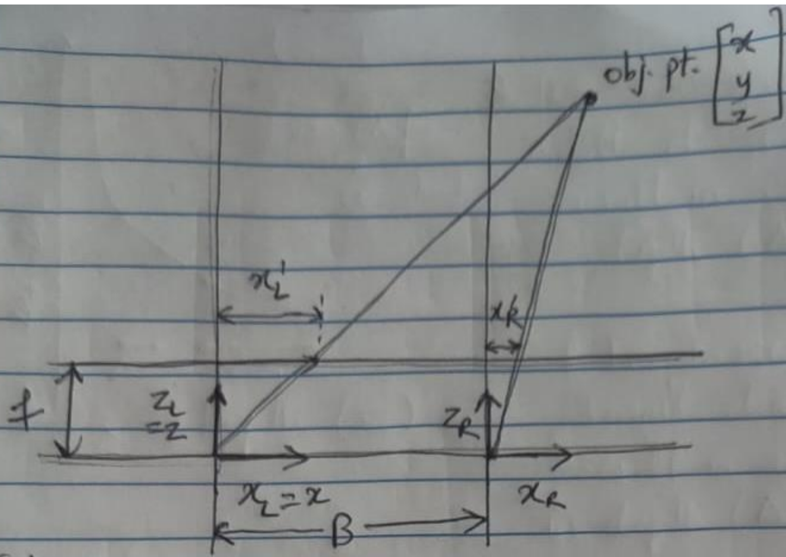
```
s =  [[3.74165739e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 3.74165739e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.71464245e-16]]
Original Matrix reconstruction :  [[-7.07106781e-01 -7.07106781e-01  2.00000000e+00]
 [ 7.07106781e-01 -7.07106781e-01 -3.00000000e+00]
 [ 7.07106781e-01  3.53553391e+00  1.73974536e-16]]
```

Rank of essential matrix is number of non-zero values in s

```
rank = 0
for element in s:
    if abs(element) > 10**(-15):
        rank+=1
print("Rank of s : ", rank)
```

```
Rank of s :  2
```

Problem 3)
a)



Given:

$f = 0.025\ m$; $B = 0.15 m$; $(x_L', y_L') = (0.005, 0)$ & $(x_R', y_R') = (0.003, 0)$

Using eqn. for stereo backprojection:

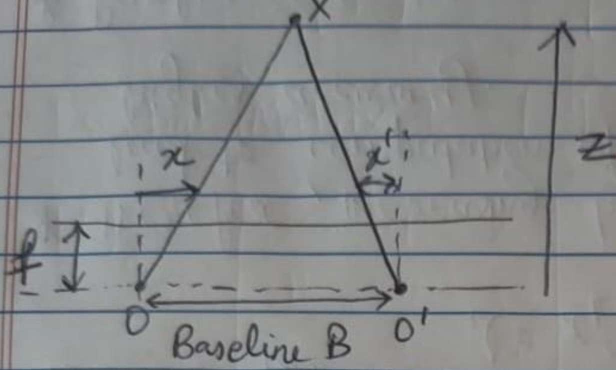
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{B}{x_L' - x_R'} \begin{bmatrix} x_L' \\ y_L' \\ f \end{bmatrix} \quad (\because y_L' = y_R')$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{0.15}{0.005 - 0.003} \begin{bmatrix} 0.005 \\ 0 \\ 0.025 \end{bmatrix} = \begin{bmatrix} 0.375 \\ 0 \\ 1.875 \end{bmatrix}$$

$(x, y, z) = (0.375, 0, 1.875) m$ (in meters)

Problem 3) b) We know that we can estimate depth from disparity.

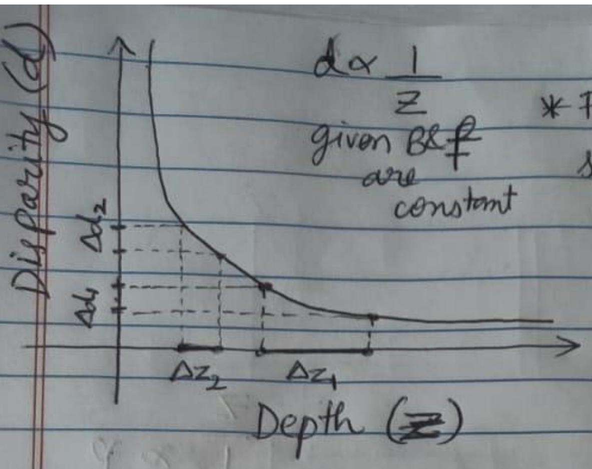
Consider the setup given below:



$$\frac{x - x'}{O - O'} = \frac{f}{Z}$$

$$\text{disparity} = x - x' = \frac{B.f}{Z}$$

$Z \leftarrow$ depth



* For small values of d, a small change in d (Δd_1) will result in a large change in depth (Δz_1).

* For large values of d, a small change in d (Δd_2), will result in an even smaller change in depth (Δz_2).

* For the problem of depth estimation: if we want lower error in depth, we should keep the value of disparity large.

* Since, we cannot change f (focal length) or depth to estimate depth, and in order to increase accuracy of depth, we need to measure a large value for disparity which can be done by increasing the baseline distance.

$$d = x_L' - x_R' = \frac{bf}{z} \quad \boxed{b \propto d}$$

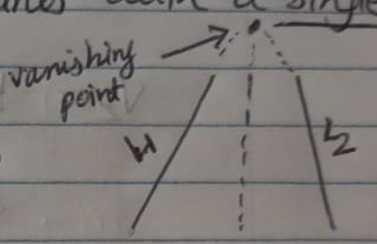
* Therefore, increasing baseline ($B \rightarrow B + \Delta B$) should lead to computation of values of depth (z) with higher accuracy.

* Baseline is serving as a magnification factor for depth resolution but at the cost of higher probability of a false match because of a larger disparity.

Problem 4)

* We have a lane (set of 2 parallel lines) in world co-ordinate system.

* These 2 lanes lines when represented in the image co-ordinate system, represent 2 lines with a single vanishing point.



$$\text{Line 1: } a_1 u + b_1 v + c_1 = 0$$
$$\text{Line 2: } a_2 u + b_2 v + c_2 = 0$$

* The transformation of 3D points to the 2D image plane using a camera is given by:

$$\underbrace{\begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} f_x & s & p_x \\ & f_x & p_y \\ & & 1 \end{bmatrix}}_{K} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \ldots (I)$$

R (rotation) t (translation)

K (camera caliberation matrix)

$X \rightarrow$ world co-ordinate system

* Vanishing point in world-coordinate system:

$$Z_\infty = [0 \;\; 0 \;\; 1 \;\; 0]^T \quad \ldots (II)$$

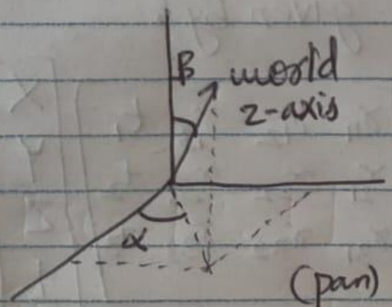
* Vanishing point in Image-coordinate system:

$$V_z = K [r_1 \; r_2 \; r_3 \,|\, t] \, Z_\infty \qquad \text{using (I)}$$

$$V_z = K [r_1 \; r_2 \; r_3 \,|\, t] \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \text{using (II)}$$

$$V_z = K r_3 \qquad \boxed{\therefore \; r_3 = \frac{K^{-1} V_z}{\| K^{-1} V_z \|}} \quad \ldots (\text{III})$$

Using a geometric interpretation:



$$\therefore \; r_3 = \begin{bmatrix} \sin\alpha \sin\beta \\ \cos\beta \\ \cos\alpha \sin\beta \end{bmatrix} \quad \ldots (\text{IV})$$

(pan) $\alpha = \tan^{-1}(r_3(1) / r_3(3))$

Camera axis (tilt) $\beta = \cos^{-1}(r_3(2))$

We also know :

Vanishing pt. for 2 lines : in image

$$\boxed{u = \frac{b_1 c_2 - b_2 c_1}{a_2 b_1 - a_1 b_2} \; ; \; v = \frac{c_1 a_2 - c_2 a_1}{a_1 b_2 - a_2 b_1}} \quad \ldots (\text{V})$$

Reference for (I), (III) & (IV): Multiple View Geometry in *Computer Vision* Second Edition. Richard Hartley and Andrew *Zisserman*

* We know the camera caliberation matrix K.
We also know V_Z (vanishing point in image).

So Using (III), we can get r_3 matrix.

* This inturn can give us the pan angle (α) & tilt angle (β) by using (IV).

* Therefore, we can estimate α & β using the information given in the question.
We cannot estimate yaw angle with only this vanishing point.

Therefore, from the information provided in the question we can only estimate the pan and tilt angle (or a single column in the rotation matrix). Position cannot be estimated (since we do not have 4 points to compute Homography matrix), and complete orientation can also not be estimated (since we do not have 2 sets of orthogonal parallel lines leading to 2 vanishing points).

Below are 2 additional scenarios where we can compute additional information regarding R & t:

I) If we are provided 2 sets if vanishing points in the image, we can estimate the whole rotation matrix in the below fashion:

* In order to get all 3 matrices, we need 2 sets of parallel lines representing 2 vanishing points which are orthogonal:

$r_3 = K^{-1} Z V_Z$ .... (first vanishing point info)

$r_1 = K^{-1} Z V_X$ .... (second vanishing point info)

$\boxed{r_2 = r_3 \times r_1}$ .... (third rotation matrix)

* We have recovered orientation of the camera.

II) If we are provided with 4 corresponding pairs of points in world and image co-ordinate system, we can estimate the whole rotation matrix and translation matrix in the below fashion (see next page):

* For position (translation) for the camera, we can orient our world co-ordinate system such that

$$X = [X \; Y \; 0 \; 1]$$

∴ Let m be an image point which can be written as:

$$m = [u \; v \; 1]^T$$

$$m = K[r_1 \; r_2 \; r_3 \mid t] X$$

$$m = K[r_1 \; r_2 \; r_3 \mid t] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \underbrace{K[r_1 \; r_2 \; t]}_{\text{2D Homography matrix}} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\text{Pose} = [R \; t] \qquad \tilde{H} = K[r_1 \; r_2 \; t]$$

* $\tilde{H}$ can be estimated by taking 4 points in physical ground plane and their corresponding image point.
* $\tilde{H}$ can be the used to get t & r_1, r_2.

$$H = K^{-1}\tilde{H} = [r_1 \; r_2 \; t] \qquad \text{Note: } \|r_1\| = \|r_2\| = 1$$

using normalization factor $a = \|(H_{11}, H_{21}, H_{31})\|$

we can then get: $\xrightarrow{}$ third column of H

$$t = H(:,3)/a$$
$$r_1 = H(:,1)/a$$
$$r_2 = H(:,2)/a$$

$$\boxed{r_3 = r_1 \times r_2}$$

∴ We have estimated rotation & translation of the camera from the projection matrix.