Sharif University of Technology

Computer Vision

HW03:
Vanishing points , Epipolar Geometry , 3D reconstruction , BOW

Author:
Saeed Razavi (98106542)

May 29, 2022

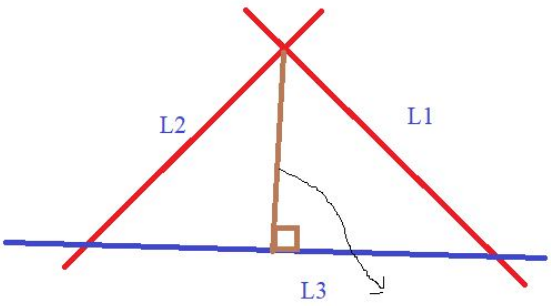
question1(Vanishing points)

1.For the buildings image, we have estimate the positions of the three orthogonal vanishing points . we manually choose some lines corresponding to the building orientations. we use 4 manually selected lines to find vanishing point for each orientation . below you can see the selected lines :

(a) original image (b) orthogonal lines

Figure 1

after selecting lines manually , we have to find "best" intersecting points for lines in each axis .to do so , we take a combination of 2 lines and then find their intersection point. This intersection point can be the vanishing point as it is expected that the lines converge near the vanishing point. But the problem here is that there are many Permutation of lines, so the number of such intersections of two lines can result in many different points. So to find the actual vanishing point we define a parameter for “error” i. The error is basically the square root of the sum of squares of the distance of this point from each line. The point corresponding to the minimum error value will possibly be the closest to the vanishing point and hence it’s our ideal Vanishing Point. Below is the image demonstrating the error calculation between the intersection of 2 lines and another line :



e3(1,2)= Error for L3 , cuased by L1 and L2 intersection
(distance metric : L2 norm)

Figure 2: intersection error line (L3)

so the goal is to minimize the error for line of each axis to reach vanishing points . to say it mathematically :

$$best\ answer\ =\ \min_{j_1 j_2}\ \sum_{i=1}^{n} e_i(j1, j2)$$
$$where\ \ (j1, j2\ , i\ \subseteq permutation\ of\ lines)$$

Figure 3: Method to find vanishing points

$x\_axis\ vanishing\ point$: $(8443.942,\ 2408.5503)$

$y\_axis\ vanishing\ point$: $(-16260.728, 3147.9487)$

$z\_axis\ vanishing\ point$: $(-928.3304, -86863.266)$

Figure 4: vanishing points

after finding vanishing points for different axis lines , we compute ground horizon line by cross product of two vectors from ground plane . we choose $V_x(vanishingpointofxaxis)$ and $V_y(vanishingpointofyaxis)$. so we reach the ground horizon line with the form $ax + by + c = 0$. at the end we normalize the vector . below you can see the formula derived for this line , also we plot this line on the image :

$$-0.029x + -0.999y + 2659.38596 = 0$$

Figure 5: line formula(L3)

Figure 6: ground horizon line on image

also we plot the image , vanishing points and ground horizon line in one plot ;

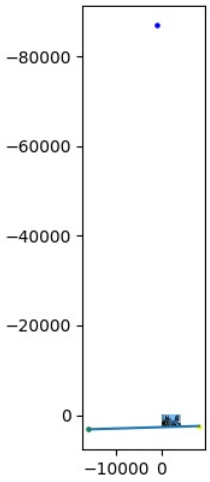


Figure 7: res02

section2(find p_x,p_y,f and rotation along z and x axis)

first of all we find p_x and p_y and focal length of camera and calibration matrix from following equations (2 finite vanishing points equations) :

$$\begin{cases} eq1-eq3: & (a_1-a_3)p_x+(b_1-b_3)p_y=a_2(a_1-a_3)+b_2(b_1-b_3) \\ eq1-eq2: & (a_2-a_3)p_x+(b_2-b_3)p_y=a_1(a_2-a_3)+b_1(b_2-b_3) \end{cases}$$

$$eq1:\quad f^2=-p_x^2-p_y^2+(a_1+a_2)p_x+(b_1+b_2)p_y-(a_1a_2+b_1b_2)$$

$$K=\begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 8: equations

below you can see the principal point on the image . in addition the title of following image is the focal length :

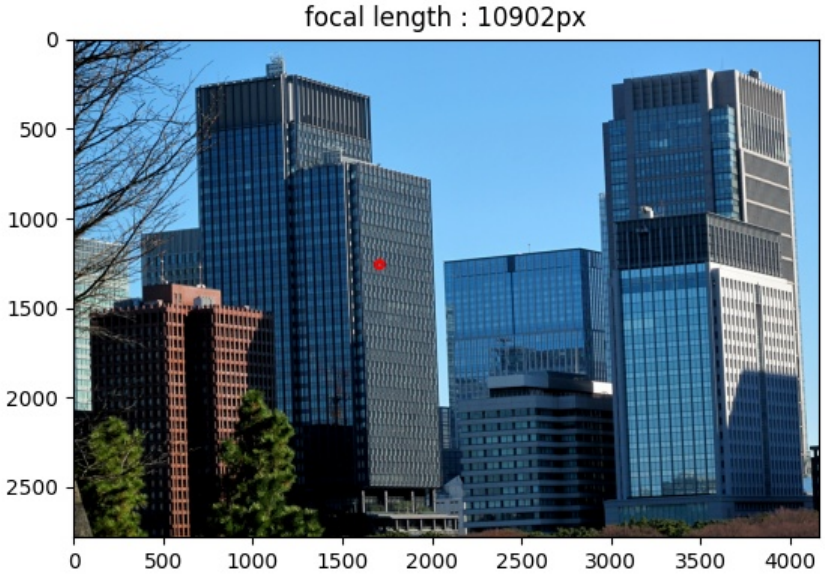


Figure 9: principal point and focal length

to find orientation along z axis we first back-project v_x,v_y to find x axis and y axis direction of camera . then we calculate cross product of these to rays to find z axis of camera . after that we can find degrees between camera axis and world standard axis .
z-axis rotation : we have to image the line on the x-y plane and find the angle between the line and x-y plane by the formula : $\arctan(\frac{y_0}{x_0}) + \pi/2 \quad radian$
x-axis rotation : we have to find angle between the line and z axis we can reach this by :
$\arctan(\frac{z_0}{magnitude \quad of \quad thevector})$
the z and x rotation is as below :

principal point : (1709,1261)

focal length : 10902px

calibratin matrix : $\begin{pmatrix} 1.09e4 & 0 & 1.7e+03 \\ 0 & 1.09e4 & 1.26e3 \\ 0 & 0 & 1 \end{pmatrix}$

z_rotation : 1.71 deg

x_rotation : 7.05 deg

Figure 10: results

section3(warp the image with rotations along X and Z axis)

we know from previous home works that the relation between pixels of one image from two different view is as below :
$x_{cam\ 2} = K_{cam\ 2}[R_{cam\ 2}|t_{cam\ 2}][R_{cam\ 1}|t_{cam\ 1}]^{-1}K^{-1}_{cam\ 1}x_{cam\ 1}$,
if we consider that :
1) the first camera is the world camera (has no translation and rotation)
2) second camera has just rotation
3) both cameras have same calibration matrix we reach simpler equation :

$$\boxed{x_{cam\ 2} = K[R]K^{-1}x_{cam\ 1}}$$

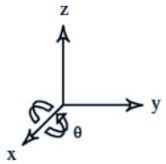
so the new homography matrix equation is as below :

$$\boxed{H_{new} = K[R]K^{-1}\ ,}$$

but the question is , how to reach the rotation matrix ?
Rotation matrix can be reached by following formula :
$Rotation\ matrix = R = R_xR_yR_z$
where :

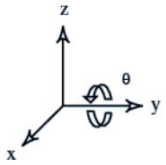
Rotation around the x-Axis

$$Rx = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$



Rotation around the y-Axis

$$Ry = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$



Rotation around the z-Axis

$$Rz = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

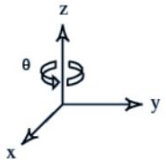


Figure 11: rotation

below you can see the result of warping image with new homography matrix :

(a) original image

(b) warped image

Figure 12

question2(Epipolar Geometry)

in this question we use set of matched points provided by sift to estimate the fundamental matrix F . we use $cv2.detectAndCompute$ to find the corners and then use $cv2.BFMatcher$ with knn to match the corners . after that we use $cv2.findFundamentalMat$ to find the fundamental matrix. below you can see the matrix :

$$\text{fundamental matrix} : \begin{pmatrix} 8.86\text{e}-9 & -7.98\text{e}-8 & -1.29\text{e}-4 \\ -4.4\text{e}-8 & 4.9\text{e}-9 & -1.35e-3 \\ -2.27\text{e}-4 & 1.59\text{e}-3 & 1 \end{pmatrix}$$

Figure 13: rotation

also this function provide the mask that shows which matched points are inlier and which ones are not using the fact that $X'tFX = 0$ below you can see the inlier and outlier points of two images :

Figure 14: inlier and outlier points

now , we want to find epipole points of two images using fact that : $Fe = 0$ and $f^t e^{'} = 0$ .because these equation have the form AX=B , we can find e and $e^{'}$ using svd decomposition , the last column of V is the answer .

below you can see the results :

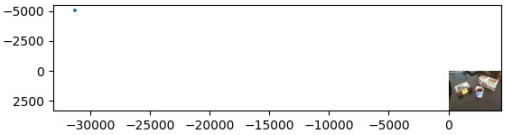


Figure 15: epipole point

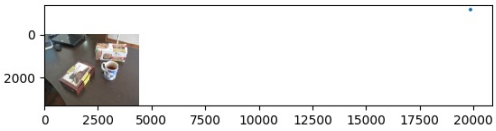


Figure 16: epipole point

at the end , we choose 10 matched points randomly and plot them on the image with their epipolar lines . to find lines we use two following equations .

$$L^{'} = Fx \ , L^{=} F^{t} x^{'}$$

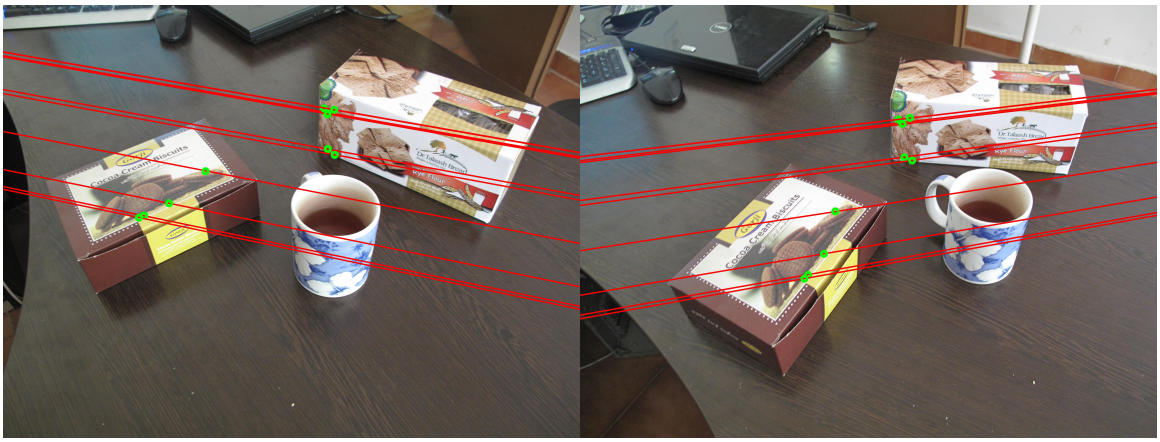


Figure 17: epipole lines

question3(3D reconstruction)

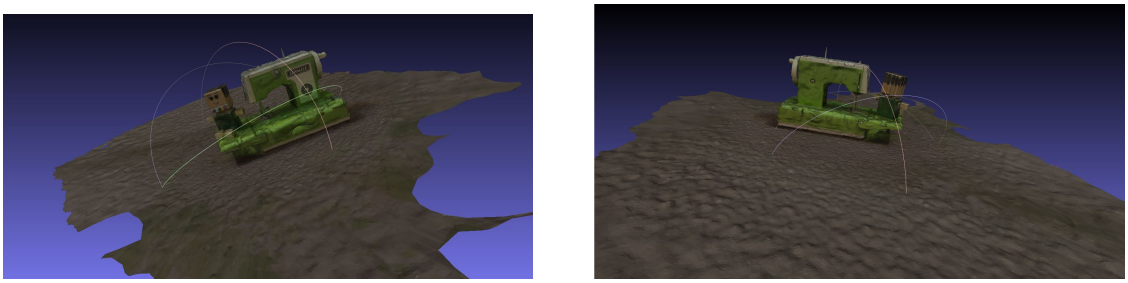
in this question we use meshroom to reconstruct the image . you can see the original images , texturing, finding point cloud and the final result below :

(a) original image (b) original image (c) original image

Figure 18

(a) texturing (b) structure from motion

Figure 19



(a) reconstructed image (b) reconstructed image

Figure 20

question4(Scene Recognition Using Bag of Words)

part A) in this section , we make a feature vector space from gray scale images. to do this we downsize each image into [8,8] image and then make the gray scale value a vector with size [1,64] (i tried [16,16] splitting too but [8,8] has better results) .then with using nearest neighbor , we find the best match for each test image with comparing its feature vector with all other feature vector in feature vector space . for matching i use knn from skitleran library . the funny fact about this section is that when i choose k=1 whit metric distance (L1 : manhattan) i get the accuracy about 25.3 but for example for k=3 i got 23.1 and for k=11 i get 21.5
below you can see the parameters and result :

$$\begin{aligned}&\text{best acc } = 25.13\,\%\\&\text{downsized size } = (8,8)\\&\text{distance metric : L1 norm}\\&\text{k in knn } = 1\,!\end{aligned}$$

Figure 21: parameters

part B) in this section we make a dictionary from features using k-means. i use from k-means implementation from *sklearn* library . to make a dictionary we use a [n,128] feature vector for each train image using sift from *cv*2. we add all of the features to the feature space . after adding all , we use k means to find 100 clusters with specific centers . this 100 centers are our new dictionary . after making dictionary , we make a histogram from each image based on this 100 features and normalize these histogram to reach pdf(probability distribution function) . then for each image in test set , we again find its histogram and compare that histogram with all train set histogram and using Knn to predict the label of test image . the best accuracy that i get is . below you can see the parameters : 37.26

$$\begin{aligned}&\text{best acc } = 37.2.\,\%\\&\text{number of clusters } = 100\\&\text{k in knn } = 15\end{aligned}$$

Figure 22: parameters

part C) this part is like the previous part. the only difference is that instead of using knn to predict the test images , we use SVM from sklearn library . the accuracy becomes better with this method and its around 51.7 . below you can see the parameters and confusion matrix :

$$\begin{aligned}&\text{best acc} = 51.7\ \%\\&\text{number of clusters } = 100\\&\text{distnace mertic } = \text{L1 norm}\\&\text{kernel } = \text{gamma}\end{aligned}$$

Figure 23: parameters

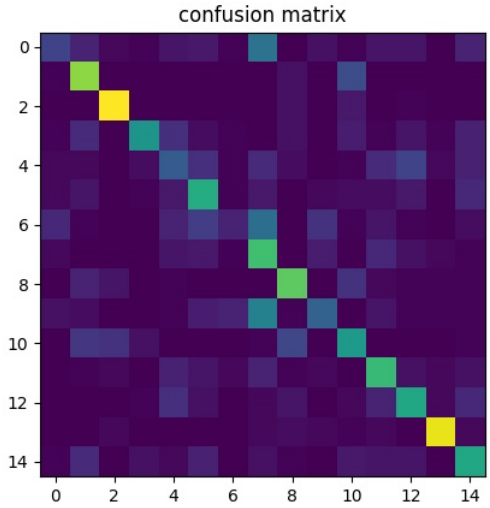


Figure 24: parameters