EN.601.454/654 Augmented Reality

Assignment 3

Name: Yuliang Xiao

JHED: yxiao39

Date: 09/29/2022

1. Camera Distortion

1.1 What is the goal of undistortion? Why should we undistort an image before processing it? (1 point)

The goal of undistortion is to achieve an overal one-to-one mapping (linear image) Because the image processing is a linear transformation and a distorted image will have errors when applying image processing without undistortion

1.2 Given $f(r) = 1 + 0.057r^2 + 0.00014r^4$, compute and sketch the undistorted image of Fig. 1. (2 points)

According to Fig.1, the distortion center $C = \begin{bmatrix} C_x \\ C_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

For point $\vec{P}_1 = [-0.9129, 0.9129]^T$

$$r = \|(-0.9129, 0.9129)\| = 1.291$$

$$f(r) = 1 + 0.057r^2 + 0.00014r^4 = 1.095$$

$$\vec{P}_{1\,correct} = f(r)\,\vec{P}_1 = 1.095 \cdot \begin{bmatrix} -0.9129 \\ 0.9129 \end{bmatrix} = \begin{bmatrix} -0.9996 \\ 0.9996 \end{bmatrix} = \boxed{\begin{bmatrix} -1 \\ 1 \end{bmatrix}}$$

For point $\vec{P}_2 = [0, 0.9509]^T$

$$r = \|(0, 0.9509)\| = 0.9509$$

$$f(r) = 1 + 0.057r^2 + 0.00014r^4 = 1.052$$

$$\vec{P}_{2\,correct} = f(r)\,\vec{P}_2 = 1.052 \cdot \begin{bmatrix} 0 \\ 0.9509 \end{bmatrix} = \boxed{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}$$

For point $\vec{P}_3 = [0.9129, 0.9129]^T$

$$r = \|(0.9129, 0.9129)\| = 1.291$$

$$f(r) = 1 + 0.057r^2 + 0.00014r^4 = 1.095$$

$$\vec{P}_{3\,correct} = f(r)\,\vec{P}_3 = 1.095 \cdot \begin{bmatrix} 0.9129 \\ 0.9129 \end{bmatrix} = \begin{bmatrix} 0.9996 \\ 0.9996 \end{bmatrix} = \boxed{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}$$

For point $\vec{P_4} = [-0.9509\ ,\ 0\ ]^T$

$$r = \|(-0.9509\ ,\ 0)\| = 0.9509$$

$$f(r) = 1 + 0.057r^2 + 0.00014r^4 = 1.052$$

$$\vec{P}_{4\,\text{correct}} = f(r)\,\vec{P_4} = 1.052\begin{bmatrix} -0.9509 \\ 0 \end{bmatrix} = \boxed{\begin{bmatrix} -1 \\ 0 \end{bmatrix}}$$

For point $\vec{P_5} = [0.9509\ ,\ 0\ ]^T$

$$r = \|(0.9509\ ,\ 0)\| = 0.9509$$

$$f(r) = 1 + 0.057r^2 + 0.00014r^4 = 1.052$$

$$\vec{P}_{5\,\text{correct}} = f(r)\,\vec{P_5} = 1.052 \cdot \begin{bmatrix} 0.9509 \\ 0 \end{bmatrix} = \boxed{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}$$

For point $\vec{P_6} = [-0.9129, -0.9129]^T$

$$r = \|(-0.9129, -0.9129)\| = 1.291$$

$$f(r) = 1 + 0.057r^2 + 0.00014r^4 = 1.095$$

$$\vec{P}_{6\,\text{correct}} = f(r)\,\vec{P_6} = 1.095 \cdot \begin{bmatrix} -0.9129 \\ -0.9129 \end{bmatrix} = \begin{bmatrix} -0.9996 \\ -0.9996 \end{bmatrix} = \boxed{\begin{bmatrix} -1 \\ -1 \end{bmatrix}}$$

For point $\vec{P_7} = [0, -0.9509\ ]^T$

$$r = \|(0, -0.9509)\| = 0.9509$$

$$f(r) = 1 + 0.057r^2 + 0.00014r^4 = 1.052$$

$$\vec{P}_{7\,\text{correct}} = f(r)\,\vec{P_7} = 1.052 \cdot \begin{bmatrix} 0 \\ -0.9509 \end{bmatrix} = \boxed{\begin{bmatrix} 0 \\ -1 \end{bmatrix}}$$

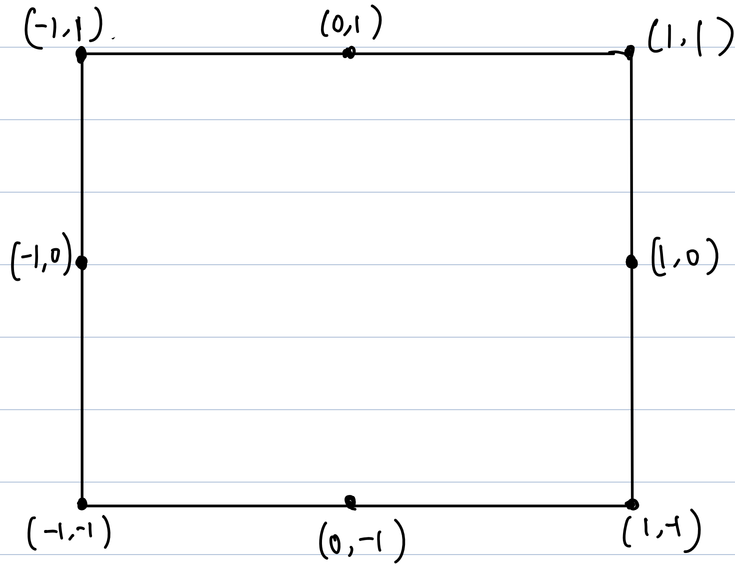
For point $\vec{P_8} = [\ 0.9129, -0.9129]^T$

$$r = \|(0.9129, -0.9129)\| = 1.291$$

$$f(r) = 1 + 0.057r^2 + 0.00014r^4 = 1.095$$

$$\vec{P}_{8\,\text{correct}} = f(r)\,\vec{P_8} = 1.095 \cdot \begin{bmatrix} 0.9129 \\ -0.9129 \end{bmatrix} \approx \begin{bmatrix} 0.9996 \\ -0.9996 \end{bmatrix} = \boxed{\begin{bmatrix} 1 \\ -1 \end{bmatrix}}$$

see next page


(-1,1)
(0,1)
(1,1)
(-1,0)
(1,0)
(-1,-1)
(0,-1)
(1,-1)

1.3 A fisheye camera is a camera that can capture images with a field of view of more than 180^o. (More details on *Wikipedia*). We are also interested in undistorted images from fisheye camera. Will Eq. 1 work for modeling the distortion? Why or why not? (2 points)

The equation 1 doesn't work for modeling the distortion. Because in the fisheye camera, there's existing tangential distortion. But in Eq.1, it doesn't contain the expression for correcting tangential distortion

1.4 *(UG Optional)* Typically, we use $f(r) = 1+k_1r^2+k_2r^4+k_3r^6$ to model the radial distortion. Note that only even power terms are used in the polynomial. Why do we not use the odd order terms? (1 point)

As far as I'm thinking, because the function domain r is $[0, r_{max}]$ and is always positive, there's no need to use odd term

Problem 2

1-IntelRGB

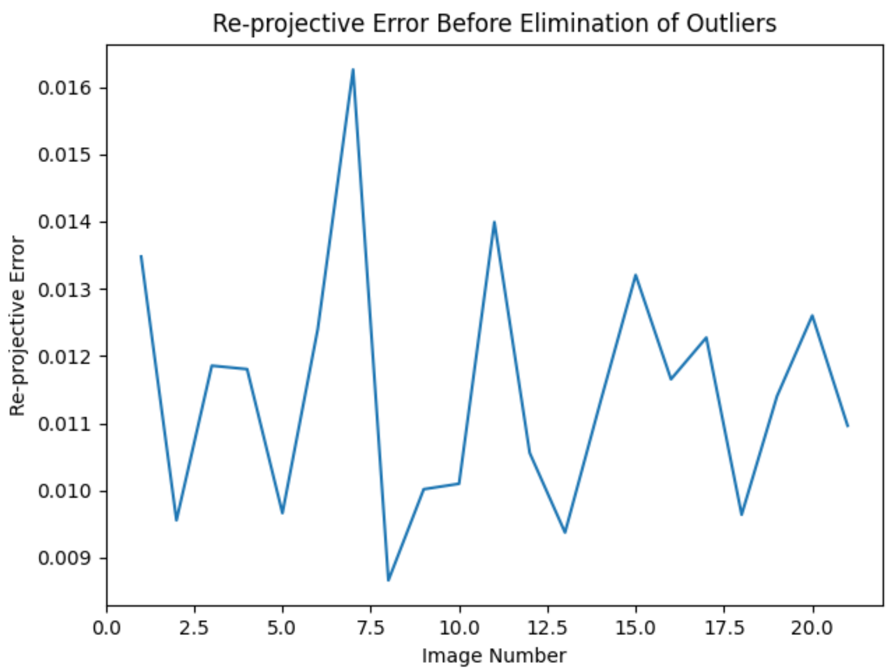
Results

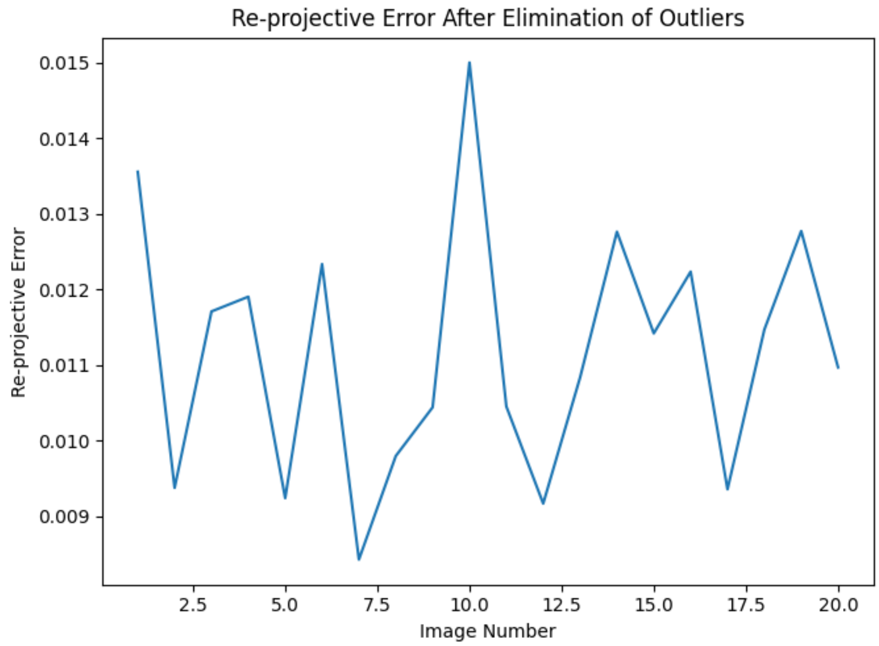
Mean Re-projective Error: 0.011160417930001304

Camera Intrinsic Matrix:
[[615.91794656 0. 317.95220817]
[0. 616.36127696 249.60556603]
[0. 0. 1.]]

Distortion Coefficients:
[[0.13701813 -0.6886125 0.00419846 -0.00135203 0.15210989]]

Re-projective Errors





Elimination of Outliers

1. In the OpenCV library, it has a function *findChessboardCorners().* which could detect the chessboard pattern. If the image quality is good, the function will return True. Otherwise, False will be returned due the poor image quality. This function could eliminate those low-quality images before calibration.
2. In the general data preprocessing, outliers are those values who have large distance or variation away from the mean value of a population. In our case, we compute a relative variance between each re-projective error and mean error. If the relative variance is larger than 25%, then this image can be treated as outlier and can be eliminated. The equation is below:

$$absolute(Error_i - Error_{mean}) \,/\, Error_{mean}$$

where $Error_{mean}$ is the mean re-projective error and $Error_i$ is the re-projective error of i-th image.

2-Intel (IR – Camera 1, RGB – Camera 2)

Results

Root Mean Square Error: 0.17041647505420962

Camera 1 Intrinsic Matrix:
```
[[475.75302667   0.         322.05865013]
 [  0.         475.72260583 235.56544396]
 [  0.           0.           1.        ]]
```

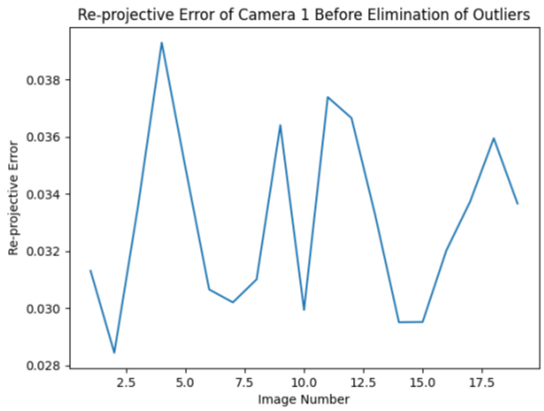
Camera 1 Distortion Coefficients:
```
[[-1.41980593e-01  1.31273402e-01  2.46139668e-04  1.35830589e-03 -4.13529137e-01]]
```

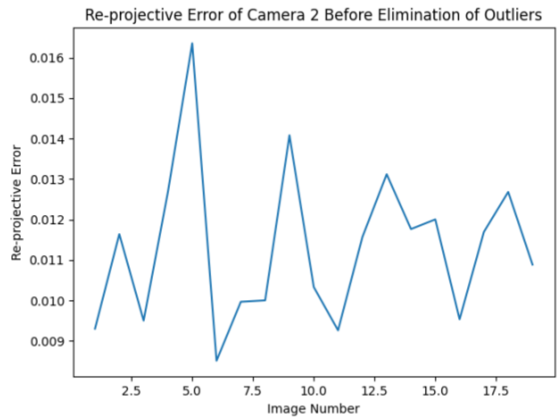
Camera 2 Intrinsic Matrix:
```
[[615.36616501   0.         317.94872177]
 [  0.         615.81319092 249.39775434]
 [  0.           0.           1.        ]]
```

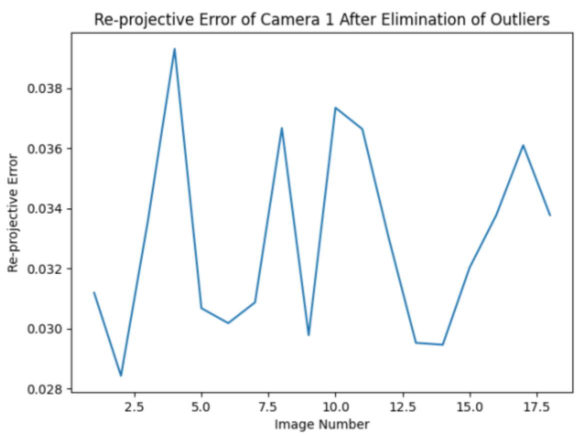
Camera 2 Distortion Coefficients:
```
[[ 0.13828937 -0.69942098  0.00423871 -0.00138966  0.18209954]]
```

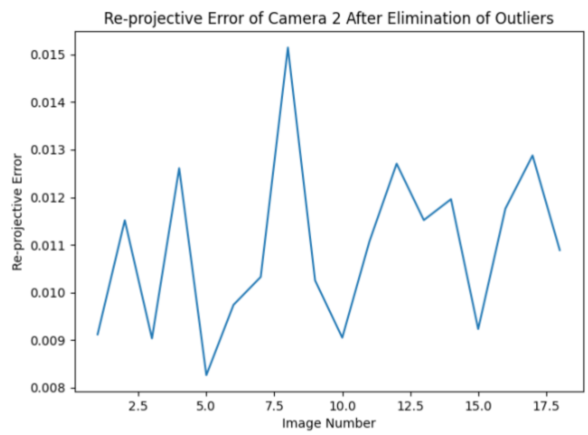
Projective Matrix:
```
[[   1.           0.           0.        -347.47930145]
 [   0.           1.           0.        -257.98200989]
 [   0.           0.           0.         509.55798306]
 [   0.           0.         -41.85572066    0.        ]]
```

Re-projective Errors









Elimination of Outliers

1. In the OpenCV library, it has a function *findChessboardCorners().* which could detect the chessboard pattern. If the image quality is good, the function will return True. Otherwise, False will be returned due the poor image quality. This function could eliminate those low-quality images before calibration.
2. In the general data preprocessing, outliers are those values who have large distance or variation away from the mean value of a population. In our case, we compute a relative variance between each re-projective error and mean error. If the relative variance is larger than 25%, then this image can be treated as outlier and can be eliminated. The equation is below:

$$absolute(Error_i - Error_{mean}) \,/\, Error_{mean}$$

where $Error_{mean}$ is the mean re-projective error and $Error_i$ is the re-projective error of i-th image. In this case, we have two cameras, either one has the relative variance larger than 25% of one image, another one will be automatically treated as outlier because stereo camera calibration needs to calibrate two cameras at same time.

3-Kinect2 (Bonus) (IR – Camera 1, RGB – Camera 2)

Results

Root Mean Square Error: 0.13332708195314444

Camera 1 Intrinsic Matrix:
```
[[362.43470722   0.         253.65318911]
 [  0.         362.69442611 207.31937449]
 [  0.           0.           1.        ]]
```

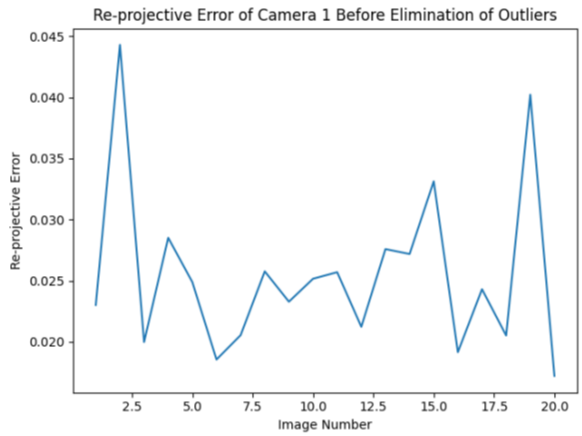
Camera 1 Distortion Coefficients:
```
[[ 0.11342378 -0.26691858  0.00115919 -0.00096709  0.03691603]]
```

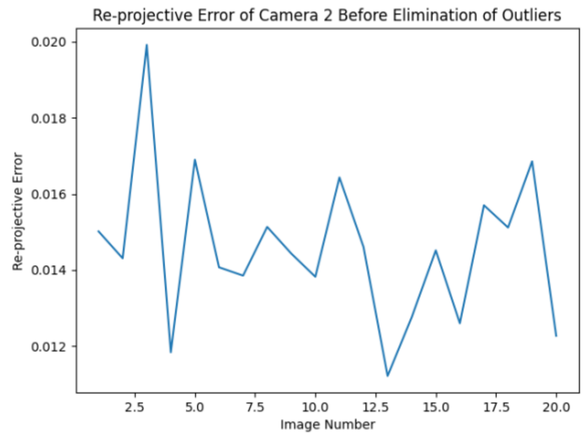
Camera 2 Intrinsic Matrix:
```
[[535.55126944   0.         498.63317955]
 [  0.         536.15039653 281.69878589]
 [  0.           0.           1.        ]]
```

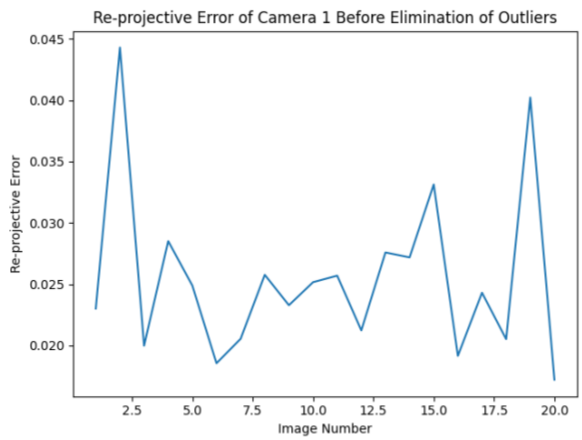
Camera 2 Distortion Coefficients:
```
[[ 0.0795811  -0.13356346  0.00133061 -0.00026759  0.05490394]]
```

Projective Matrix:
```
[[   1.           0.           0.        -375.0845871 ]
 [   0.           1.           0.        -261.95816422]
 [   0.           0.           0.         405.5177435 ]
 [   0.           0.          19.20552215   -0.        ]]
```

Re-projective Errors







Elimination of Outliers

1. In the OpenCV library, it has a function *findChessboardCorners().* which could detect the chessboard pattern. If the image quality is good, the function will return True. Otherwise, False will be returned due the poor image quality. This function could eliminate those low-quality images before calibration.
2. In the general data preprocessing, outliers are those values who have large distance or variation away from the mean value of a population. In our case, we compute a relative variance between each re-projective error and mean error. If the relative variance is larger than 25%, then this image can be treated as outlier and can be eliminated. The equation is below:

$$absolute(Error_i - Error_{mean}) \,/\, Error_{mean}$$

where $Error_{mean}$ is the mean re-projective error and $Error_i$ is the re-projective error of i-th image. In this case, we have two cameras, either one has the relative variance larger than 25% of one image, another one will be automatically treated as outlier because stereo camera calibration needs to calibrate two cameras at same time.