CS376: Computer Vision: Assignment 3 Due: Oct. 31st, 11:59 PM

**Instruction:** 100 points in total.

**Submission:** See the end of this document for submission instructions.

# Programming: Camera Calibration (50 points)

In this problem, we are interested in solving the camera calibration problem, which determines the intrinsic parameters of a camera by using an image of a rig (Calibration.jpg). We will keep this question open-minded where we do not specify the keypoints on the rig. Instead, you are required to pick the key points by yourself. Include the 2D pixel coordinates and 3D coordinates of the feature points you picked. The 2D pixel coordi- nates should be stored in a Matlab matrix *ICoord*2*dI* of dimension 2 *N* , where *N* is the number of keypoints. The 3D coordinates of the feature points should be stored in a Matlab *ICoord*3*dI* matrix of dimension 3 *N* . Include the following function for estimating the intrinsic camera parameter:

×

×

[K] = cameracali(Coord2d, Coord3d)

where ’Coord2d’ and ’Coord3d’ are the 2D and 3D coordinates of the keypoints you picked. *K* R3*×*3 is an upper-right matrix that encodes the intrinsic camera parameters.

∈

In your writeup analyze the following:

* + (5 pts) Explain how you pick the 2D keypoints. You can use the function ’ginput’ to pick these feature points. Note that depending on the resolution of your screen, you may have to rescale your image, pick the keypoints, and then scale it back. It is also recommended to ues Matlab’s SURF feature detection to pick the strongest corner features for the 2D keypoints on the calibration image., and then snap the keypoints you picked onto these detected keypoints. The ’snapping’ procedure could be simply performing nearest neighbor search.

Using the ‘ginput’ function 6 times, I click on 6 points on the box in the provided calibration image, picking corners of the black/white tiles on it on all visible faces of the box. The 2d coordinates are stored in a matrix.

* + (5 pts) Explain how you compute the corresponding 3D keypoints. To this end, it is recommended that you pick a 3D coordinate system, which is usually aligned with the axses of the cube box, and which the origin is located at one of the corners of the cube. Then you can count the *x*,*y* and *z* coordinates of the keypoints.

Hint: You can compute the corresponding 3d points by assigning xyz axis to the checkerboard box and counting the number of units for the coordinates that each of the point lies on the box’s coordinate space. I used the box corner closest to the camera as the origin.

To get the corresponding 3d points, I picked the bottom corner of the closest edge of the box as the origin of the 3d space. I decided to make the edge going right the x-axis, the edge going left the y-axis, and the edge going upward the z-axis. I decided to make each tile on the box have the dimensions of 1x1 units in this 3d space. I then counted the squares along each axis towards the 2d keypoints to get the 3d keypoints, and stored those coordinates in another matrix.

* + (20 pts) The first step is to estimate the matrix Π = [*KR, KT* ], where *K* R3*×*3 is the upright matrix that encodes intrinsic camera parameters, and *R SO*(3) and *T* R3 encode the extrinsic camera

∈ ∈

∈

parameters. Note that you will get two matrices from the smallest eigenvector computation, you can eliminate one by enforcing that each *λi* has to be positive in the following two constraints:

*λi****x****i* = Π***X****i,* (1)

where ***x****i* = (*x*2*d, y*2*d,* 1)*T* ∈ R3 is the homogeneous coordinate of the 2D pixel coordinate of the *i*-th

*i*

*i*

keypoint, and ***X****i* = (*x*3*d, y*3*d, z*3*d,* 1)*T* ∈ R4 is the homogeneous coordinate of the 3D coordinate of the

*i*

*i*

*i*

*i*-th keypoint. You can use the sign of

*T*

L ***x*** Π***X****i*

*i .*

***x****T* ***x****i*

*i i*

Pi Matrix:

*Table

Description automatically generated*

* + (15 pts) The second step is to estimate *K*, *R* and *T* from matrix Π. This step would involve QR decom- position as well as other operations. For QR decomposition, please refer to ’CS376 Lecture 15 note.pdf’ on canvas. Compute the determinant of your estimated *R*. You should put your estimated *K*, *R*, *T* and det(*R*) in the PDF writeup in order to earn credits.

K:

Table

Description automatically generated

R:

Graphical user interface, table

Description automatically generated

T:

Table

Description automatically generated

det(R):

A picture containing box and whisker chart

Description automatically generated

* + (5 pts) Test your program with different configurations of 2D and 3D keypoints. Compare the resulting intrinsic and extrinsic parameters. Answer the question where to pick keypoints for robust estimation of intrinsic and extrinsic camera parameters.

Keypoints should span a variety of distances in all three axes. Have at least one, but ideally more, that have nonzero values in each axis individually with zeros in the other two, points that have nonzero values in two of the three axes in all combinations, as well as points with nonzero values in all three axes. In practice, this means several points on all three visible faces of the calibration box.

* + **Extra credit (5 pts).** Suppose you have marked 20 keypoints, and your goal is to select 10 of them to estimate the camera parameters. Write a program to output the indices of these 10 keypoints. Hint: You will need to consider a statistical model for the pixel and 3D coordinates of the feature points. Then you need to write out the matrix Π as a function of the perturbations of the coordinates.
  + **Extra credit (5 pts).** So far we have talked about using points for image calibration. Propose a strategy that uses lines for calibration, i.e., correspondences between lines in the input image and 3D lines.

Ask the user to provide the dimensions of the calibration box (x, y, z). Then use edge detection to get the edges of the box or simply ask the user to pick the endpoints, i.e. the corners of the box to provide the edges representing the x, y, and z axis. For the 2d lines, subtract one endpoint’s 2d coordinates from the other endpoint’s coordinates to get a 2d vector representing that line. Use the dimensions of the box to determine the corresponding 3d vector. Plug these values in place of the 2d pixel coordinates and the 3d key point coordinates.

# Programming: Structure-From-Motion (50 points)

In this problem, we are interested in estimating the relative camera pose between two images of the same underlying 3D scene (SourceImage.jpg and TargetImage.jpg). Our goal is to leverage the intrinsic camera parameters estimated in Problem [1](#_bookmark0) Camera Calibration, turning this into a calibrated two-view structure- from-motion problem. We will use the so-called eight-point method. Similar to the previous problem, you are asked to mark feature correspondences between two images (you can do this manully or using an automatic algorithm). Note that in the required part, the marked correspondences are supposed to be compatible with each other. In the extra credit part, you may run RANSAC to pick consistent correspondences across a pair of images.

Again, please submit the two sets of feature correspondences you pick. The first set of feature points will be given by ’sCoord2D’, and the second set of feature points will be given by ’tCoord2D’. Both sets of feature correspondences have *n* 8 keypoints. The same as above, you can first mark the keypoints, and then snap them onto the detected corners.

≥

Now given the feature correspondences, and the intrinsic camera parameters *K* estimated in Problem [1.](#_bookmark0) Our goal is to estimate the rigid transformation (*R* R3*×*3*, T* R3) that relates the second image and the first

∈ ∈

image:

[*R, T* ] = relativepose(sCoord2D, tCoord2D, K) In your writeup analyze the following:

* + (10 pts) Please mention how to pick the feature correspondences. You need to provide two visual- izations, one for the source image, and another for the target image. You can either color-code the corresponding keypoints, or you can directly draw correspondences across the input images.
    - Features that are clearly visible in both images with matching similarities around them. They should be spaced out and in differing depths.

I picked feature correspondences by using a side-by-side comparison of the two images to identify objects in the 3d world and using ginput to click on a feature of those objects in both the source and target. The object must be the same. Similar looking objects in the image must be carefully paid attention to, e.g. we have to make sure the desk in one image is the same exact desk in the other image—not just an identical looking desk.

A picture containing indoor, ceiling, device, area

Description automatically generated

My selected keypoints, source on left, target on right.

A picture containing ceiling, indoor, area, appliance

Description automatically generated

Table

Description automatically generated

Source 2D (pixel) coordinates:

Target 2D (pixel) coordinates

Table

Description automatically generated

Table

Description automatically generatedTable

Description automatically generatedSource 3D coordinates:

Target 3D coordinates:

* + (20 pts) Use the provided intrinsic camera parameter matrix *K* and the corresponding keypoints to solve for the essential matrix *E*. Note that you will have two potential solutions, where one is the negation of the other.
    - Done with estimation.

Estimated essential matrix:

Table

Description automatically generated

* + (15 pts) Extract the rotation *R* and the translation *T* from each resulting essential matrix *E*. Use the sign of the induced depth for each keypoint to eliminate implausible essential matrices and the associated rotations and translations. Compute the determinant of *R*. You should put *R*, *T* and det(*R*) in the PDF writeup in order to earn credits.

Extracted R:

Table

Description automatically generated

Extracted T:

Table

Description automatically generated

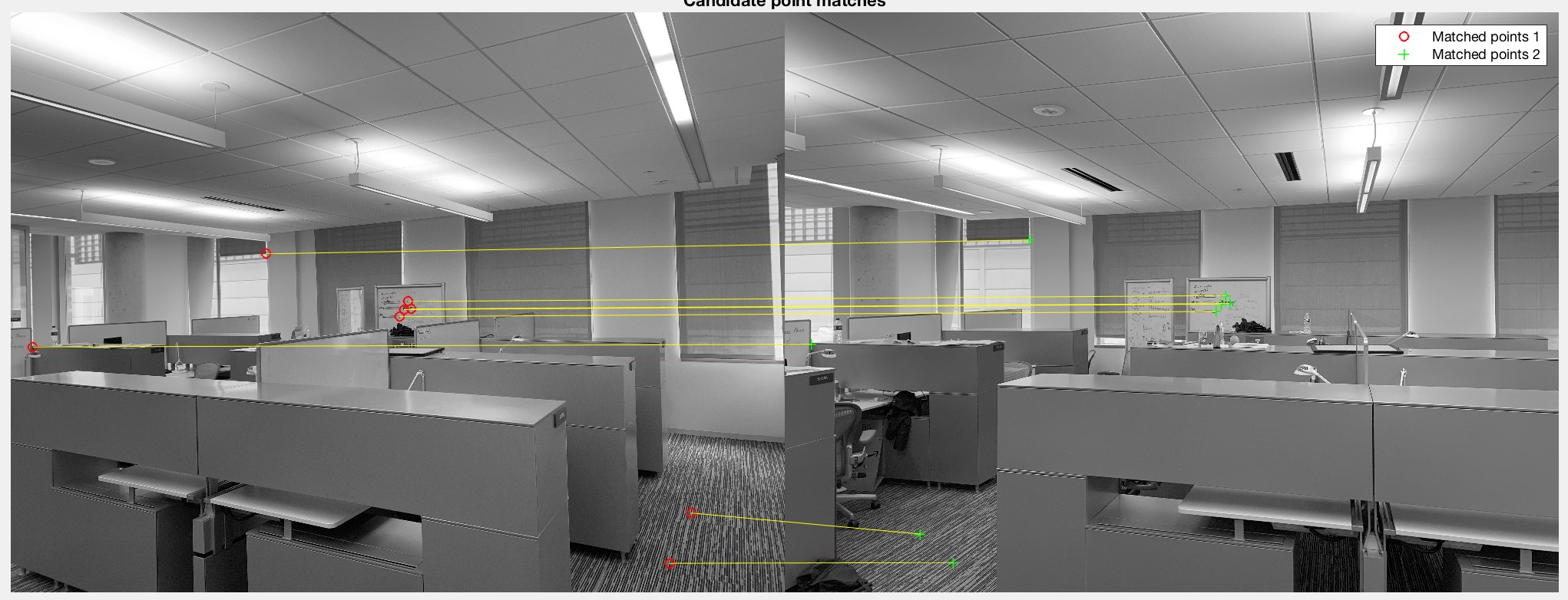
Determinant of R:



* + (5 pts) Please play with five sets of correspondences and compare the resulting relative transformations. Discuss which set of feature correspondences lead to potentially more accurate relative pose estimations.

After playing with different sets of points, it seems that there are a few guidelines that lead to more accuracy:

* A variety of keypoints on spaced out objects in the 3D world is beneficial.
* Identifying the objects in the 3d space with the most dramatic difference in pixel locations in the 2d images is good, but works much better when they are keyed in tandem with objects in the 3d space that undergo the least shift in 2d pixel locations. This seems to dramatically increase the accuracy of the extracted rotation.
* Increasing the amount of keypoint pairs helps if the new keypoints identify new objects in different locations, otherwise they don’t help much—i.e. “clustering” several keypoints in a small area doesn’t seem to improve accuracy.
* Imprecision in keypoint placement between the source and target obviously reduces accuracy.
* Picking strongest features of the 3d world to mark is ideal, because as an extension of the previous point, the stronger the feature, the less likely there is to be imprecision, whether caused by human error or otherwise.
  + **Extra credit (5 pts).** Instead of using manually marked feature correspondences, please run RANSAC to extract consistent correspondences across the two input images.
    - Done and commented out of the way. Proof below



# Submission instructions:

Create a single **zip** file and submit on Canvas that includes

* + Your well-commented code, including the files and functions named as specified above.
  + A **PDF** writeup of your results with embedded figures where relevant. Please do not include any saved matrices or images etc. within your zip file.