Hands-on exercises

Question 1.

(a) Based on the given information, the projection matrix is given by:

Assumption: skew = 0

$$P = K_{3\times3}\,[\,R \mid t\,]_{(3\times4)} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} [\,R_{3\times3} \mid t_{3\times1}]$$

$$P = \begin{bmatrix} 480 & 0 & 320 \\ 0 & 480 & 270 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5363 & -0.8440 & 0 & -451.2459 \\ 0.8440 & 0.5363 & 0 & 257.0322 \\ 0 & 0 & 1 & 400 \end{bmatrix}$$

$$P = \begin{bmatrix} 257.42 & -405.12 & 320 & -88598.032 \\ 405.12 & 257.42 & 270 & 231375.456 \\ 0 & 0 & 1 & 400 \end{bmatrix}$$

(b) In total, the projection of a 3D point onto the screen is performed in 3 stages:

1. The points is translated and rotated into the camera coordinates using Rotation + translation matrix
2. The point is translated into the homogeneous coordinates
3. The homogenous coordinates are being normalized into the regular pixel position coordinates

Solving each step separately:

1. Convert the 3D point into Camera coordinates

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [R \mid t\,] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5363 & -0.8440 & 0 & -451.2459 \\ 0.8440 & 0.5363 & 0 & 257.0322 \\ 0 & 0 & 1 & 400 \end{bmatrix} \begin{bmatrix} 350 \\ -250 \\ -35 \\ 1 \end{bmatrix} = \begin{bmatrix} -52.54 \\ 418.357 \\ 365 \end{bmatrix}$$

2. Get the homogenous coordinates:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = K_{3\times3} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} 91580.368 \\ 299361.456 \\ 365 \end{bmatrix}$$

3. Obtaining the normalized (real) coordinates:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\tilde{u}}{\tilde{w}} \\ \frac{\tilde{v}}{\tilde{w}} \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 250.9 \\ 820.17 \end{bmatrix}$$

(c) Re-Projection error – is defined as:

$$v = z - \pi(x, l)$$

In our case:

$$z = \begin{bmatrix} 241.5 \\ 169 \end{bmatrix}$$

$$\pi(x, l) = \begin{bmatrix} 250.9 \\ 820.17 \end{bmatrix}$$

Meaning:

$$v = \begin{bmatrix} 241.5 \\ 169 \end{bmatrix} - \begin{bmatrix} 250.9 \\ 820.17 \end{bmatrix} = \begin{bmatrix} -9.405 \\ -651.168 \end{bmatrix}$$

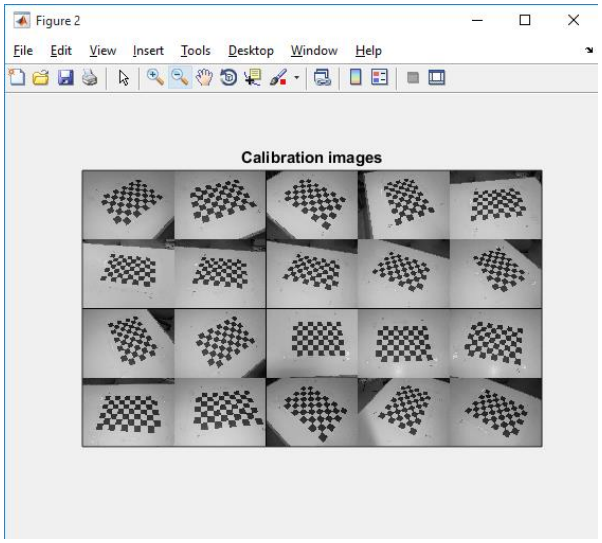
Question 2 – (c) - camera calibration

Here are presented the snapshots of the calibration process.

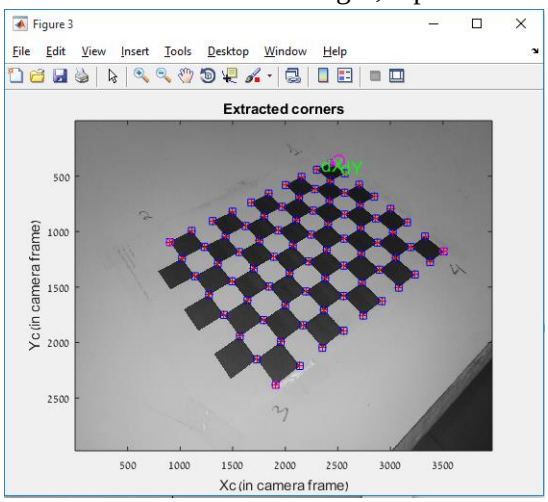
1. First, the images of the checkerboard plane have been taken from various camera poses. The camera used is the Huawei P9 dual-lenses camera. Thus, in order to keep the process authentic, one of the lenses was covered. Each of the corners was marked by a number for a more convenient calibration process.

Figure 1 : the camera used in the calibration process

2. Following the instructions, the images are being uploaded into the calibration program:



3. Corner extraction of the images, repeated on all the images



4. Calibration parameters are obtained:

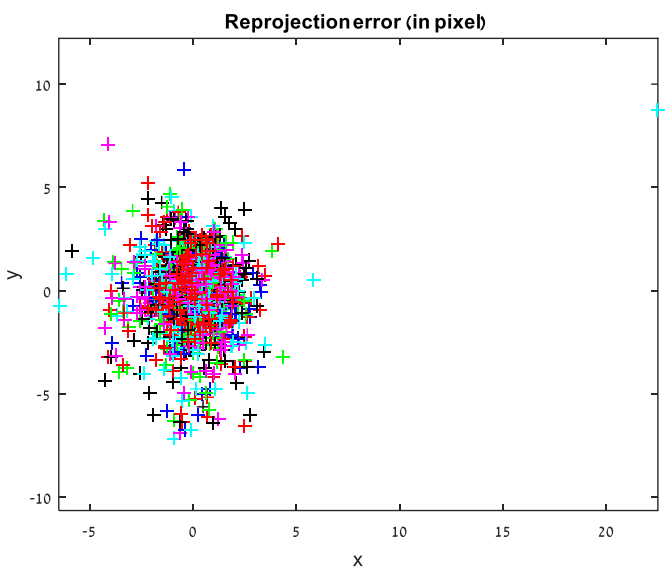
Focal Length: fc = [3189.24204 3189.24204]
Principal point: cc = [1983.50000 1487.50000]
Skew: alpha_c = [0.00000] => angle of pixel = 90.00000 degrees
Distortion: kc = [0.00000 0.00000 0.00000 0.00000 0.00000]

Main calibration optimization procedure - Number of images: 19
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...done
Estimation of uncertainties...done

Calibration results after optimization (with uncertainties):

Focal Length: fc = [3020.08805 3056.13647] +/- [39.75145 35.31200]
Principal point: cc = [1999.65339 1592.93877] +/- [9.42775 35.21215]
Skew: alpha_c = [0.00000] +/- [0.00000] => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion: kc = [0.10233 -0.31015 -0.00415 0.00271 0.00000] +/- [0.00874 0.03295 0.00096 0.00121 0.00000]
Pixel error: err = [1.41622 1.74597]

5. The reprojection error looks quiet large:



Our aim is to decrease it to the biggest scale.

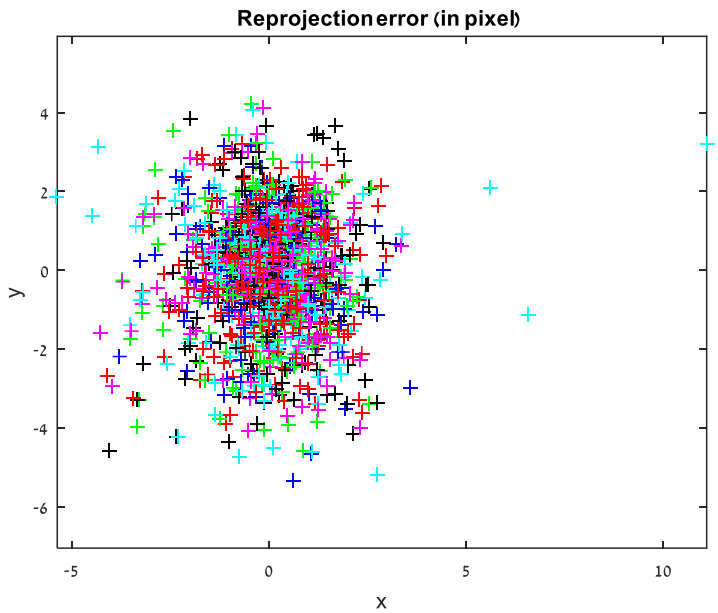
6. Using the Recomp.corners function provided in the GUI, a new calibration is performed, resulting in the following calibration results:

Calibration results after optimization (with uncertainties):

Focal Length: fc = [3043.25574 3075.15744] +/- [28.86633 26.23904]
Principal point: cc = [2000.49865 1569.94521] +/- [7.74148 24.72921]
Skew: alpha_c = [0.00000] +/- [0.00000] => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion: kc = [0.12149 -0.40301 -0.00409 0.00292 0.00000] +/- [0.00753 0.02930 0.00080 0.00097 0.00000]
Pixel error: err = [1.20546 1.47363]

It may be observed that the Pixel error has decreased in comparison to the first calibration results.

The reprojection error has decreased as well:

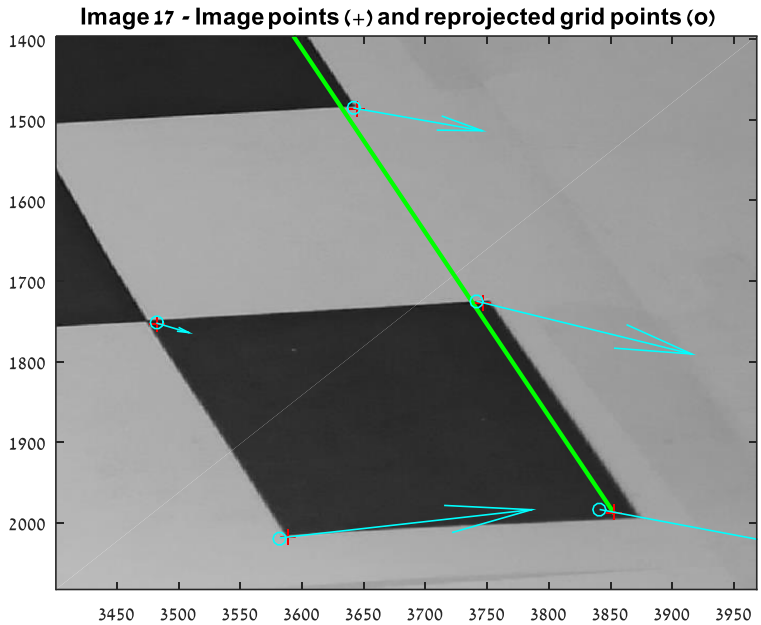


7. Analyzing some of the critical errors (in this example – the points in teal color in thereprojection error graph from step 6):

Pixel error: err = [1.20546 1.47363] (all active images)

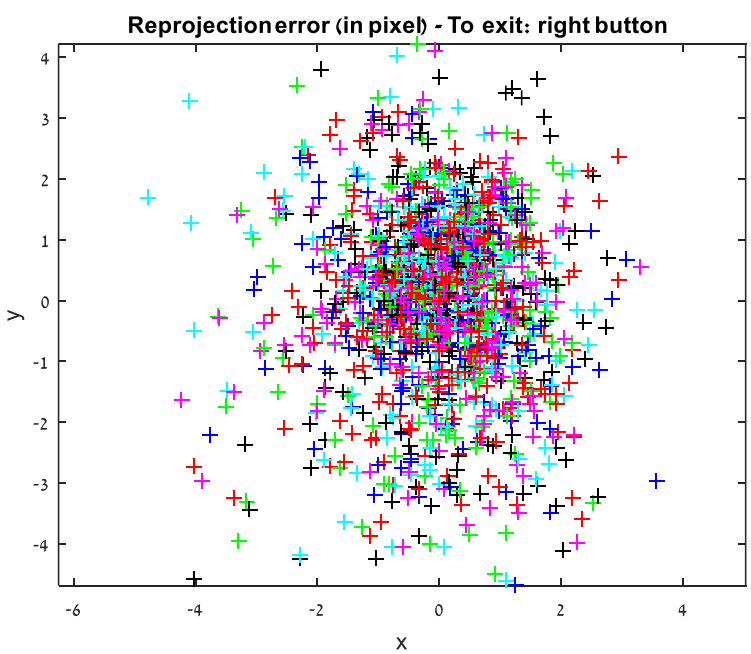
Selected image: 17
Selected point index: 11
Pattern coordinates (in units of (dX,dY)): (X,Y)=(0,6)
Image coordinates (in pixel): (3746.05,1726.11)
Pixel error = (5.61498,2.11331)
Window size: (wintx,winty) = (5,5)

As it may be seen, image 17 was not processed in a correct way, perhaps the corners were poorly extracted. By checking the image, it is validated:



Using a larger values for *wintx* and *winty* parameters should help to increase the ability to find the problem.

By using the values $\begin{cases} wintx = 8 \\ winty = 8 \end{cases}$, the following performance improvement is noticed:



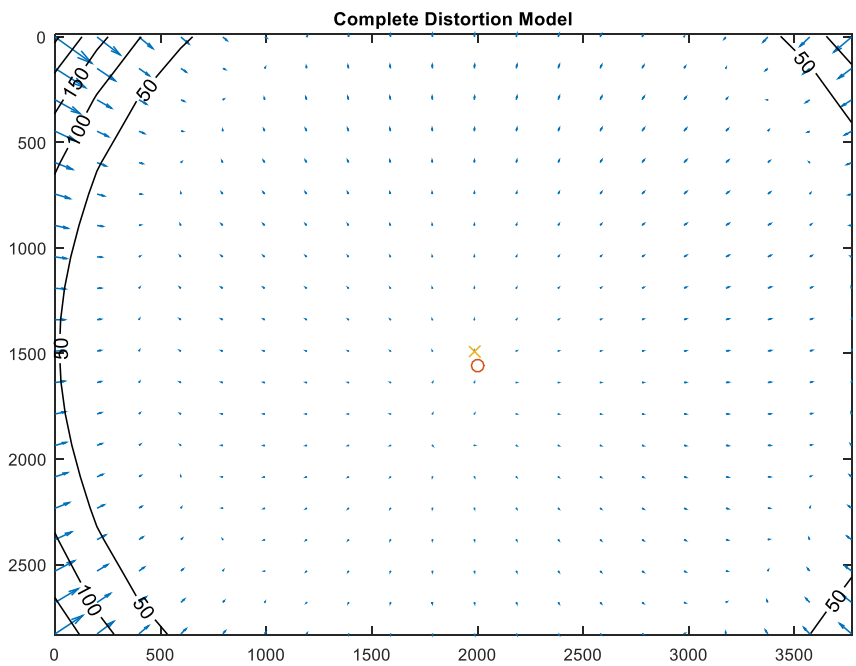
By looking at numerical representation of the calibration, it is obtained:

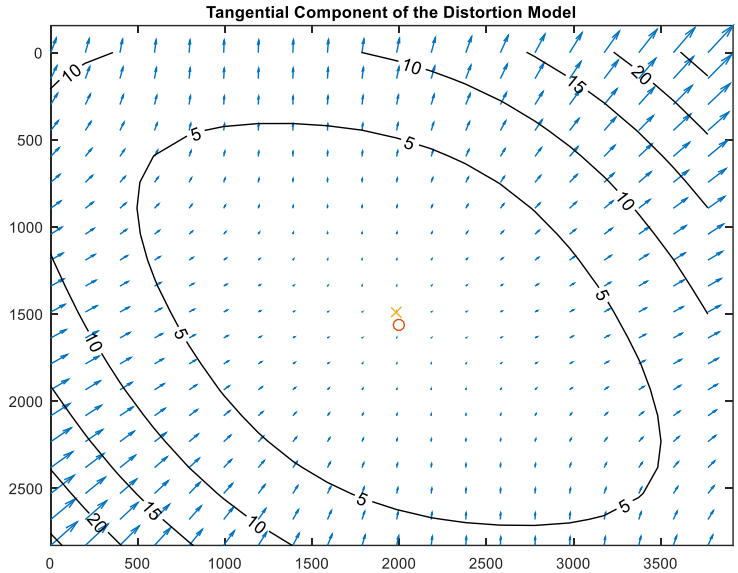
Calibration results after optimization (with uncertainties):

Focal Length: fc = [3064.80707 3093.11220] +/- [24.15153 22.46772]
Principal point: cc = [1999.02148 1548.76380] +/- [7.03138 19.94824]
Skew: alpha_c = [0.00000] +/- [0.00000] => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion: kc = [0.13995 -0.49685 -0.00421 0.00270 0.00000] +/- [0.00721 0.02887 0.00074 0.00086 0.00000]
Pixel error: err = [1.09649 1.41861]

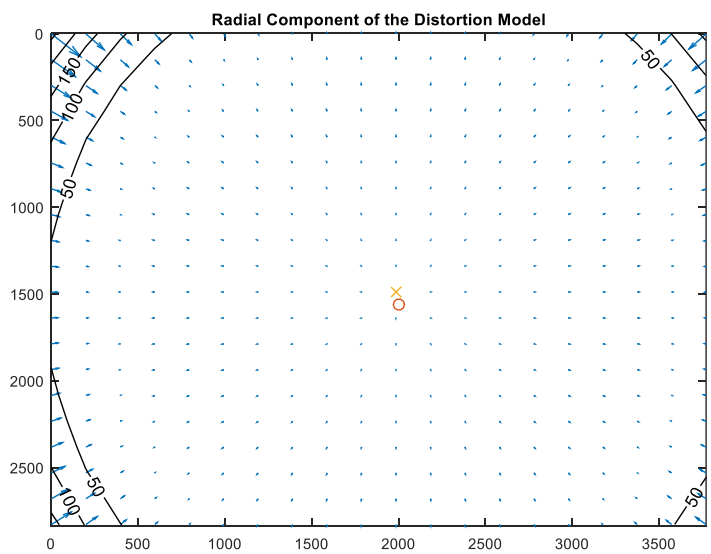
As it may be observed, as a consequence from using all those tools, the pixel error has decreased from the initial [1.41622 1.74597] to [1.09649 1.41861]

8. Visualizing the distortions creates the following plots. The guess is that indeed, since the smartphone possesses two cameras, the distortion is "shifted" away from the second camera, so that together they create a full "symmetric" distortion model.





Pixel error	= [1.207, 1.472]	
Focal Length	= (3053.92, 3084.16)	+/- [29.44, 26.62]
Principal Point	= (1998.19, 1560)	+/- [7.87, 25.49]
Skew	= 0	+/- 0
Radial coefficients	= (0.1196, -0.3962, 0)	+/- [0.00752, 0.02929, 0]
Tangential coefficients	= (-0.004411, 0.002749)	+/- [0.0008022, 0.000984]



Pixel error	= [1.207, 1.472]	
Focal Length	= (3053.92, 3084.16)	+/- [29.44, 26.62]
Principal Point	= (1998.19, 1560)	+/- [7.87, 25.49]
Skew	= 0	+/- 0
Radial coefficients	= (0.1196, -0.3962, 0)	+/- [0.00752, 0.02929, 0]
Tangential coefficients	= (-0.004411, 0.002749)	+/- [0.0008022, 0.000984]

9. Using one of the images, the extrinsic parameters were also calculated:

Extrinsic parameters:

Translation vector: Tc_ext = [157.196418 -33.157908 330.557975]
Rotation vector: omc_ext = [0.919513 -2.736548 0.820951]
Rotation matrix: Rc_ext = [-0.803385 -0.594159 0.039334
-0.517696 0.664303 -0.539160
0.294217 -0.453516 -0.841285]
Pixel error: err = [1.70132 1.94694]

Question 2 – (d)

By taking the data provided by the toolbox in the previous section of the question, the calibration matrix K is obtained: (values are shortened up to the closest natural number)

$$K = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3065 & 0 & 1999 \\ 0 & 3093 & 1549 \\ 0 & 0 & 1 \end{bmatrix}$$

If, for example, we use the image that the extrinsic parameters were extracted for (Question 3 (c) – 9), we obtain the following projection matrix:

$$K\,[\,R \mid t] = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} [\,R_{3\times3} \mid t_{3\times1}] =$$

$$= \begin{bmatrix} 3065 & 0 & 1999 \\ 0 & 3093 & 1549 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -0.803385 & -0.594159 & 0.039334 & 157.196418 \\ -0.517696 & 0.664303 & 0.039334 & -33.157908 \\ 0.294217 & -0.453516 & -0.841285 & 330.557975 \end{bmatrix} = M_{3\times4}$$

$$M_{3\times4} = \begin{bmatrix} -1874.23 & -2727.67 & -1561.17 & 1142592.4 \\ -1145.49 & 1352.19 & -1181.49 & 409476.89 \\ 0.294 & -0.4535 & -0.8412 & 330.55 \end{bmatrix}$$

This matrix M is unique for each of the images, since the extrinsic parameters do change for every camera pose.

Question 3

(a) The images were taken

(b) Using the provided vlfeat library features and descriptors have been extracted.

- Each of the feature vectors is a 4-row vector: $\begin{cases} (f(1:2)) - location\ of\ the\ feature \\ f(3) - scale \\ f(4) - orientation \end{cases}$
- Each of the descriptor vectors is a 128 elements vectors, describing orientation histogram (for 8 orientations) for each of the 16 cells in the vicinity of the feature (16 * 8 = 128).

The vlfeat library allows the usage of customized parameters for feature extraction:

- Peak threshold
- Edge threshold

They define the threshold for a potential feature to be declared a feature. In our example, 57 features were extracted for #1 and 64 features were extracted for Image #2:

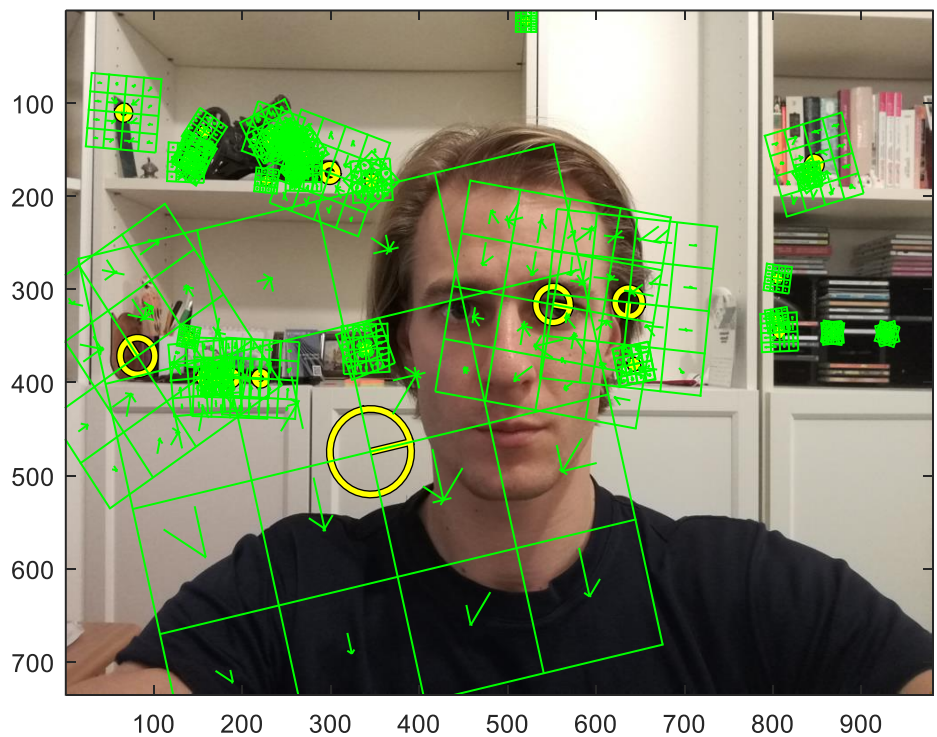


Figure 1 Image 1: 57 features detected

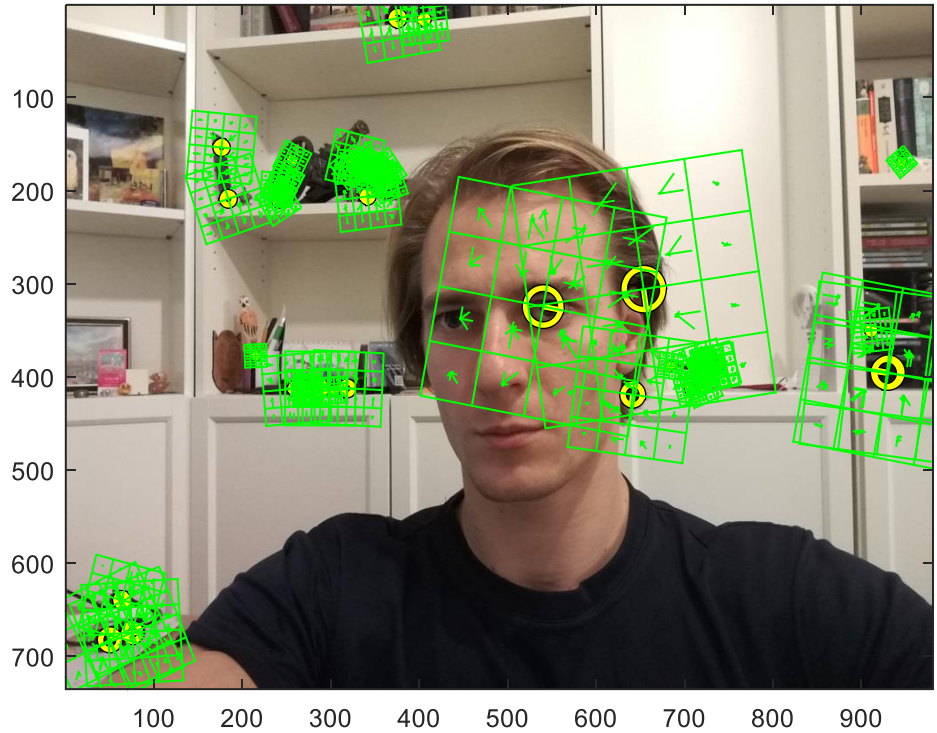
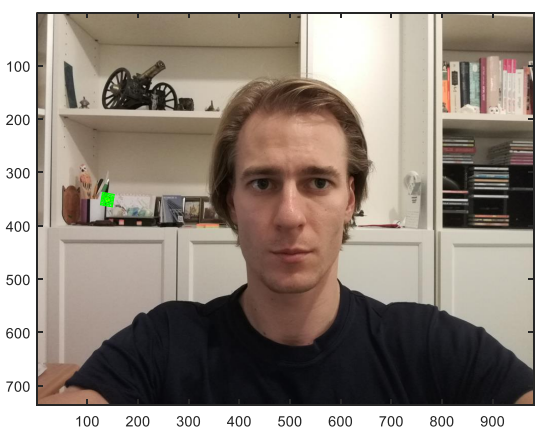
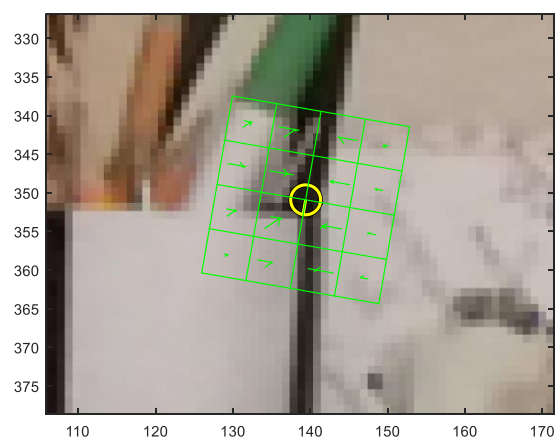


Figure 2 Image2: 64 features detected

Scale and orientation for a representative feature: select arbitrary index 1. The feature 1 on Image1 is the following: (right image is the close-up)



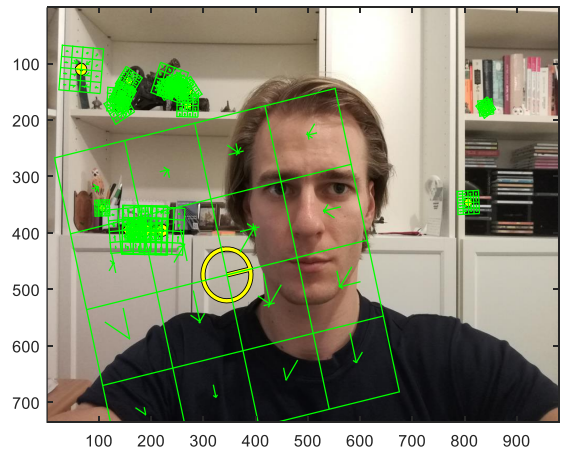


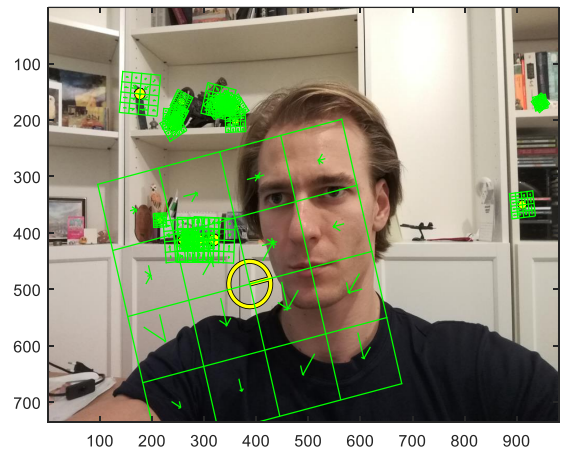
The representative feature is then : $\begin{cases} x \\ y \\ scale \\ orientation \end{cases} = \begin{cases} 139.4 \\ 350.91 \\ 1.937 \\ 0.172 \end{cases}$; with scale =1.937 & orientation = 0.172

(c) The vlfeat library does possess the function for the feature matching, but since the question asks to perform the calculation as explained in class, the algorithm is written manually, while the library function helps us to validate the results. The algorithm for feature matching is the following:

1. Calculate the Euclidean distance from every feature descriptor (128 row vector) in Image 1 to every feature descriptor in Image 2.
2. For every feature in Image 1, find the minimum Euclidean distance out of Euclidean distances calculated to every feature in Image 2 (calculated in step 1)
3. Use the threshold parameter to validate this feature is “unique” enough. Meaning, the minimal Euclidean distance multiplied by threshold is still smaller than any other Euclidean distance to other features. Default value in vlfeat library is 1.5
4. If the condition in Step 3 holds, add this couple of matched features (their indexes) to the matched array.

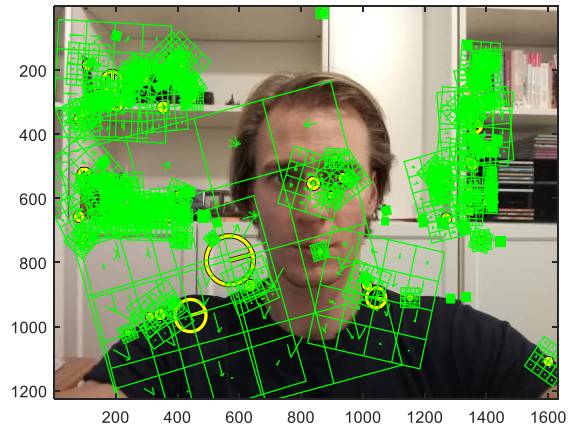
The features matched by this way using a certain threshold (1.5) may be observed in the following image:

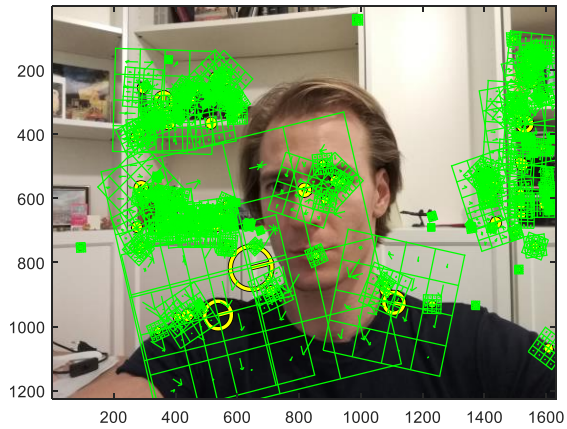




By the first look, it is indeed observable that those features are corresponding to each other. No outliers were found on this image, since the threshold parameters were set quiet to make the features amount small (for easier visual inspection).

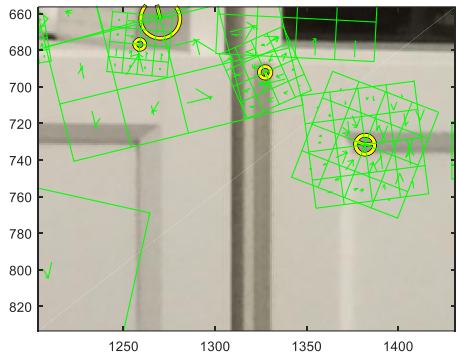
In order to find the outliers, the higher resolution image was used, with smaller threshold parameters for feature extraction. As a result, in the following specific example, for Image 1 and 2 : 1188 and 1210 features were extracted, with 237 matches:

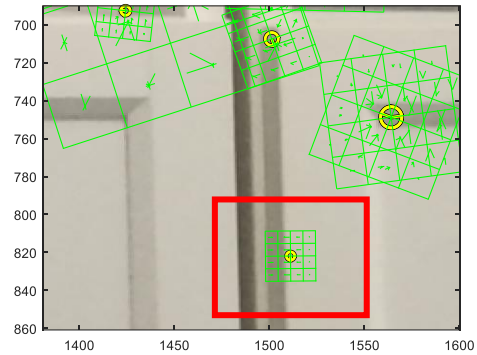




After the fast inspection, the following are the two examples of the outliers: (marked with red boxes)

Outlier 1 :





Outlier 2 :