ECE661 Homework6

Lin Yuan

Section 1 Problem Description

This homework is about implementing Zhang's algorithm for camera calibration. Besides the techniques from the paper for calculating K and $[R|t]$, we need to use Hough Transform to find lines from the calibration pattern image and then locate the corners, in order to generate the point correspondences automatically.

Section 2 Solution

(1) Generating Correspondences

For this homework, we take a camera to shoot multiple pictures of a fixed flat calibration pattern (Fig. 2) from multiple positions. Thus the world coordinates of the corners on the calibration pattern is fixed as the same. In order to get the pixel positions of each corner, the following steps are taken:

- Use Canny edge detector to generate a binary image of the pattern which highlights the edge of the black blocks. (Fig. 2)
- Hough Transform is performed on the image first to get the lines. These lines are represented as polar coordinate, and because of noise of image, usually multiple lines can be found, which actually correspond to the same line. (Fig. 2)
- These lines are sorted into 2 groups (horizontal and vertical) first depending on their orientations, and then clustered together depending on sampled points' distance on each line (i.e. the parallel gap).
- After exactly 10 horizontal lines and 8 vertical lines are found, the corners are first located as the crossing of the lines.
- Then, Harris corner of the image is found and each crossing is relocated to its nearest Harris corner.

After these steps, exactly 80 pixel positions will be generated for one image's 80 corner points.

(2) Initial estimation of camera parameters

2.2.1 Intrinsic Parameters

The estimation of intrinsic parameters are done in the following steps:

- For each image, estimate H on the correspondence previously generated using the RANSAC algorithm.
- Use Zhang's algorithm, section 3.1 to solve for K.

2.2.2 Extrinsic Parameters

After K is estimated, the extrinsic parameters for each H, R and t is calculated according to $H = K[R|t]$. The initial result of $[R|t]$ need to be rescaled according to the norm of R_1 and then corrected using the Frobenius method as in Appendic C of Zhang's paper.

2.2.3 Radial Distortion

The radial distortion parameters $k1$ and $k2$ are estimated according to Section3.3 of Zhang's paper. We use the estimated intrinsic parameters to generate the equation (13) in Zhang's paper to solve for $k1$ and $k2$.

(3) Refining the estimation using LM

As in last homework, we can refine the parameters $H = K[R_1R_2|t]$ using the LM method based on the objective function which is the distance of projected world points with real corner pixel positions. Using the Rodrigues' representation, each rotation matrix R can be represented with only 3 independent parameters. Thus enabling us to use the LM method. So all of parameters for LM are: $u_0, v_0, \alpha, \beta, \gamma, k_1, k_2$ and $6 \times n$ extrinsic parameters for n images where in each 6 parameters 3 of them are the Rodrigues representation of R, and the other 3 are t. The objective function then becomes $\|\mathbf{x}' - K[R_1R_2t]\mathbf{x}\|^2$ where $\mathbf{x}'$ is the image corner positions and $\mathbf{x}$ are the world coordinates of the corners. Sum all the points from all n images, we get $80 \times n$ distances added together as the total objective function.

(4) Software Structure

I implemented this homework mostly with C++. For the LM optimization, I did it both in C++ and in MATLAB. The MATLAB code was adapted from last homework, but because it is based on the symbolic toolbox, it is slow and can only perform the optimization for at most 5 images as in the total objective function. For the case that 40 images need to be considered, the levmar package is used in

my C++ code. In order to write own LM algorithm in MATLAB that can be applied to 40 images, some tricks like vectorization of the parameters and breaking the calculation of jacobian into parts need to be carried out.
There are several configurations available in my software. All available ways of running the binary hw6 can be found in the file "test.bash". All the results (figures) shown below are from the C++ levmar configuration, per the requirement that experiments need to be carried out on at least 20 images.

Section 3 Experiences and Discussion

Through extensive programming work done these days, I'd like to summarize the lessons learned from this experiment.

- Divide the Hough lines into 2 groups first before merging the lines. Because it is difficult to design a cost function for all orientations of lines. If the lines are initially aligned somehow, the cost function works better. At first I tried to merge the lines first, and then separate them to 2 groups, but the merging didn't work well for all sets of images.

- When using the Frobenius matrix norm to refine R, use only 1 scale factor as λ, in the last part of Section 3.1 of Zhang's paper. I used $1/\|K^{-1}h_1\|$ as λ and then rescale t also using that λ. And then t is left there.

- When using levmar, the distance function provided to levmar must not have memory leak. i.e. All memories allocated inside that function need to be released because that function need to be called many times.

- When calculating Jacobian in MATLAB using symbols, after calculation, one can store the symbol representation of the Jacobian into a file as a function. So that one do not need to use the $subs$ method which is very slow.

- According to the last homework, I used my own LM MATLAB code at first. However, it works for at most 5 images. Let alone that I need to generate the MATLAB code of symbols from C++ iteratively, the size of the Jacobian is just intractable, for more than 5 images. So, levmar is suggested for use when you need to optimize on many images, while extra techniques need to be applied to avoid the symbolic toolbox for MATLAB to work for more than 5 images as in the total objective function.

- My own MATLAB code for this homework is also attached. Some of the corrected images using the my MATLAB LM function is listed on Fig. 1

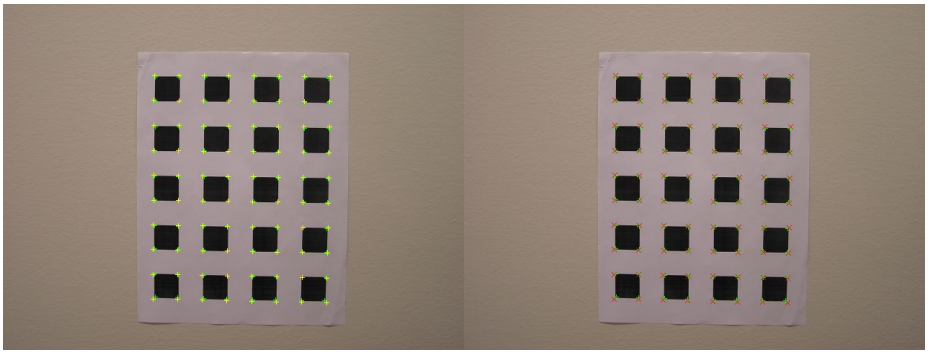
Section 4 Results

	Initial	Refined
K	$\begin{pmatrix} 717.359251 & 1.582683 & 324.590701 \\ 0 & 718.503845 & 237.299279 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 731.435860 & 2.014288 & 320.486042 \\ 0 & 733.096893 & 240.158381 \\ 0 & 0 & 1 \end{pmatrix}$
R_1	$\begin{pmatrix} 0.999410 & -0.012255 & 0.032091 \\ 0.013624 & 0.998991 & -0.042801 \\ -0.031534 & 0.043213 & 0.998568 \end{pmatrix}$	$\begin{pmatrix} 0.999159 & -0.012466 & 0.039061 \\ 0.013768 & 0.999353 & -0.033221 \\ -0.038621 & 0.033730 & 0.998684 \end{pmatrix}$
t_1	$\begin{pmatrix} -3.299247 \\ -3.979002 \\ 20.786686 \end{pmatrix}$	$\begin{pmatrix} -3.177474 \\ -4.066574 \\ 21.101353 \end{pmatrix}$
D	(0.098037,-1.639184)	(-0.201687 0.039493)
F	7429.61	3771.05

Table 1: Parameters For Image Set 1, R,t for image P1010001s.jpg

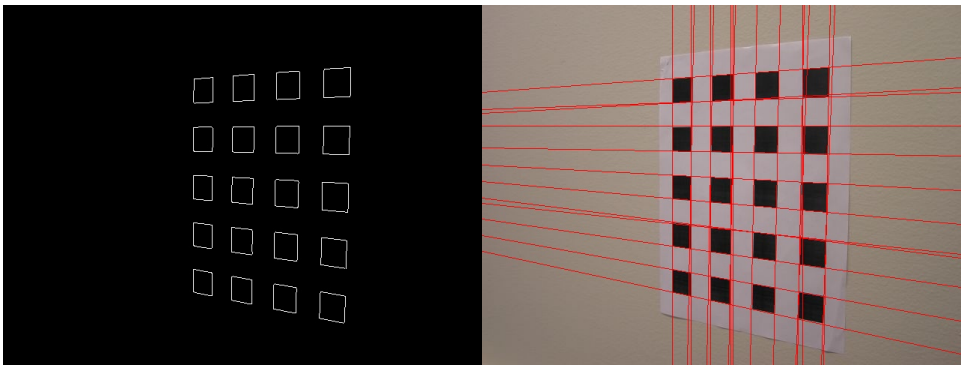
	Initial	Refined
K	$\begin{pmatrix} 644.667493 & -3.084936 & 324.993269 \\ 0 & 646.253247 & 245.778708 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 638.467493 & -3.956473 & 323.982361 \\ 0 & 639.516005 & 250.617438 \\ 0 & 0 & 1 \end{pmatrix}$
R_1	$\begin{pmatrix} 0.996199 & 0.011029 & -0.086402 \\ -0.015785 & 0.998386 & -0.054560 \\ 0.085661 & 0.055716 & 0.994765 \end{pmatrix}$	$\begin{pmatrix} 0.996304 & 0.011689 & -0.085095 \\ -0.016630 & 0.998202 & -0.057588 \\ 0.084269 & 0.058790 & 0.994707 \end{pmatrix}$
t_1	$\begin{pmatrix} -4.320643 \\ -4.884312 \\ 18.535515 \end{pmatrix}$	$\begin{pmatrix} -4.304821 \\ -5.046637 \\ 18.430106 \end{pmatrix}$
D	(0.033083,-0.394379)	(0.112342,-0.941011)
F	3264.81	2634.34

Table 2: Parameters For Image Set 2, R,t for image IMG-20101111-00032.jpg

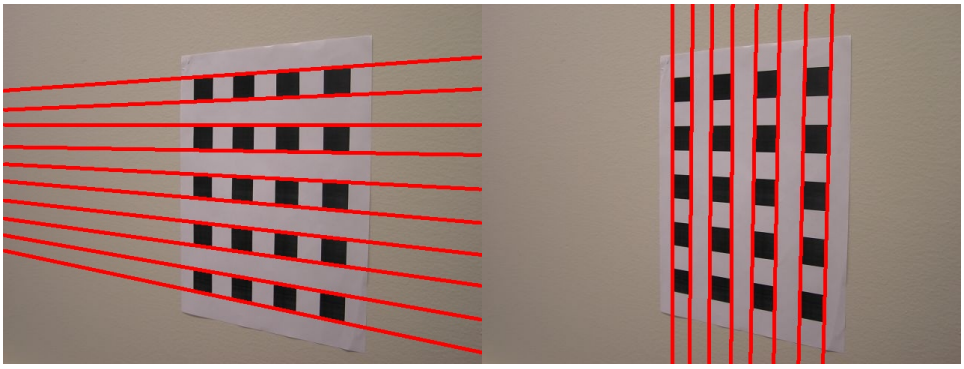


(a) (b)

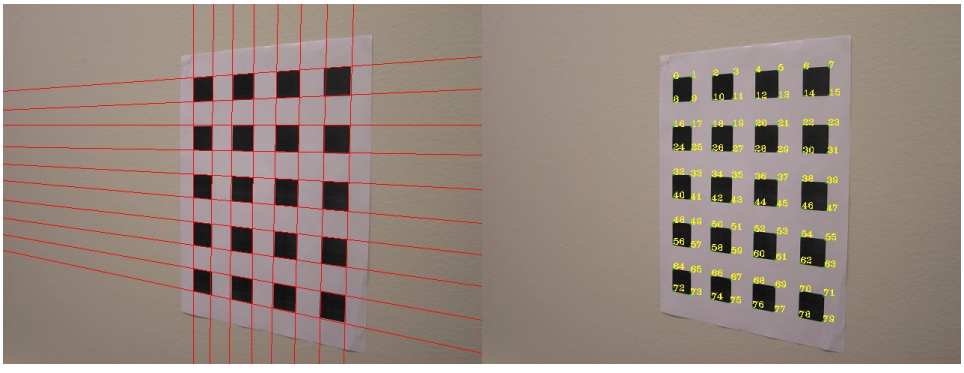
Figure 1: MATLAB LM optimization using 5 images only



(a) Canny Image (b) Hough Lines

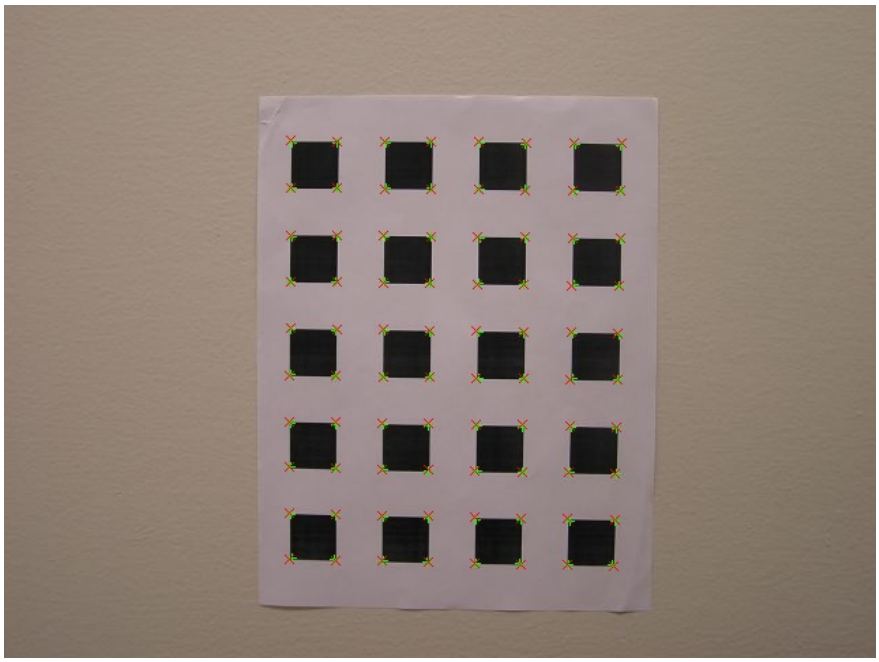


(c) Merged Horizontal Lines (d) Merged Vertical Lines

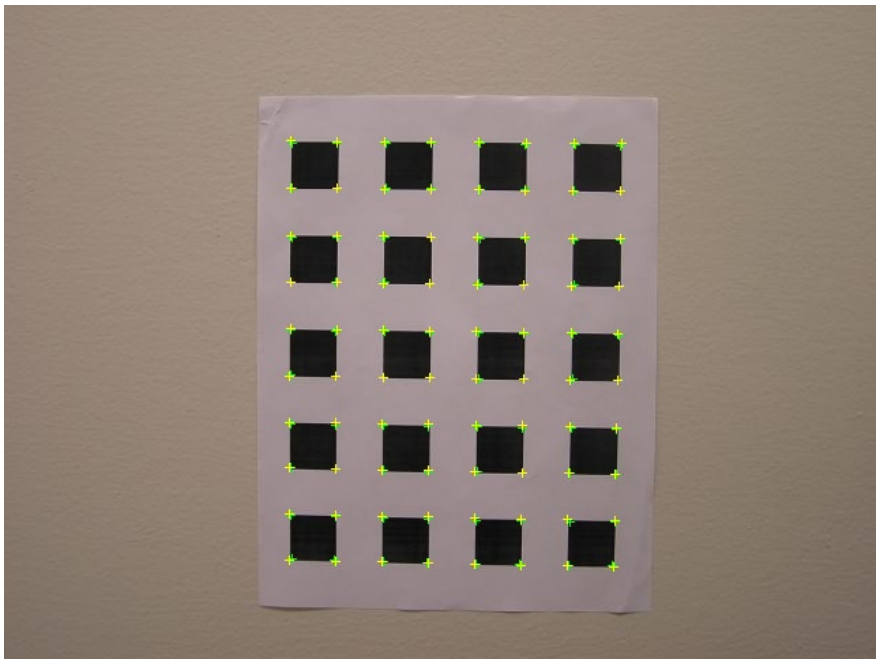


(e) Final Line Set (f) Labeled Corners

Figure 2: Intermediate results

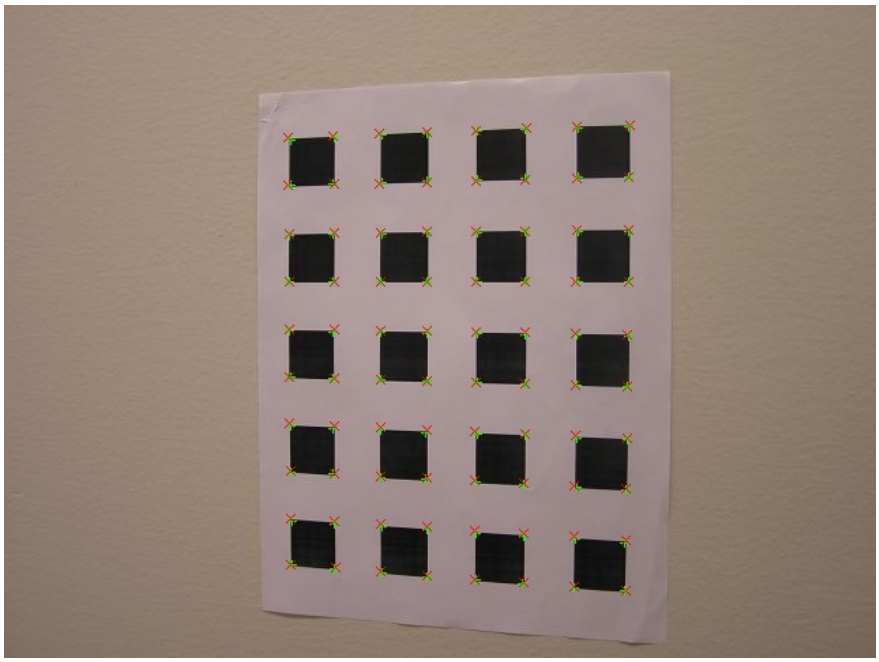


(a) Initial Parameters Reprojected

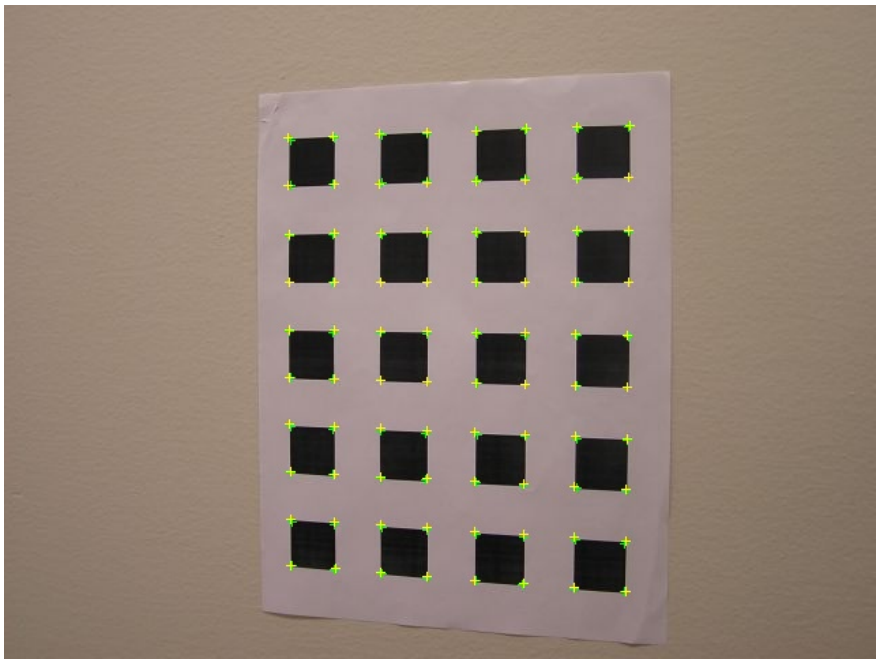


(b) Refined Parameters Reprojected

Figure 3: Image Set 1

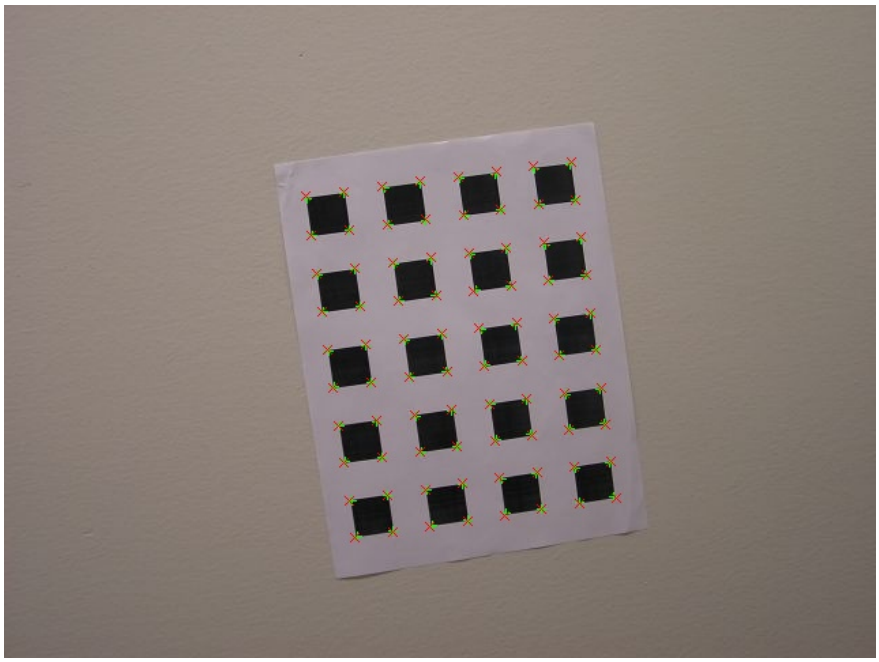


(a) Initial Parameters Reprojected

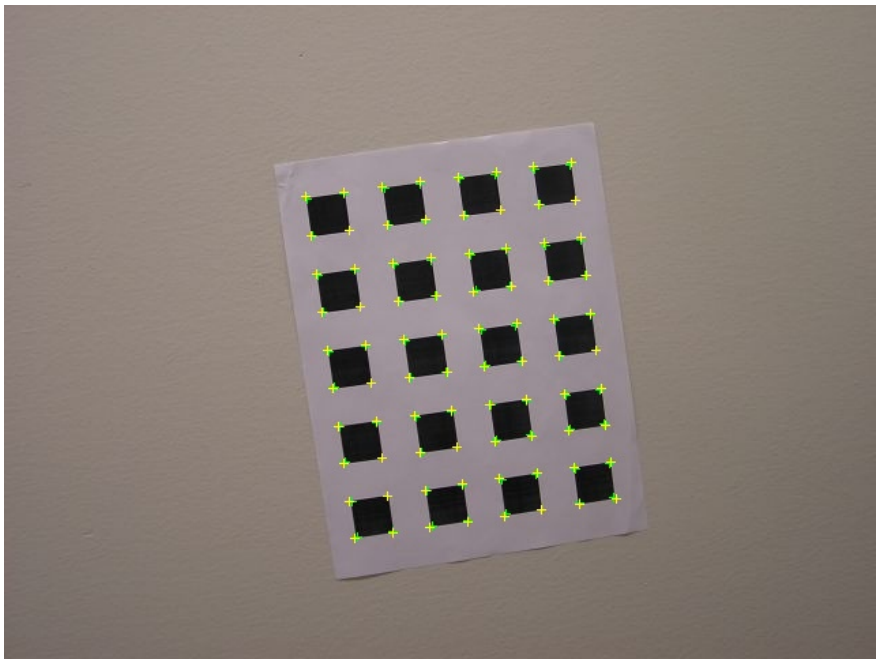


(b) Refined Parameters Reprojected

Figure 4: Image Set 1

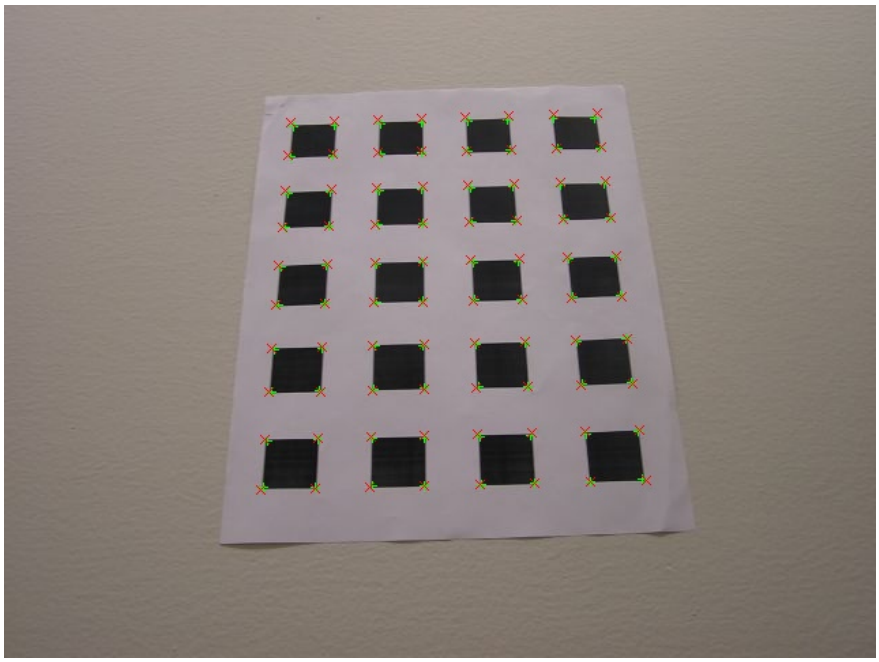


(a) Initial Parameters Reprojected

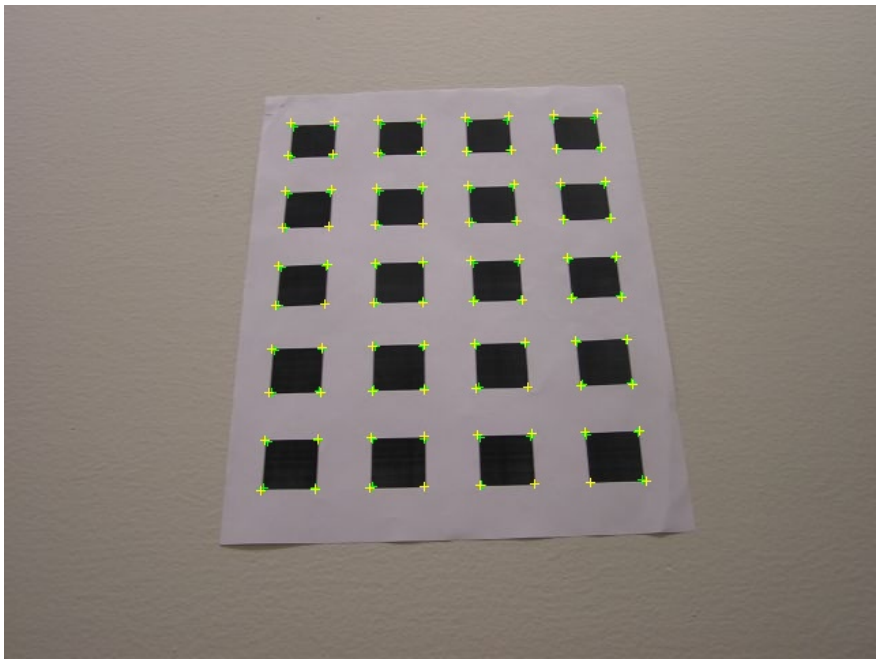


(b) Refined Parameters Reprojected

Figure 5: Image Set 1

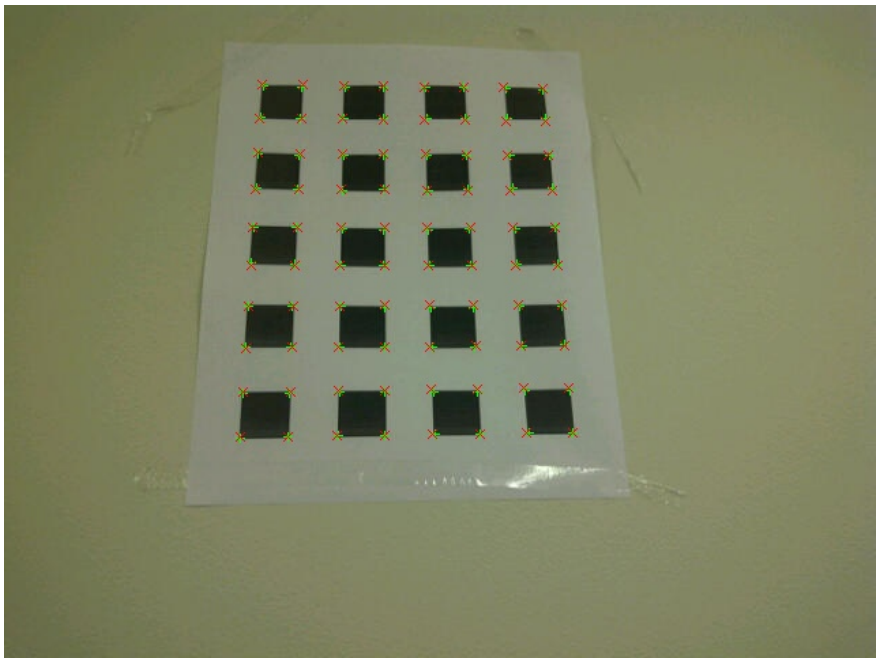


(a) Initial Parameters Reprojected

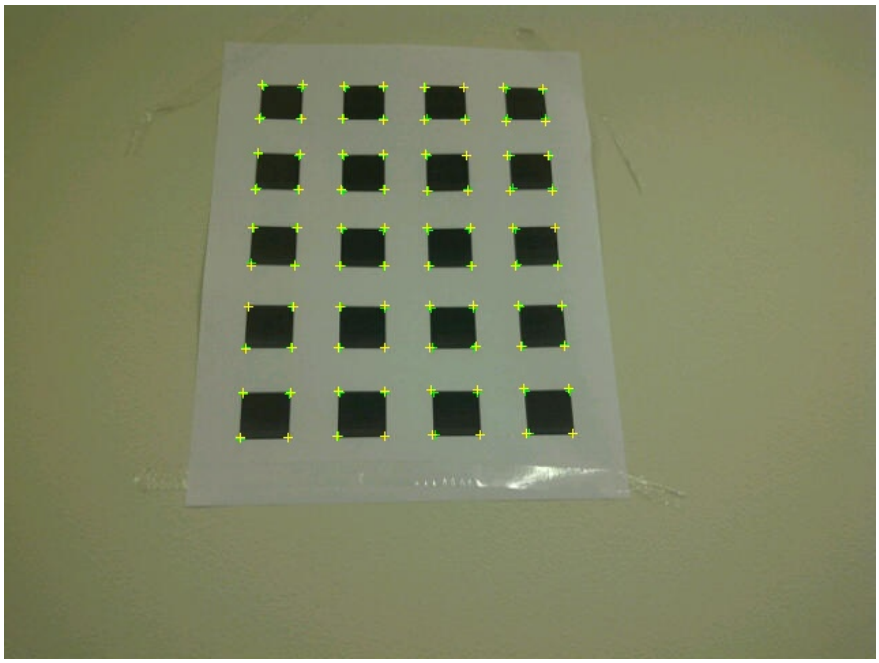


(b) Refined Parameters Reprojected

Figure 6: Image Set 1

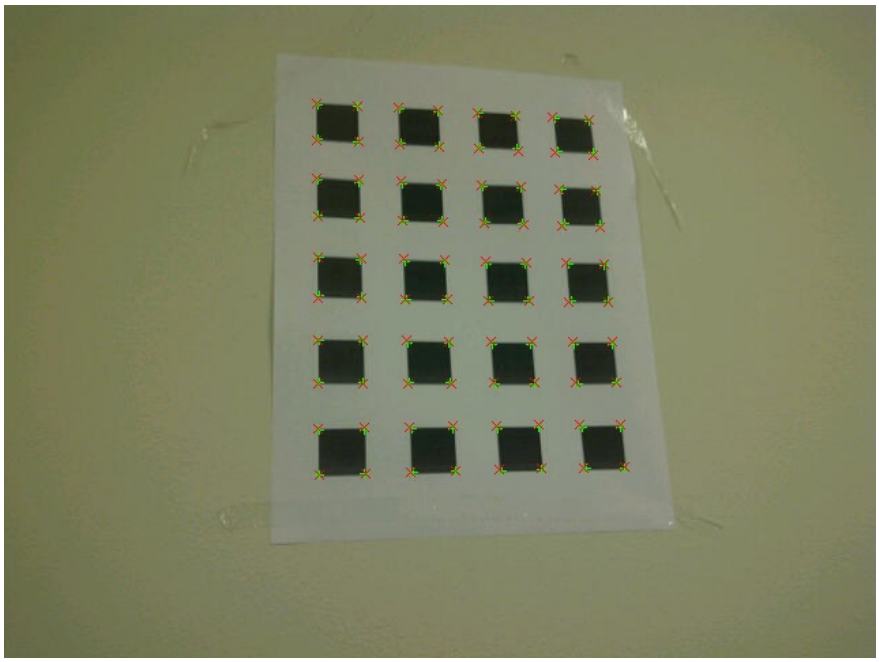


(a) Initial Parameters Reprojected

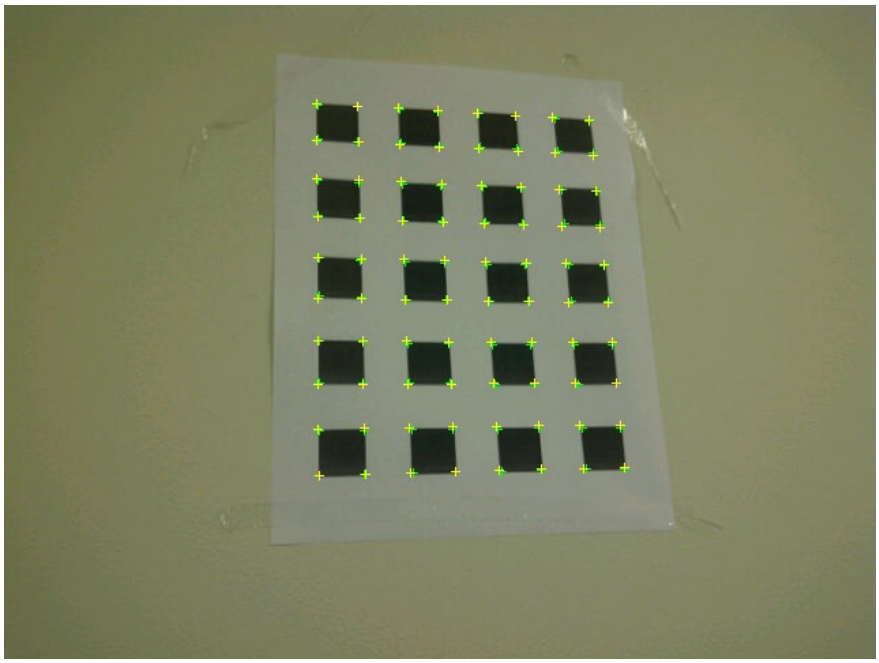


(b) Refined Parameters Reprojected

Figure 7: Image Set 2

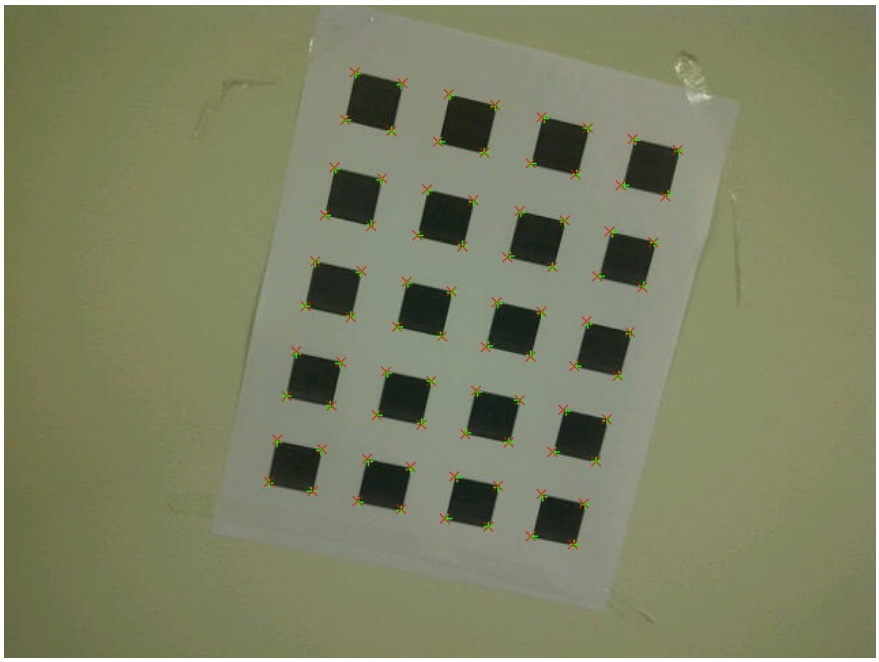


(a) Initial Parameters Reprojected

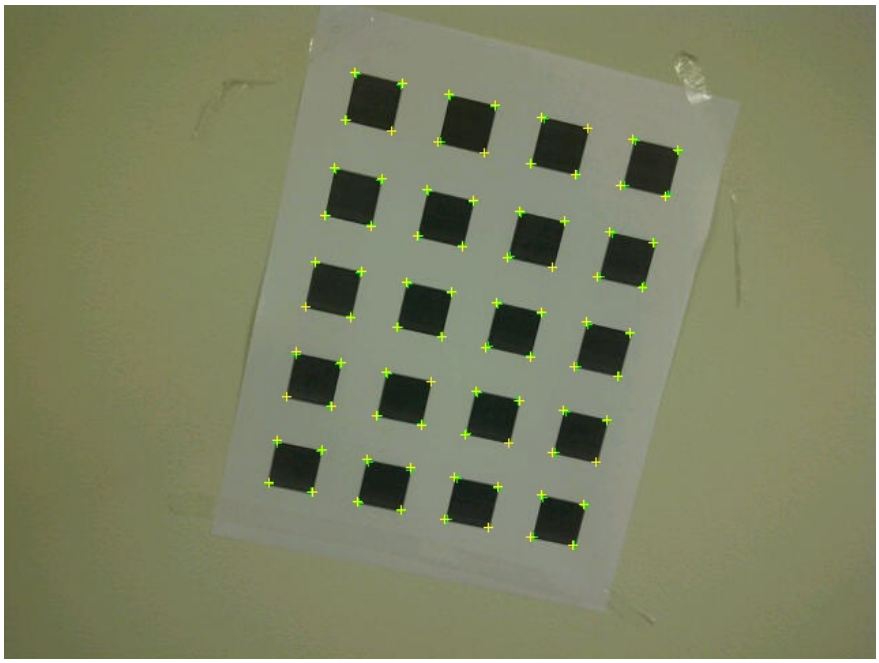


(b) Refined Parameters Reprojected

Figure 8: Image Set 2

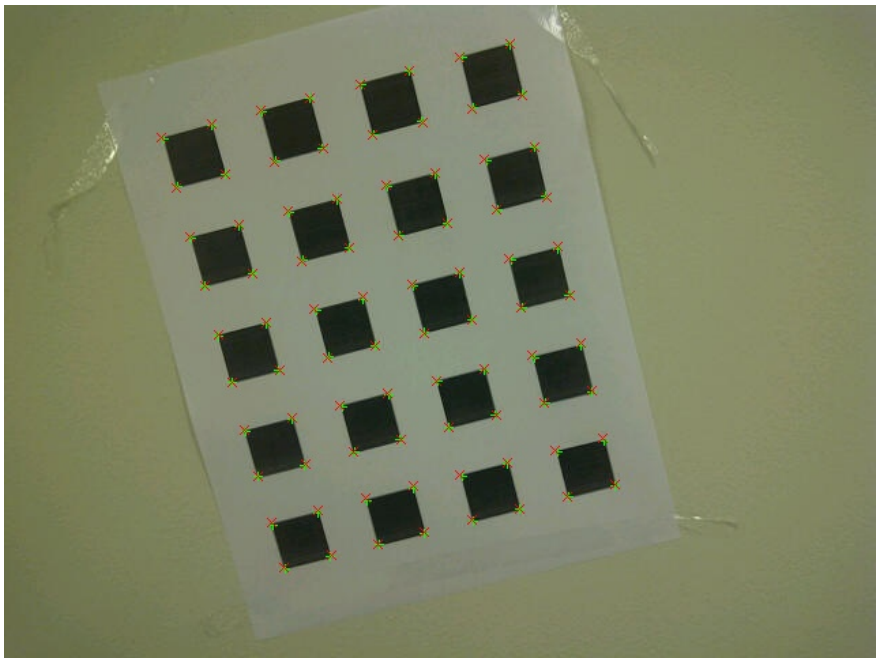


(a) Initial Parameters Reprojected

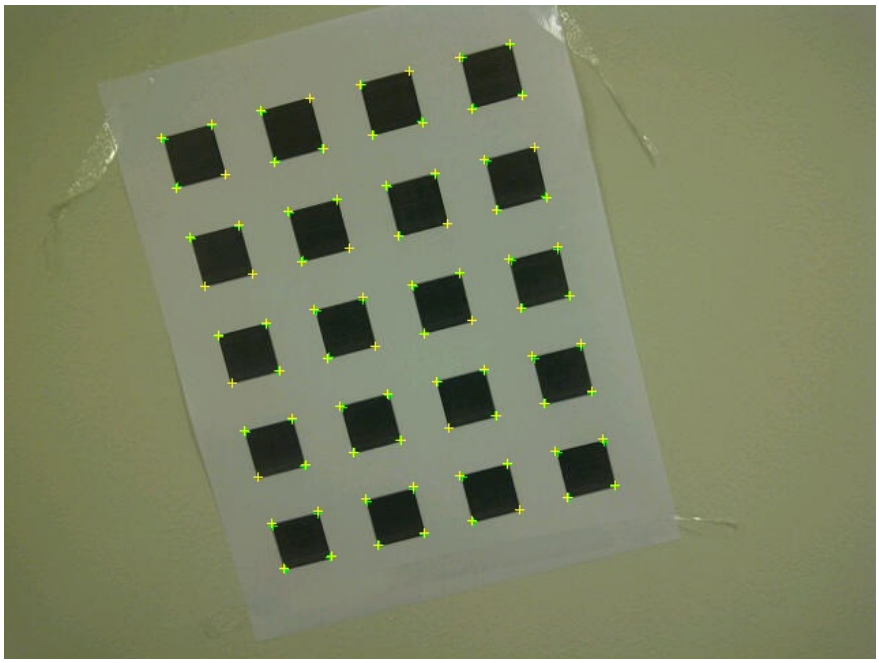


(b) Refined Parameters Reprojected

Figure 9: Image Set 2



(a) Initial Parameters Reprojected



(b) Refined Parameters Reprojected

Figure 10: Image Set 2