CS-512: Computer Vision – FALL 2018

ASSIGMENT 5

Diego Martin Crespo

A20432558

1. PROBLEM STATEMENT

The problem chosen to solve in this assignment is coplanar calibration.

1. WRITE A PROGRAM TO EXTRACT FEATURE POINTS FROM THE CALIBRATION TARGET AND SHOW THEM ON THE IMAGE USING OPENCV FUNCTIONS.

2. WRITE A PROGRAM THAT DOES THE CALIBRATION PROCESS FROM 3 TEXT FILES WITH 2D POINTS AND ITS CORRESPONDENT 3D POINTS. THE PROGRAM SHOULD DISPLAY THE INTRINSIC AND EXTRINSIC PARAMETERS AS WELL AS THE MENA SQUARE ERROR BETWEEN THE KNOWN AND THE COMPUTED POSITION OF THE IMAGE POINTS.

3. IMPLEMENT THE RANSAC ALGORITHM FOR ROBUST ESTIMATION.

2. PROPOSED SOLUTION

2.1. EXTRACTION OF 2D AND 3D POINTS

Using the functions suggested by the professor it is possible to extract the points of each image passed as argument and they are saved into different files (world and image points). Also the image given as argument will be shown with the draw chess corners.

2.2. CAMERA CALIBRATION: COPLANAR

As the problem chosen is coplanar, for the camera calibration process, the program takes 4 arguments:

- The first argument (0) is the name of the program (hw5_2.py).
- From 1 to 3 correspond to three 2d-image points files.
- The fourth is the file with the 3d coordinates points.

These files taken as arguments have been generated by the first program. Also, files from the calibration data repository that the professor gave as have been use in order to verify the proper work of the program.

The program will calculate the intrinsic and extrinsic parameters of each image (using the files with 2d and 3d points). After the MSE for each image is computed comparing the calculated projection points with the know ones.

2.3. RANSAC

For the RANSAC algorithm, which is implemented in the same program as the camera calibration for each of the three images, we have to do the next k times:

1. Compute the matrix M with n random points.
2. Compute the estimated image points using the matrix M.
3. Compute the distance between the 2D estimated points and the real ones.
4. Compute the t as 1.5*median of the values in the step 3.
5. Find all inliers in the data with distances smaller than t.
6. Recompute matrix M with all the inliers.
7. Compute MSE for all the points using the matrix M in step 6.

To compute the matrix M and the distance in all these loops I use the functions that I created for the second part.

After doing this k times we have some matrices M with the corresponding MSE of each of them. We choose the one with the smaller MSE to be the best one and its corresponding matrix M and its parameters.

3. IMPLEMENTATION DETAILS AND PROBLEMS

The implementation of the first part was easier as there where some examples pretty similar in internet. I had several problems with saving the points into files which gave problems in the second program. First, I was saving the points as rows of arrays, so then the extraction at program 2 was a bit tedious. Then I changed so the points were saved as "X,Y" per row. By using panda's library at program 2 it was pretty straight forward to manage the points pretty easy once they are converted from de data frame to array form.

For the camera calibration process I had many problems:

- Dimensions of arrays and operations. Using numpy.squeeze(), numpy.expand_dim(), numpy.asarray() etc. I managed to give the data the proper form.
- Generating the H vectors, I was taking the second return argument of the svd function. Then I realized it was the last column of the third argument the correct one.
- Generation the V12, V11, and V22 matrix. I was not ordering right the values of the H into the Vii matrixes.
- Generation of the R|T matrix. I was not ordering right the R values.
- I the error calculation function. I was first subtracting the projection of the 3DH point and the 3DH itself. Then I realized it was the 2D points instead of the 3DH logically.

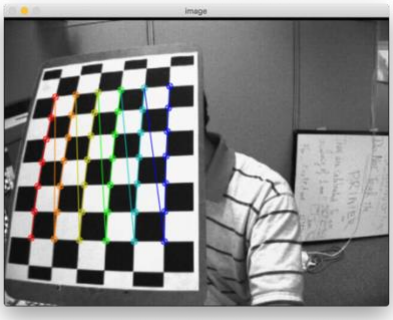
For the implementation of the RANSAC algorithm I followed the steps detailed in the proposed solution. I had some problems with dimensions of arrays and operations. The parameters n (number of points), d (threshold of inliers) and k (times the program executes to find the smaller MSE) are loaded for the RANSAC.config file. Therefore, in order to modify them this file should be modified.

4. RESULTS AND DISCUSSION

4.1. PROGRAM 1:

With the first program we can calculate the 2D and 3D points for each image given as argument.

These are the shown images with the draw chess corners for 3 images given as arguments (one at a time) to the program:



For each image given if the name was “image.jpg” the program generates 2 files:

- At folder “../data/2d” it saves “image.txt” with the 2d points.
- At folder “../data/3d” it saves “word.txt” with the 3d points.

4.2. PROGRAM 2:

CAMERA CALIBRATION

First the program is executed with the files provided by the professor to get the right results. We got the same almost the same results as the ones specified in the repository (with 3 decimal difference):

If we execute the program with the previous chosen images we got

ERROR ESTIMATION

Using the files at the repository provided by the professor be got the following errors for each image:

- Image0 Error: 1.7972878174377108e-09
- Image1 Error: 1.6597041126765092e-09
- Image2 Error: 1.3802549910448174e-09

As you can see we got a pretty low error, almost none. Therefore, we made a good estimation.

However, if we use the files generated with the test images we got higher values of error:

- Image0 Error: 0.8036525836456981

- Image1 Error: 1.7448488759040421
- Image2 Error: 0.015759371710744978

This higher error is normal due to the noise of the different images.

RANSAC ESTIMATION

We got the following results for the repository files with n=16, d=6 and k=3:

- MSE0 Image0 = 0.008154903713573015
- MSE1 Image1 = 0.008154240787836844
- MSE2 Image2 = 0.00814980428464886

However, if we use the files generated with the test images we got with n=30, d=6 and k=3:

- MSE0 Image0 = 7.491915365450244
- MSE1 Image1 = 6.687885453394721
- MSE2 Image2 = 8.89691684704164

These results may variate as a random variable is used in the program. And sometimes it gives an error so it may be something wrong programed or the parameters won fit well to find the model

5. REFERENCES

https://docs.opencv.org/3.0-beta/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

http://www.cs.iit.edu/~agam/cs512/data/calibration/index.html

https://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

https://docs.opencv.org/3.0-beta/doc/tutorials/features2d/trackingmotion/corner_subpixeles/corner_subpixeles.html

https://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html

https://www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/

http://ais.informatik.uni-freiburg.de/teaching/ws09/robotics2/pdfs/rob2-08-camera-calibration.pdf

http://www.cs.iit.edu/~agam/cs512/share/Zhang.pdf

http://www.cs.iit.edu/~agam/cs512/share/c11.pdf

https://www.cc.gatech.edu/~hays/compvision/proj3/

https://docs.scipy.org/doc/numpy-1.15.0/index.html

https://en.wikipedia.org/wiki/Random_sample_consensus