EE417 - Homework Assignment 2

Baris Sevilmis
Due Date: 09/12/2018

Overview

Homework Assignment 2 focuses on two different camera calibration techniques. Method 1 provides the same camera calibration technique as in Lab 5, on the other hand Method 2 ensures an advanced camera calibration technique, in which automatization of camera calibration is achieved. In addition, Method 2 is based on decomposition of the problem into sub problems, where approaching sub problems gives much more accurate results and possible error analysis chance as the algorithm proceeds. Both of the Camera Calibration Techniques will be provided with their results and lastly, comparison of these techniques will be provided as well in terms of efficiency, results, accuracy and time and effort management. For this reason, both algorithms will proceed on images taken from same camera, namely Macbook Pro camera. Lastly, all the millimeter/centimeter based measurements are done by ruler in real world(Method 1).

Method 1: Orthogonal Planes for Camera Calibration

As already known, earlier camera calibration techniques required orthogonal planes, in which minimum of 2 planes were crucial. Understanding calibration needs understanding of perspective projection that can be explained from the following formula:

$$[u_i v_i 1]^T = M[X_i Y_i Z_i 1]^T$$

$$[u_i v_i 1]^T = K[RT][X_i Y_i Z_i 1]^T$$

X_i, Y_i, Z_i stands for the world frame $point_i$. As it is clear, world frame matrix is $4X1$ instead of $3X1$. Homogenization of the world point is mandatory for calculating pixel coordinates through extrinsic and intrinsic parameters. For this reason, pixel coordinates are also in homogeneous format, to be more specific $3X1$ matrix. Extrinsic and intrinsic parameters ensure transformation of the world coordinates into the corresponding pixel coordinates given that they are calibrated. Intrinsic parameter matrix consists of the following elements:

$$K = \begin{bmatrix} \alpha & -\alpha \cot\theta & u_0 \\ 0 & \frac{\beta}{\sin\theta} & v_o \\ 0 & 0 & 1 \end{bmatrix}$$

α & β stands out as focal lengths of image plane. u_0 & and v_0 are origin points of the pixel coordinate system and θ stands for the skew parameter. On the other hand, extrinsic parameters consist of following two matrices:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

R is the rotation matrix and T is the translation matrix. When they are combined, $3X4$ matrix is produced that contains all the necessary extrinsic parameters. Intrinsic

and extrinsic parameters produce M matrix when combined, in which all the calibration parameters are maintained. Determination of the parameters is the real issue in terms of calibrating camera. To specify M matrix, following formula is used:

$$Pm = 0$$

$$P = \begin{bmatrix} X_0 & Y_0 & Z_0 & 1 & 0 & 0 & 0 & 0 & -u_0X_0 & -u_0Y_0 & -u_0Z_0 & -u_0 \\ 0 & 0 & 0 & 0 & X_0 & Y_0 & Z_0 & 1 & -v_0X_0 & -v_0Y_0 & -v_0Z_0 & -v_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad m = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix}$$

Since there are 11 unknown parameters combined, minimum of 6 points(12 equations) are required for estimating these parameters, however 6 points would not provide reliable parameter values. Therefore, accuracy of parameters increase with the increase of points ensured from different planes of the calibration object. For this reason, P matrix has size of $2nX12$. n stands for the point amount and 2n for the equation amount. As P matrix is formed by gathering all corresponding world points and pixel coordinates(detected corners), SVD is used to decompose the matrix. UDV^T is obtained from the SVD method, as last column of V^T contains our parameter matrix of M.

In Homework 2 Assignment, Figure 1 was used as the calibration object for method 1 that consists of 2 orthogonal planes. Coordinate system can be seen in Figure 1.

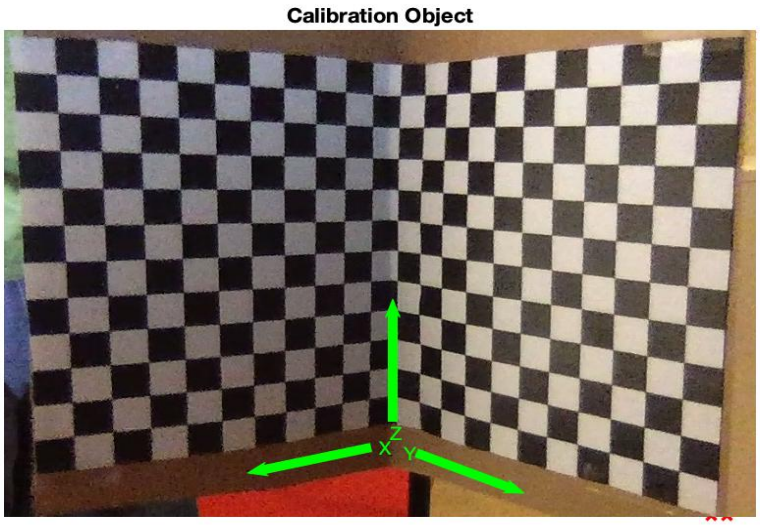


Figure 1: Original Calibration Object

For accuracy purpose, sub accurate corner detection is used, and therefore hough lines and their intersections are detected.

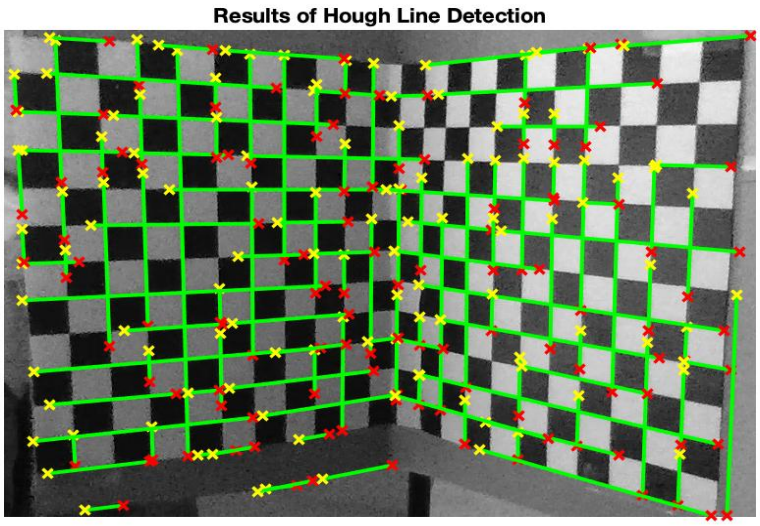


Figure 2: Hough Lines Detected Calibration Object

Detecting sub accuracte corners manually requires too much effort, for this reason a basic mathematical formula is used to detect required amount of line intersections. This formula can be seen in Listing 1. Results of the sub accurate corner detection are depicted in Figure 3, as total of 80 points are detected. Both planes contain more than 30 detected corner points. To be more specific, left plane has 37 detected and right plane has 33 corners detected.

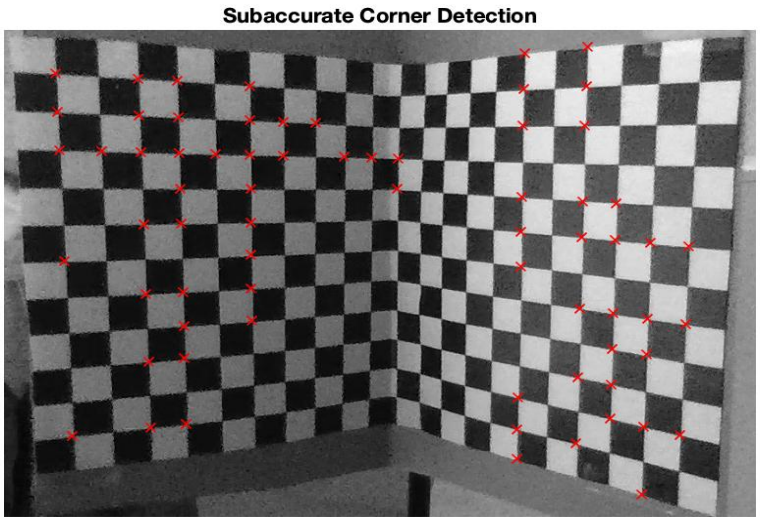


Figure 3: Sub accurate Corner Detected Calibration Object

As for the world coordinates to be entered into the MatLab, we store world coordinates in P_{world} and pixel coordinates in P_{prev} such that they will be multiplied later on to produce P matrix. Notice that, origin point in Figure 1 is bottom middle and length

of one side of squares is equal to 14 millimeters. Therefore, world coordinates will be assigned in mm format. For convenience, since P_{prev} & P_{world} are $160X12$ matrices, following matrices will ensure the assumption of this matrix element-wise multiplication operation and world coordinates:

$$P_{world} = \begin{bmatrix} 140 & 0 & 126 & 1 & 0 & 0 & 0 & 0 & 140 & 0 & 126 & 1 \\ 0 & 0 & 0 & 0 & 140 & 0 & 126 & 1 & 140 & 0 & 126 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 112 & 98 & 1 & 0 & 0 & 0 & 0 & 0 & 112 & 98 & 1 \\ 0 & 0 & 0 & 0 & 0 & 112 & 98 & 1 & 0 & -112 & 98 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$P_{prev} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & -u_0 & -u_0 & -u_0 & -u_0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -v_0 & -v_0 & -v_0 & -v_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & -u_i & -u_i & -u_i & -u_i \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -v_i & -v_i & -v_i & -v_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

P_{world} is given as a parameter to the function, however P_{prev} is established during sub-accurate corner detection. As P matrix is created after $P_{world} * P_{prev}$, SVD is used to decompose P. As mentioned previously, last column of V^T contains parameters of M matrix. To find intrinsic and extrinsic parameters, we simply use the formulas provided in lecture slides. To conclude, Listing 1 & 2 provides necessary MatLab codes to achieve above operations. For the results, Figure 4 depicts the Rotation Matrix in $9X1$ format and Figure 5 depicts the Translation Matrix.

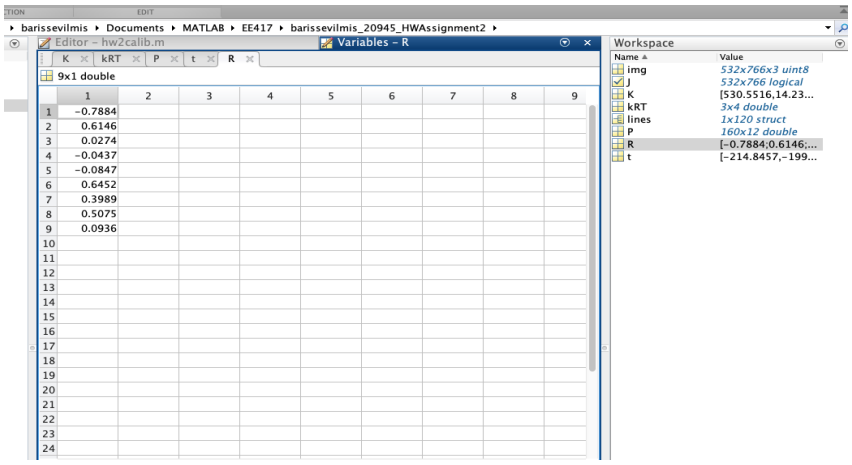


Figure 4: Rotation Matrix in 9X1 format

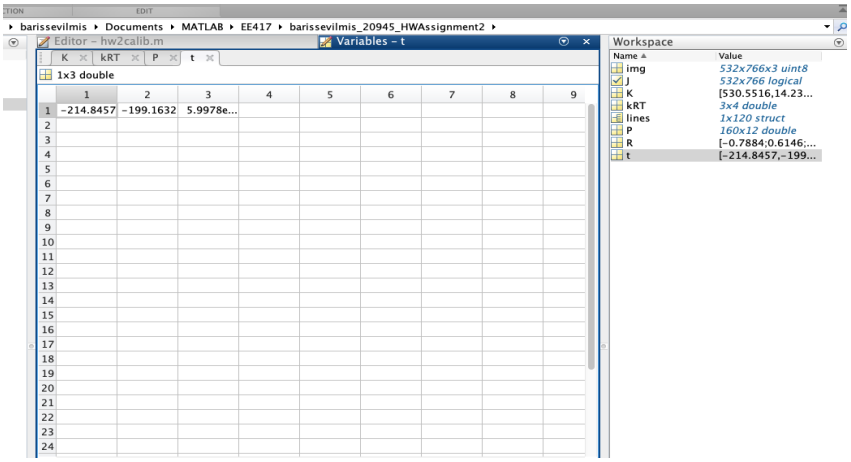


Figure 5: Translation Matrix

Figure 6 demonstrates the K matrix, namely intrinsic parameters and lastly, Figure 7 clearly shows the M matrix where both intrinsic and extrinsic parameters are combined.

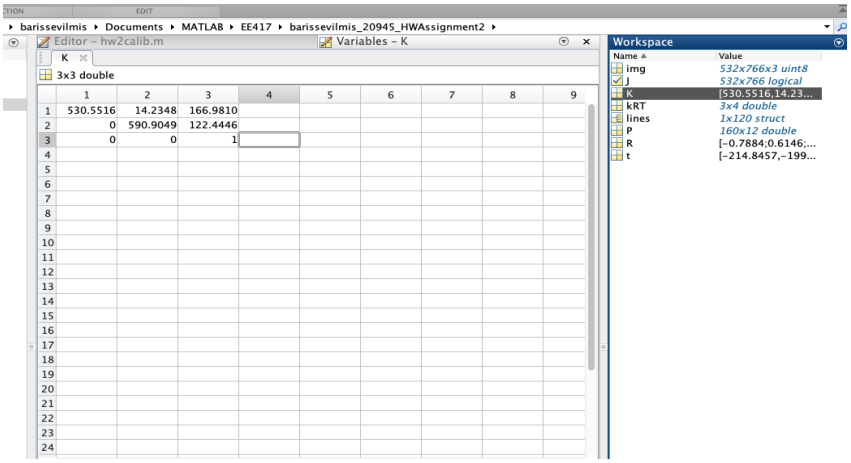


Figure 6: Intrinsic Parameter Matrix

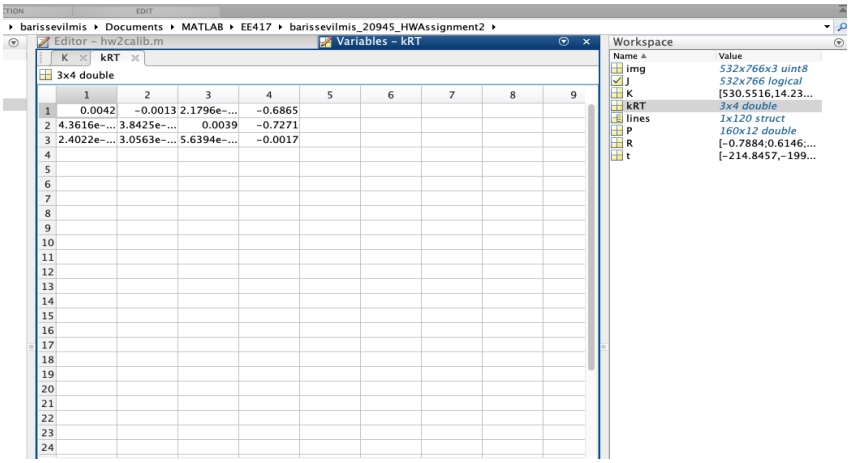


Figure 7: Intrinsic and Extrinsic Parameter Combined Matrix

Listing 1 implies the Hough Line Detection part of the Matlab code. Listing 2 & Listing 3 provide next part of the code that is sub-accurate pixel detection. Method to detect sub accurate corners is actually very basic, as only requirement is to detect whether two lines are intersecting. Since we know both the start and end points of the lines, we can easily detect sub accurate corners. Finally, Listing 4 provides the establishment of matrix P, single value decomposition of P and calculation of intrinsic and extrinsic parameters.

Listing 1: MatLab Code for Camera Calibration Method 1- Hough Lines Detection

```
function [J, lines, kRT,K,t,R, P, P_world] = hw2calib(img,
    P_world)
    [row, col, ch] = size(img);
    if(ch == 3)
        rgbimg = rgb2gray(img);
    end
    %Canny edge detection
    J = edge(rgbimg,'Canny');
    %Hough Transform
    [H,theta,rho] = hough(J,'RhoResolution',1,'Theta'
        ,-90:0.5:89);
    %Hough Peaks
    threshold = 0.5*max(H(:));
    J_peaks = houghpeaks(H,60, 'Threshold', 100);
    figure(1);
    imshow(img);title('Calibration Object');
    figure(2);
    imshow(rgbimg);title('Results of Hough Line Detection');
    hold on;
    %Hough Lines
    lines = houghlines(J,theta,rho,J_peaks,'FillGap',10, '
        MinLength',30);
    %Hough Lines Plot
    max_len = 0;
    min_len = 200;
    for k = 1:length(lines)
        xy = [lines(k).point1; lines(k).point2];
        plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
        % Plot beginnings and ends of lines
        plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow')
            ;
        plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
        % Determine the endpoints of the longest line segment
        len = norm(lines(k).point1 - lines(k).point2);
    end
    figure(3);
    imshow(rgbimg);title('Subaccurate Corner Detection');
    hold on;
```

Listing 2: MatLab Code for Camera Calibration Method 1- Sub accurate Corner Detection - Part 1

```
P_prev = [];
counter = 0;
%14 mm box width-height
%Detect all the intersections
for ii = 1:length(lines)
    for jj = 1: length(lines)
        if ii ~= jj

            if ((lines(ii).point1(1) > lines(jj).point1(1)) && (lines(ii).point1(1) < lines(jj).point2(1)))...
                && ((lines(jj).point1(2) < lines(ii).point2(2)) && (lines(jj).point1(2) > lines(ii).point1(2)))
                rho1 = lines(ii).rho;
                theta1 = lines(ii).theta;
                rho2 = lines(jj).rho;
                theta2 = lines(jj).theta;
                X1 = 0:0.1:350;
                Y1 = (rho1 - X1*cosd(theta1))/sind(theta1);
                X2 = 0:0.1:350;
                Y2 = (rho2 - X2*cosd(theta2))/sind(theta2);
                A = [cosd(theta1) sind(theta1); cosd(theta2) sind(theta2)];
                rhoMat = [rho1; rho2];
                invA = pinv(A);
                point = invA * rhoMat;
                    plot(point(1), point(2),'rx');
                    %Fill P matrice
                    P_prev = [P_prev; 1 1 1 1 0 0 0 0 -1*point(1) -1*point(1) -1*point(1) -1* point(1);...
                        0 0 0 0 1 1 1 1 -1*point(2) -1*point(2) -1*point(2) -1* point(2)];
                    counter = counter + 1;
```

Listing 3: MatLab Code for Camera Calibration Method 1- Sub accurate Corner Detection - Part 2

```
1
2               elseif ((lines(jj).point1(1) > lines(ii).point1(1))
                  && (lines(jj).point1(1) < lines(ii).point2(1)))...
3                     && ((lines(ii).point1(2) < lines(jj).point2
                          (2)) && (lines(ii).point1(2) > lines(jj).
                          point1(2)))
4                   rho1 = lines(ii).rho;
5                   theta1 = lines(ii).theta;
6                   rho2 = lines(jj).rho;
7                   theta2 = lines(jj).theta;
8                   X1 = 0:0.1:350;
9                   Y1 = (rho1 - X1*cosd(theta1))/sind(theta1);
10                  X2 = 0:0.1:350;
11                  Y2 = (rho2 - X2*cosd(theta2))/sind(theta2);
12                  A = [cosd(theta1) sind(theta1); cosd(theta2)
                        sind(theta2)];
13                  rhoMat = [rho1; rho2];
14                  invA = pinv(A);
15                  point = invA * rhoMat;
16                  plot(point(1), point(2),'rx');
17                  P_prev = [P_prev; 1 1 1 1 0 0 0 0 -1*point(1)
                        -1*point(1) -1*point(1) -1* point(1);...
18                  0 0 0 0 1 1 1 1 -1*point(2) -1*point(2) -1*point
                        (2) -1* point(2)];
19                  counter = counter + 1;
20              end
21              if counter == 80
22                  break
23              end
24          end
25      end
26      if counter == 80
27          break
28      end
29  end
```

Listing 4: MatLab Code for Camera Calibration Method 1- SVD, Intrinsic and Extrinsic Parameters

```
1
2       P = P_prev .* P_world;
3
4       [U,D,V] = svd(P);
5       M = V(:, end);
6       kRT = [M(1:4)'; M(5:8)'; M(9:12)'];
7
8       b = kRT(:,4);
9       a1 = kRT(1, 1:3);
10      a2 = kRT(2, 1:3);
11      a3 = kRT(3, 1:3);
12
13      l = 1 / norm(a3, 1);
14      r3 = l*a3;
15      u0 = l*l*dot(a1, a3);
16      v0 = l*l*dot(a2, a3);
17      alpha = l*l*norm(cross(a1, a3),1);
18      beta = l*l*norm(cross(a2, a3), 1);
19
20      cosTheta = -dot(cross(a1,a3),cross(a2,a3))/(norm(cross(a1,a3
            ),2)*norm(cross(a2,a3),2));
21      sinTheta = sqrt(1-cosTheta^2);
22      r1 = cross(a2,a3)/norm(cross(a2,a3),2);
23      r2 = cross(r3,r1);
24      % Intrinsic Matrix
25      K = [alpha -1*alpha*(cosTheta/sinTheta) u0;...
26           0 beta/sinTheta v0;...
27           0 0 1];
28      % Translation Matrix
29      t = l * (b\inv(K));
30      % Rotation Matrix
31      R = [r1' ;r2' ;r3'];
32  end
```

All together, these Listings(1-4) ensure implementation of first technique of camera calibration by detecting sub accurate corners as features. As mentioned before, results of these algorithms can be seen in Figures above.

Method 2: Zhengyou Zhang's Camera Calibration Technique

Zhengyou Zhang offers a new approach for camera calibration, in which one takes image of a single plane from different rotations and translations instead of using orthogonal multiple planes at the same time. This technique uses the same set parameters as previous technique since both of the methods deal with camera calibration and set of concepts are same. In other words, estimated parameters & formulas are valid for this method.

Therefore, concepts will not be explained here again. For convenience, formulation in Method 1 can be followed.

As for this new method, there are certain advantages in comparison to the first method. Camera calibration toolbox provides all the necessary camera calibration steps and for this reason, camera calibration is automatized. For instance, there is no need to pick world frame coordinates manually with a ruler as before, instead algorithm detects all the corresponding world coordinates and pixel coordinates in terms of detected corners. Secondly, main camera calibration problem is decomposed into subproblems, in which following algorithm is much easier and algorithm itself becomes much more reliable, efficient and maintainable. For the sake of clarity, algorithm first focuses on corner detection of images one by one, then focuses on calibrating intrinsic and extrinsic camera parameters and finally, provides visualization, error analysis and recalibration options. Nevertheless, as already said toolbox ensures visualization and error analysis for the calibrated camera. As a consequence, one can understand the extrinsic parameters much better from the visualization and related error measurement. With help of error analysis, camera can be calibrated again with toolboxes automatic corner detection. If done carefully, calibrated camera proves much better results as these steps will be provided in advance. Lastly, in addition to the mentioned advantages, taking different images from different rotations and translations and therefore, providing data for camera calibration is definitely easier than Method 1. Using multiple planes is limited at a level such that taking an image with more than 3 orthogonal planes is not possible. Therefore, Method 1 is restricted to a level, in which maximum amount of planes that can be used for calibration gets fixed. Controversy, Method 2 can be supplied with as much as image needed and therefore, can give much more accurate parameter estimation as there is no upper limit of incoming data.

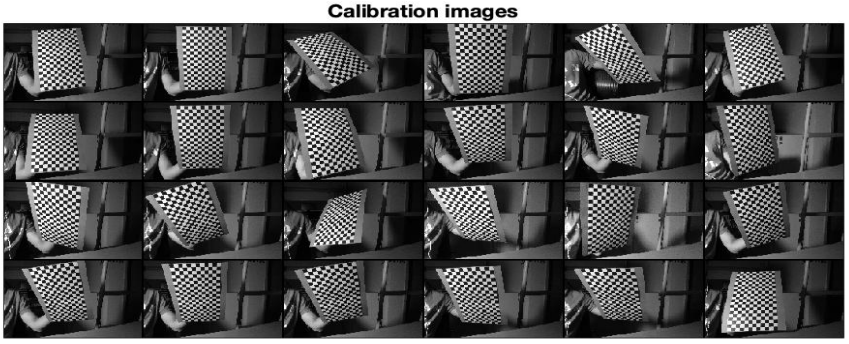


Figure 8: Calibration Images

As for the results, Figure 8 depicts all of the 24 images supplied to camera calibration toolbox. For the sake of comparison, all of these images were captured from the same camera with different rotations and translations, as image in Method 1 was also captured from the same camera. Since 24 images were taken, demonstrating all of them would be waste of space. Therefore, first 4 images will be shown for every step. Figure 9, 10, 11 & 12 are depicting calibration object from different rotations and translations, as their grids are determined manually. As the grids are determined, red points on the images are demonstrating the detected corners within the grid.

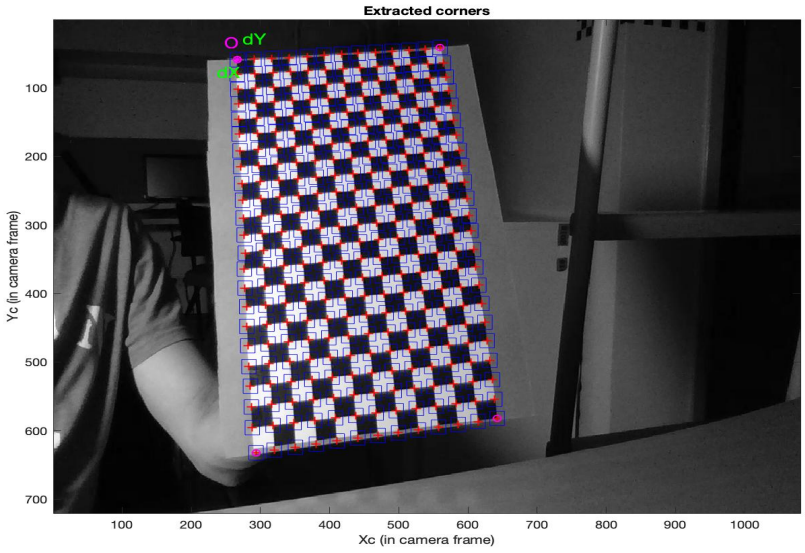


Figure 9: Calibration Image-I

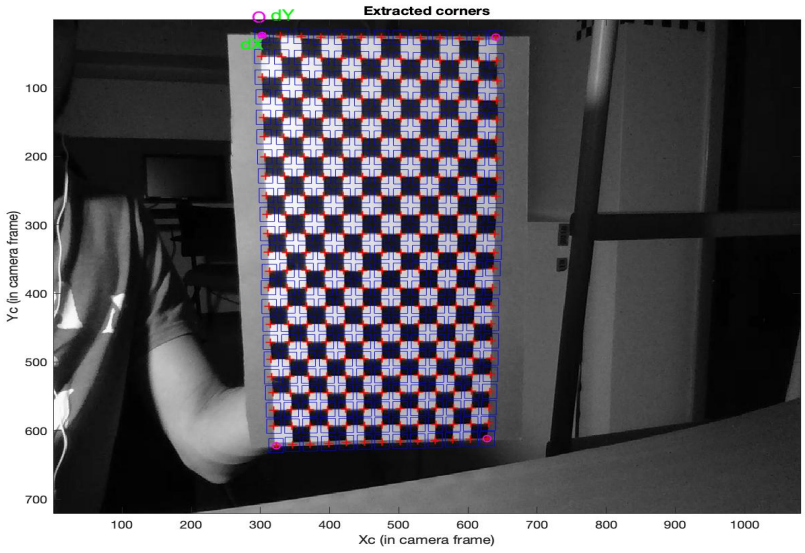


Figure 10: Calibration Image-II

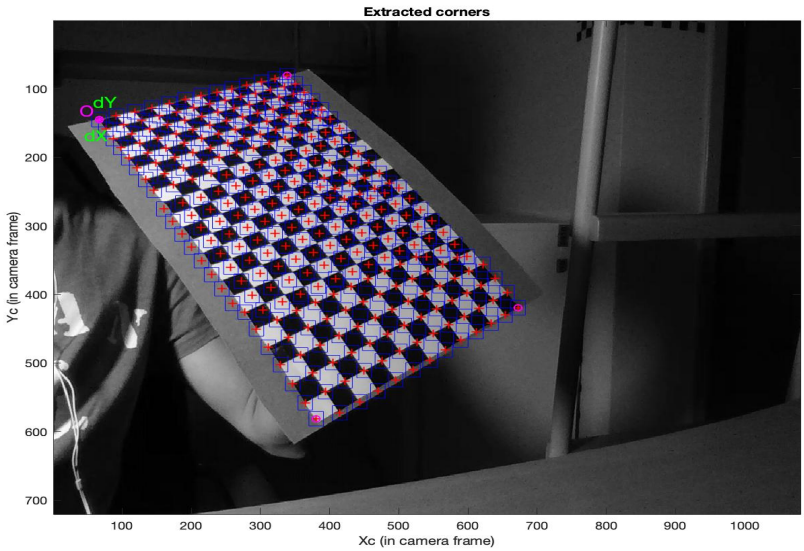


Figure 11: Calibration Image-III

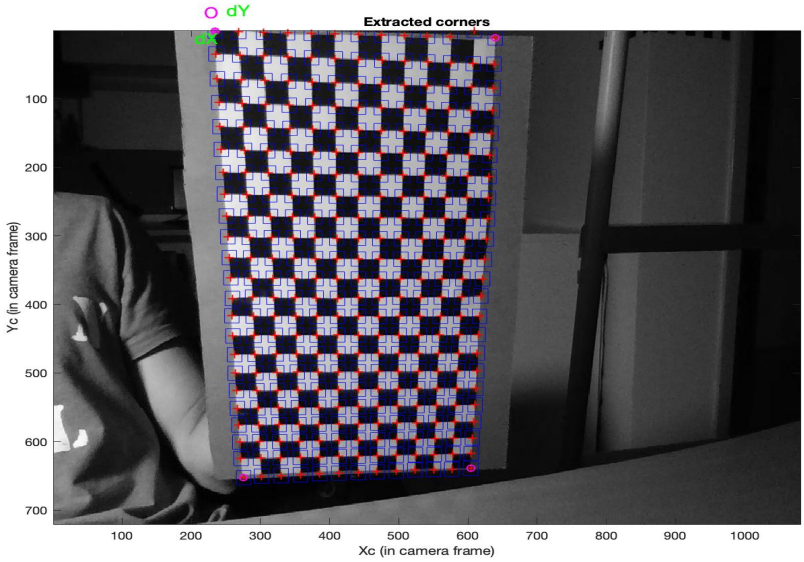


Figure 12: Calibration Image-IV

As the corners are detected, main camera calibration can be started. Figure 13 & 14 demonstrates camera calibration results before and after the optimization. From these results, one can obtain intrinsic parameters of the camera, however as it can be seen standard deviation and err levels are high. Decreasing these would be crucial to make a better camera calibration.

```
Calibration parameters after initialization:

Focal Length:          fc = [ 957.80049   957.80049 ]
Principal point:       cc = [ 539.50000   359.50000 ]
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: 24
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15..
Estimation of uncertainties...done
```

Figure 13: Calibration parameters after Initialization

```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 954.49177   958.70136 ] +/- [ 6.08426   7.19255 ]
Principal point:       cc = [ 532.94488   351.91275 ] +/- [ 7.65604   10.65883 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +
Distortion:            kc = [ -0.00545   0.08401   0.00211   -0.00214  0.00000 ] +/- [ 0.02601
Pixel error:          err = [ 1.57532   2.63866 ]

Note: The numerical errors are approximately three times the standard deviations (for referenc


Recommendation: Some distortion coefficients are found equal to zero (within their uncertainti
                To reject them from the optimization set est_dist=[0;1;1;1;0] and run Calibrat
```

Figure 14: Calibration parameters after optimization

Following Figure 15, 16, 17 & 18 demonstrate the reprojection results of camera calibration after optimization. Gradient descent is used for optimization that takes around 23-24 steps to optimize. As it can be clearly seen from Figure 15 to Figure 18 there are some arrows on some corners indicating that corners were not detected accurately before optimizing calibration.

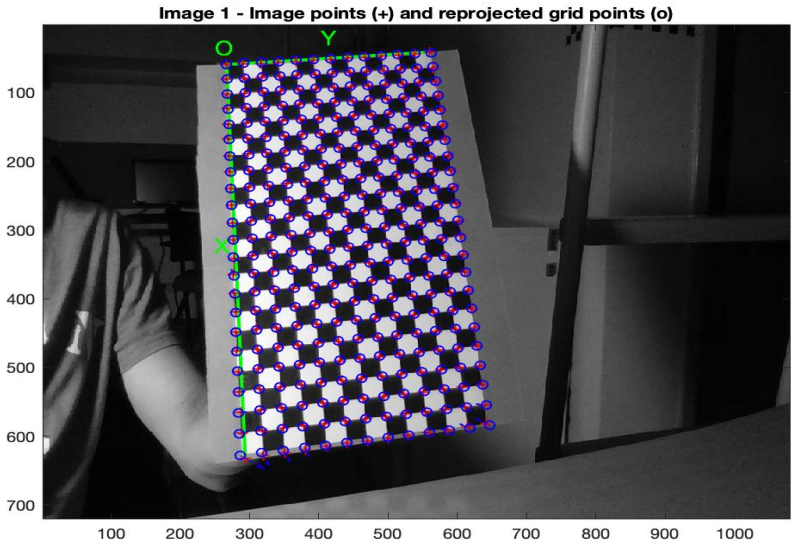


Figure 15: Reprojected Calibration Image-I

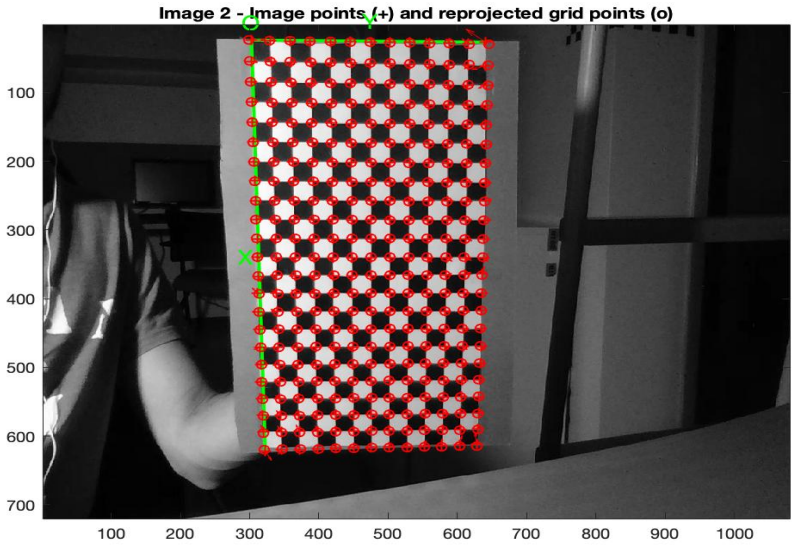


Figure 16: Reprojected Calibration Image-II

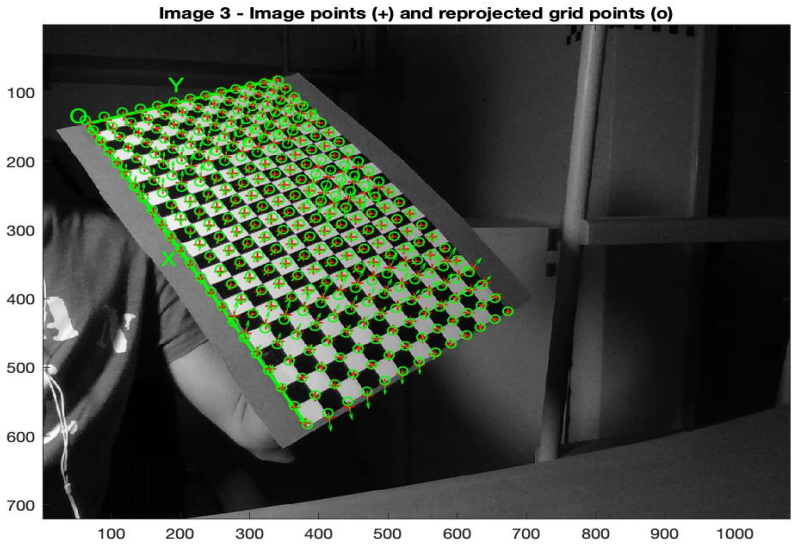


Figure 17: Reprojected Calibration Image-III

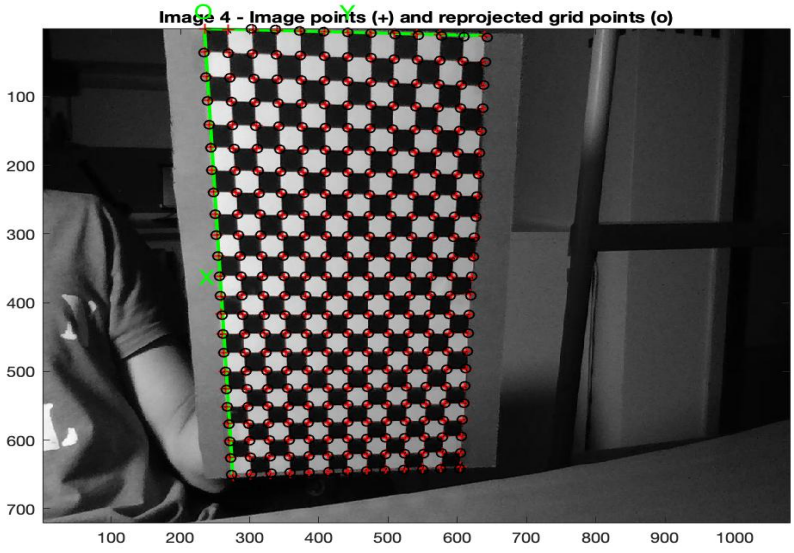


Figure 18: Reprojected Calibration Image-IV

Figure 19 depicts the reprojection error in terms of error analysis, in which error rate is high. Figure 19 implies that as closer these colored crosses, greater is the error rate. In other words, some grid corners were not precisely extracted and therefore, error rate is high. On the other hand, Figure 20 and Figure 21 provides visualization of the extrinsic parameters of the camera calibration. These plots help the user to improve understanding of 3D calibration, and therefore very useful to connect beginning of the algorithm up to the after optimization process.

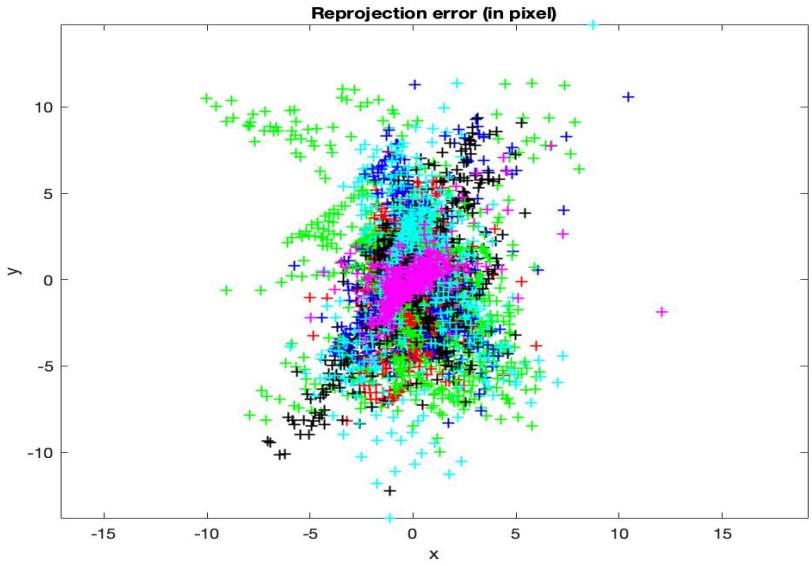


Figure 19: Reprojection Error

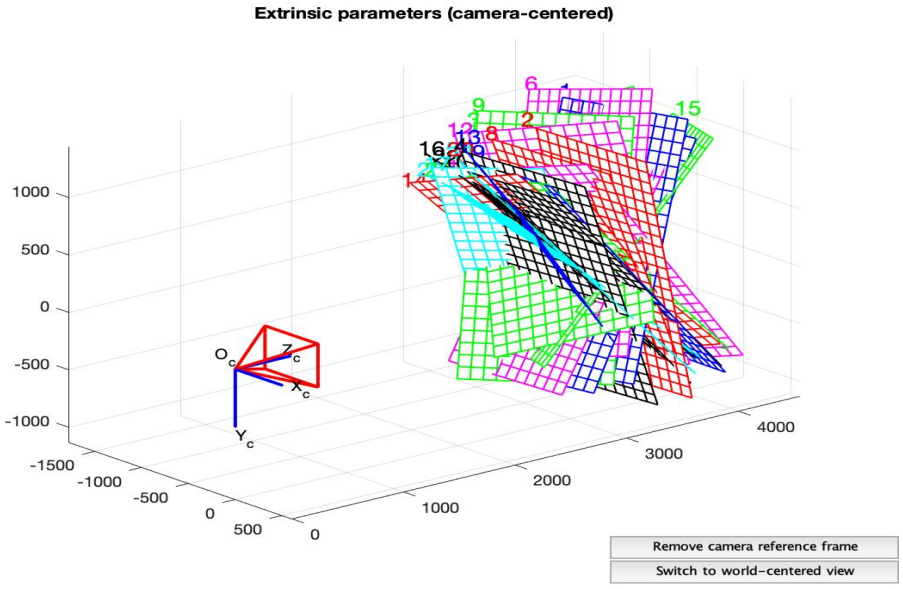


Figure 20: Extrinsic Parameters-Image Plane View

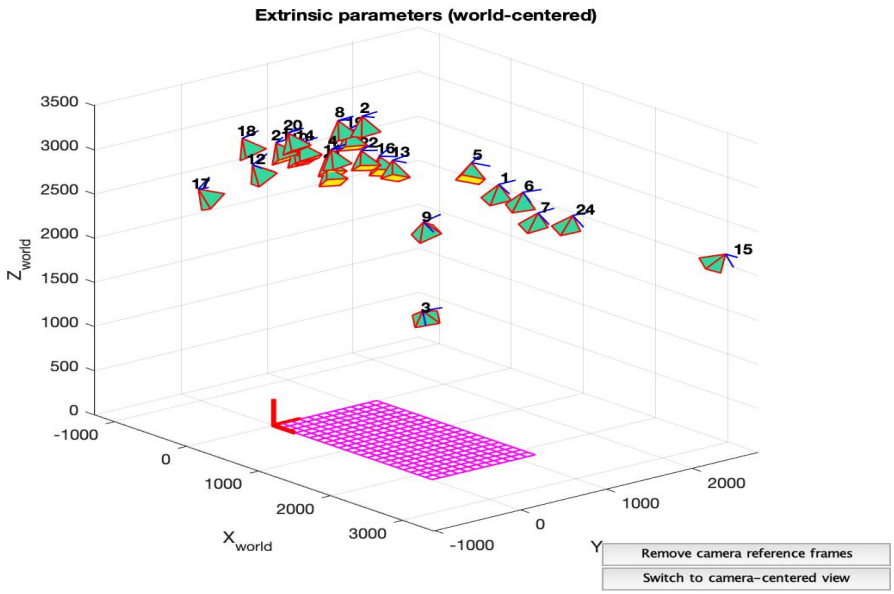


Figure 21: Extrinsic Parameters-World Centered View

To improve accuracy of the calibration parameters even more, toolbox ensures Recomputation of the Corners automatically. By using Recomp. Corners, we have much more reliable results in terms of decreased error and standard deviation. Figure 22 indicates the reduced error and standard deviation.

```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 947.41390   951.30927 ] +/- [ 3.66132   4.33939 ]
Principal point:       cc = [ 528.69451   354.17600 ] +/- [ 4.60641   6.45127 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +
Distortion:            kc = [ -0.00516   0.09201   0.00296   -0.00434  0.00000 ] +/- [ 0.01578
Pixel error:          err = [ 1.14513   1.48845 ]

Note: The numerical errors are approximately three times the standard deviations (for referenc
```

Figure 22: Calibration parameters after optimization

Figure 23, 24, 25 & 26 depicts the reprojected grid points after recomputing corners. Gradient descent continued again for 23-24 iterations, but it can be seen that arrows on the same corners as previous reprojection are larger. Therefore, it can be said that optimization resulted in a more reliable and accurate calibration.

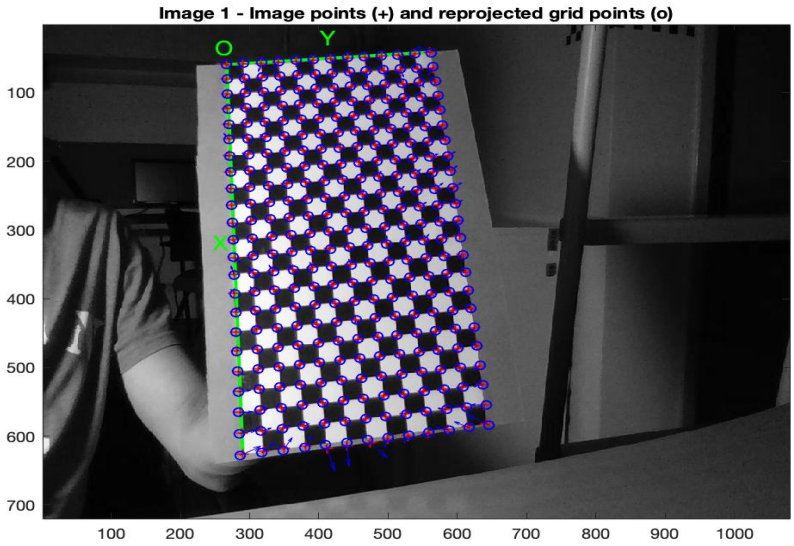


Figure 23: Reprojected Calibration Image-I (Recomp. Corners)

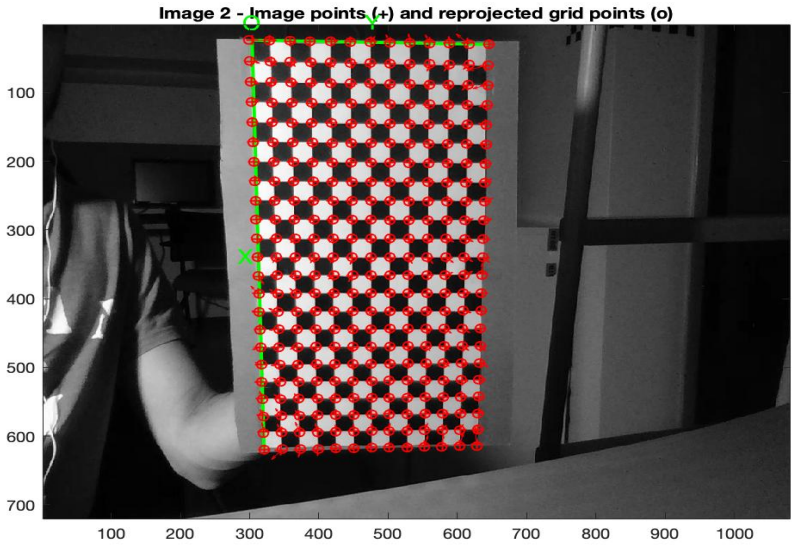


Figure 24: Reprojected Calibration Image-II (Recomp. Corners)

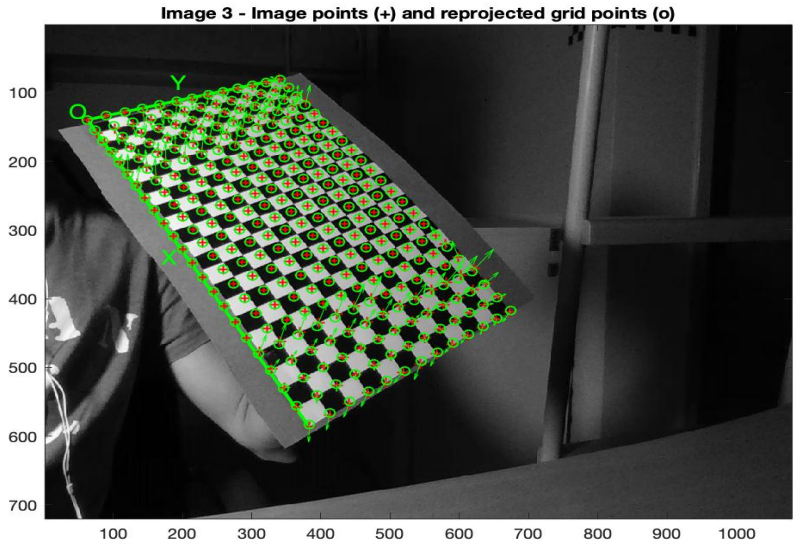


Figure 25: Reprojected Calibration Image-III (Recomp. Corners)

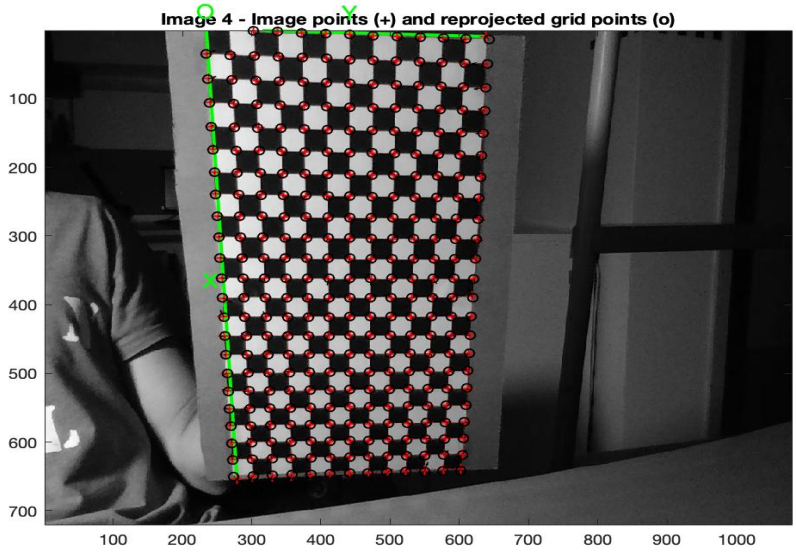


Figure 26: Reprojected Calibration Image-IV (Recomp. Corners)

To improve our understanding, one can check again the error function and the extrinsic parameters. Figure 27 depicts new reprojection error, as the overlap ratio is reduced. In other words, grid corners are detected with a higher accuracy resulting in a lower error rate. In addition, Figure 28 and Figure 29 provide visualization of the recomputed extrinsic parameters in order to enhance the understanding of 3D calibration and extrinsic parameters themselves. From Figure 28 & 29 it can be also seen that error rate is reduced given the understanding of 3D plots.

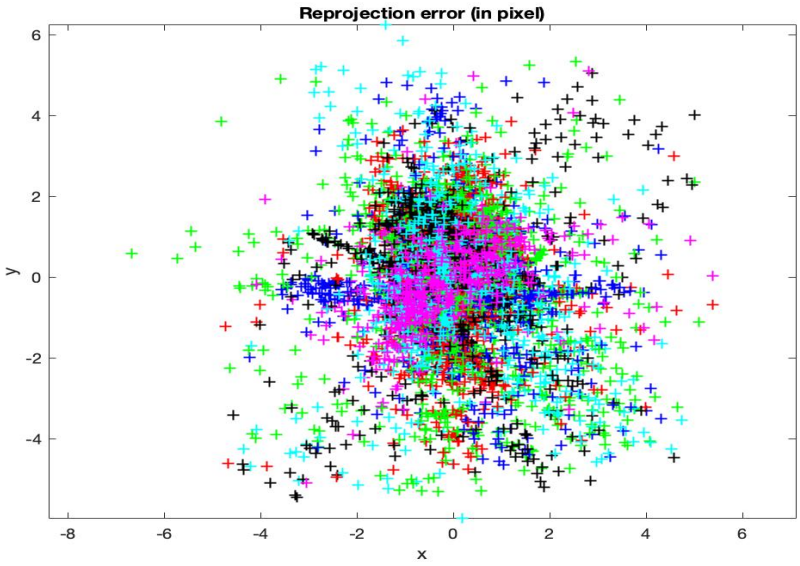


Figure 27: Reprojection Error (Recomp. Corners)

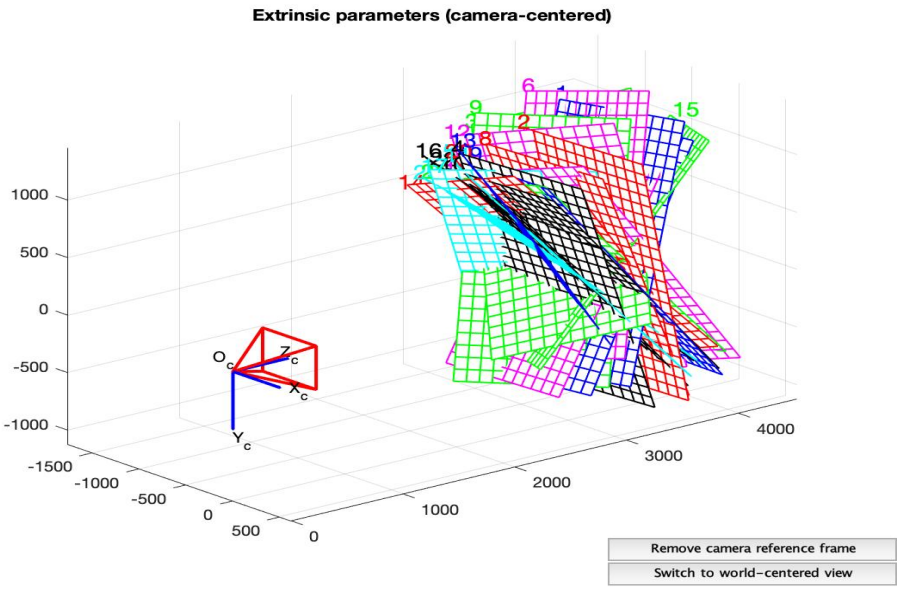


Figure 28: Extrinsic Parameters-Image Plane View (Recomp. Corners)

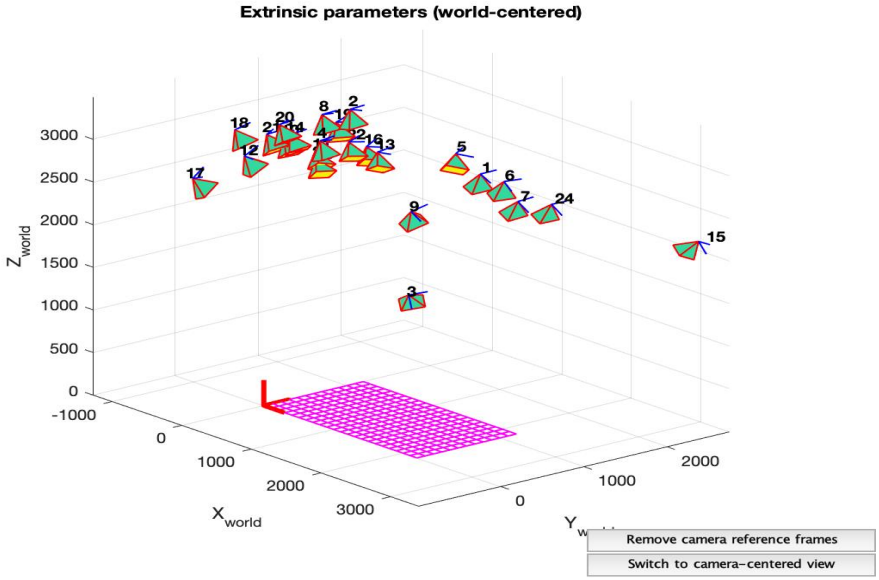


Figure 29: Extrinsic Parameters-World Centered View (Recomp. Corners)

Lastly, distortion takes a big place in this calibration as well such that some images required distortion parameters before being calibrated. Therefore, visualizing distortion models become also essential. Figure 30 demonstrates complete distortion model, as Figure 31 depicts the tangential and Figure 32 depicts the radial distortion model. As it can be seen also on Figures, distortion has a huge role in grid corner detection and if not careful may increase the reprojection error on high rates.

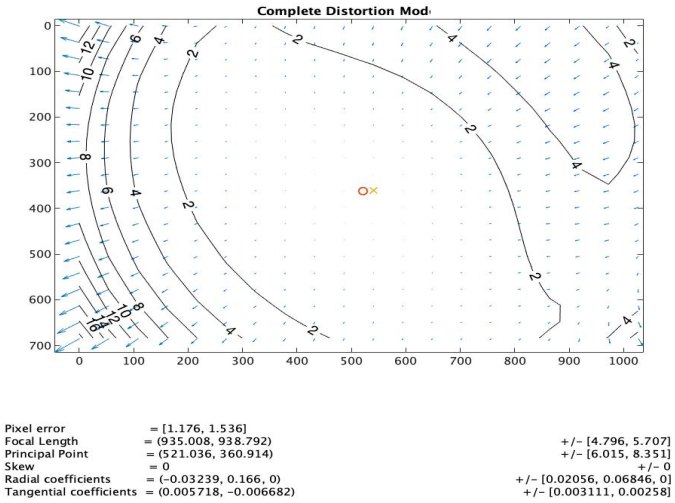
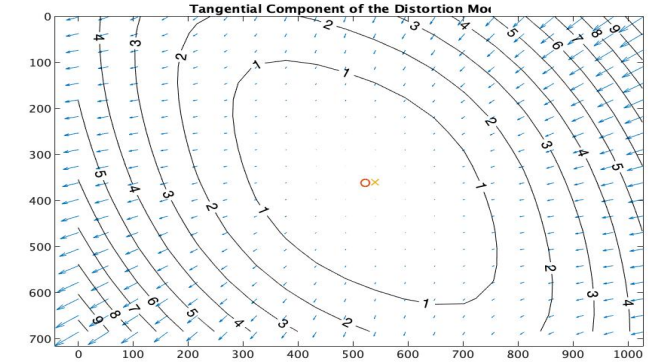
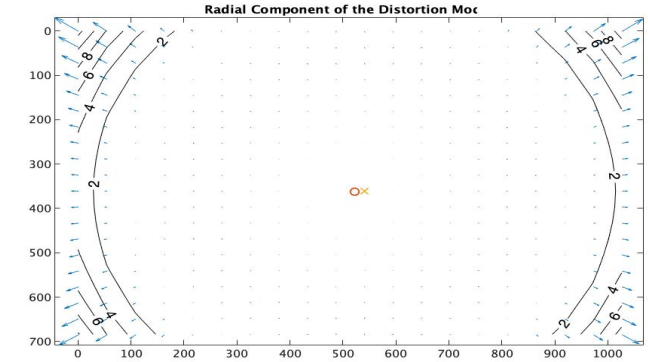


Figure 30: Complete Distortion Model



Pixel error = [1.176, 1.536]
Focal Length = (935.008, 938.792) +/- [4.796, 5.707]
Principal Point = (521.036, 360.914) +/- [6.015, 8.351]
Skew = 0 +/- 0
Radial coefficients = (-0.03239, 0.166, 0) +/- [0.02056, 0.06846, 0]
Tangential coefficients = (0.005718, -0.006682) +/- [0.003111, 0.00258]

Figure 31: Tangential Distortion Model



Pixel error = [1.176, 1.536]
Focal Length = (935.008, 938.792) +/- [4.796, 5.707]
Principal Point = (521.036, 360.914) +/- [6.015, 8.351]
Skew = 0 +/- 0
Radial coefficients = (-0.03239, 0.166, 0) +/- [0.02056, 0.06846, 0]
Tangential coefficients = (0.005718, -0.006682) +/- [0.003111, 0.00258]

Figure 32: Radial Distortion Model

Conclusion

To conclude, by using different sized windows on different subset of images reprojection error can be further reduced. However, this process would need attention, otherwise reprojection error may also increase. We already discussed comparison between provided Method 1 and Method 2, and reasons of the superiority of Method 2. However, intrinsic parameters can be also compared for better results. In our case, the focal lengths of the images are different and principal point differs, though all images were captured by the same camera. Therefore, for comparing both Methods algorithms, error analysis, recomputation options, efficiency and reliability must be considered. In order to see these comparisons with necesessary algorithm details, it would be best to check again the previous parts of Method 1 and Method 2.