

# RCVWS 2020 Calibration 실습

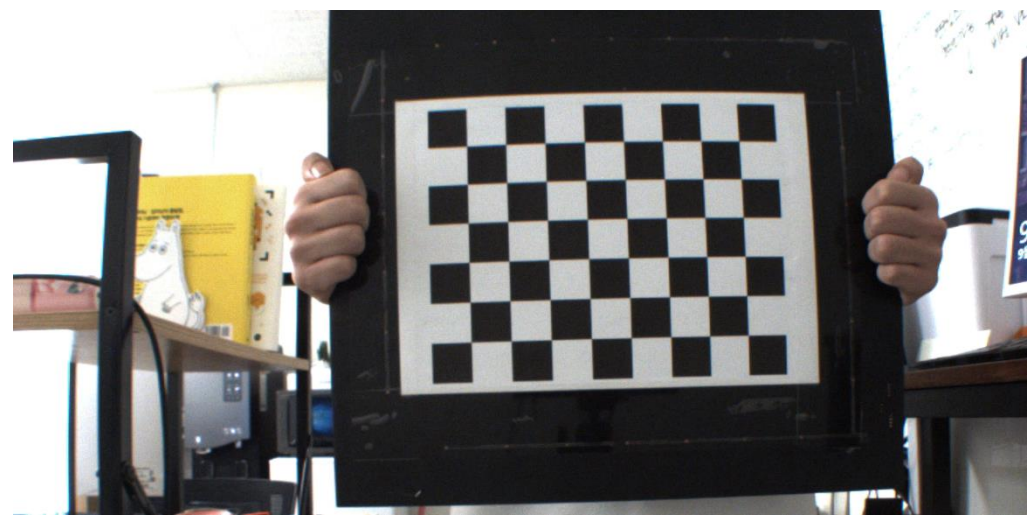
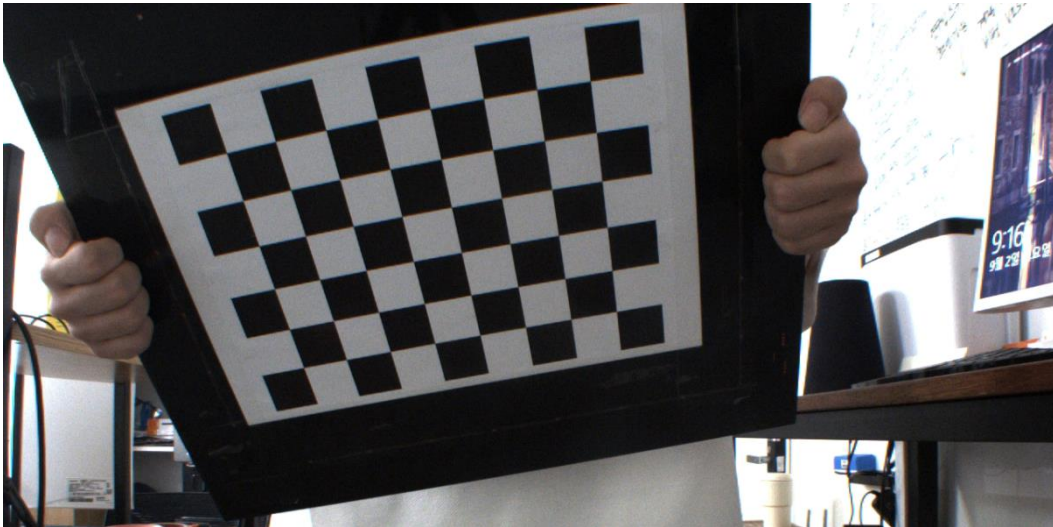
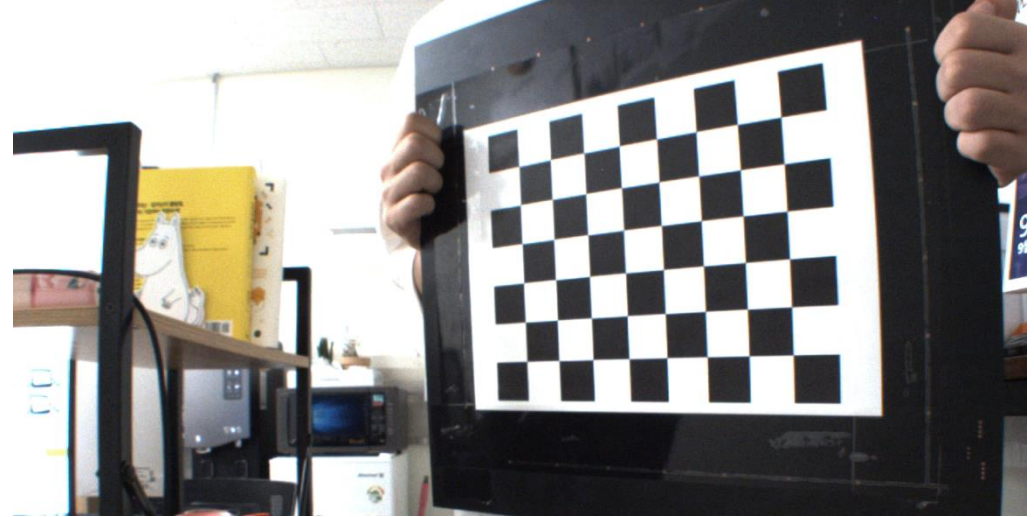
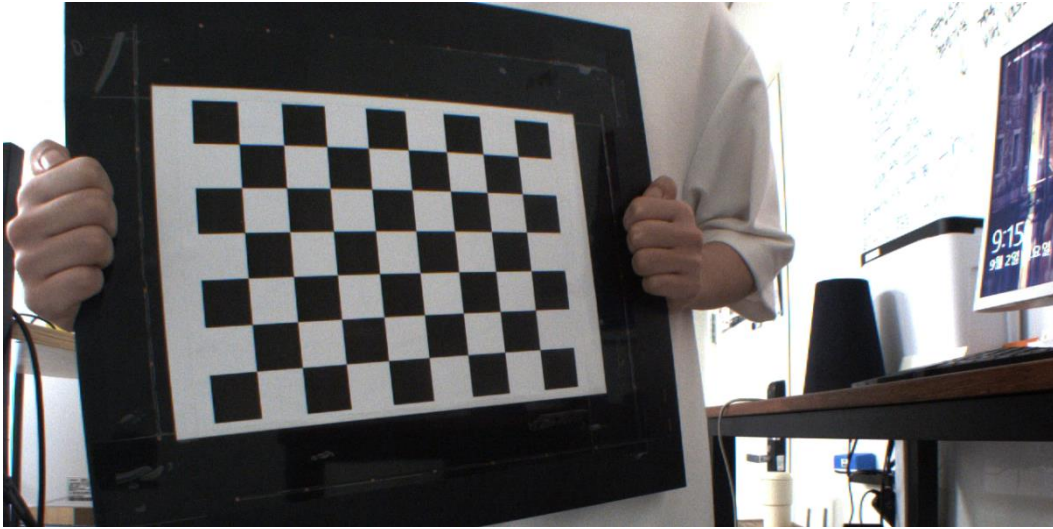
신정민

---

# **Calibration & Disparity**

---

# 1. 사진 촬영



## 2. Matlab for calibration

Caltech calibration 홈페이지에 가서 matlab file 을 다운받습니다.

[http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)

- [Calibration examples](#)
- [Description of the calibration parameters](#)
- [Description of the functions in the calibration toolbox](#)
- [Doing your own calibration](#)
- [Undocumented features of the toolbox](#)
- [References](#)
- [A few links related to camera calibration](#)

### ★ System requirements

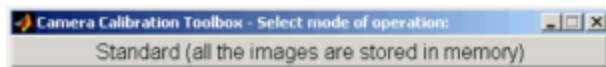
This toolbox works on Matlab 5.x to Matlab 8.x on Windows, Unix and Linux systems and does not require any specific Matlab toolbox (for example, the optimization toolbox is not required).

**Note:** Please help me maintaining this toolbox by reporting them to me. Include in the email subject the type of the bug, and copy in the body the complete error message. Thank you!

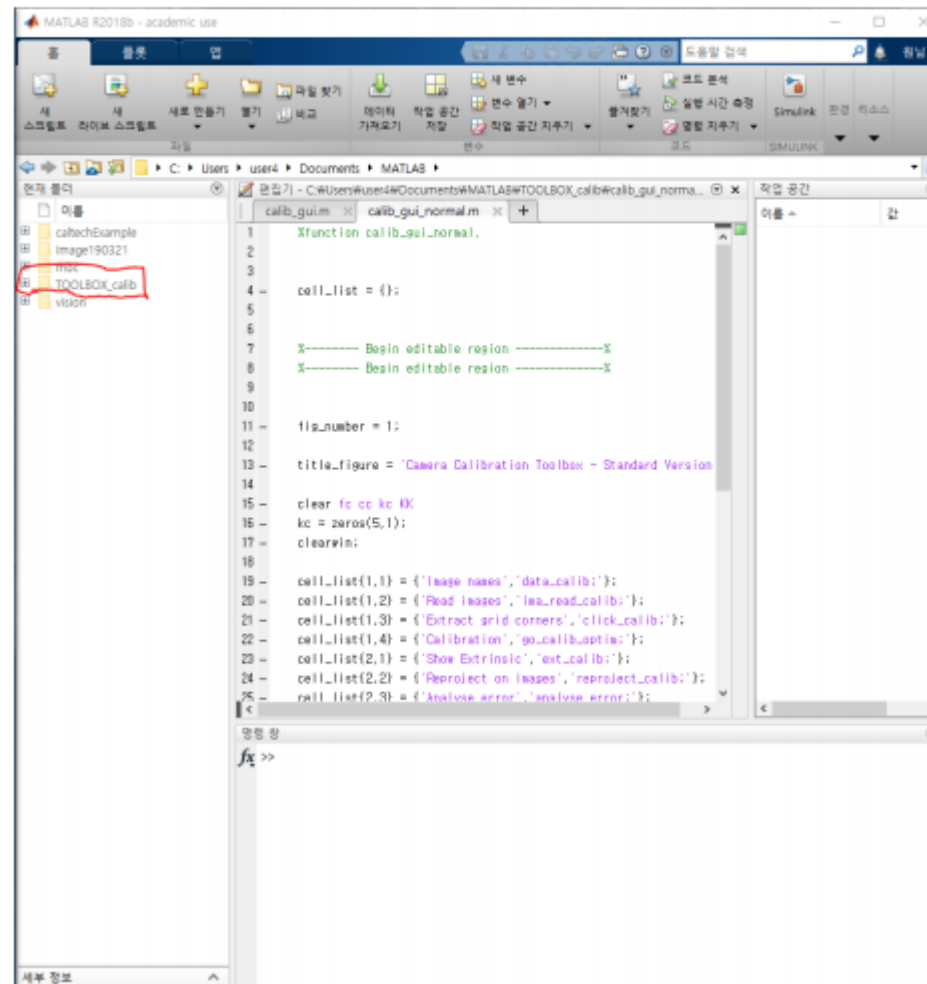
### ★ Getting started

- Go to the [download page](#), and retrieve the latest version of the complete camera calibration toolbox for Matlab.
- Store the individual matlab files (.m files) into a unique folder **TOOLBOX\_calib** (default folder name).
- Run Matlab and add the location of the folder **TOOLBOX\_calib** to the main matlab path. This procedure will let you call any of the matlab toolbox functions from anywhere. Under Windows, this may be easily done by using the path editing menu. Under Unix or Linux, you may use the command **path** or **addpath** (use the **help** command for function description).
- Run the main matlab calibration function **calib\_gui** (or **calib**).

A mode selection window appears on the screen:

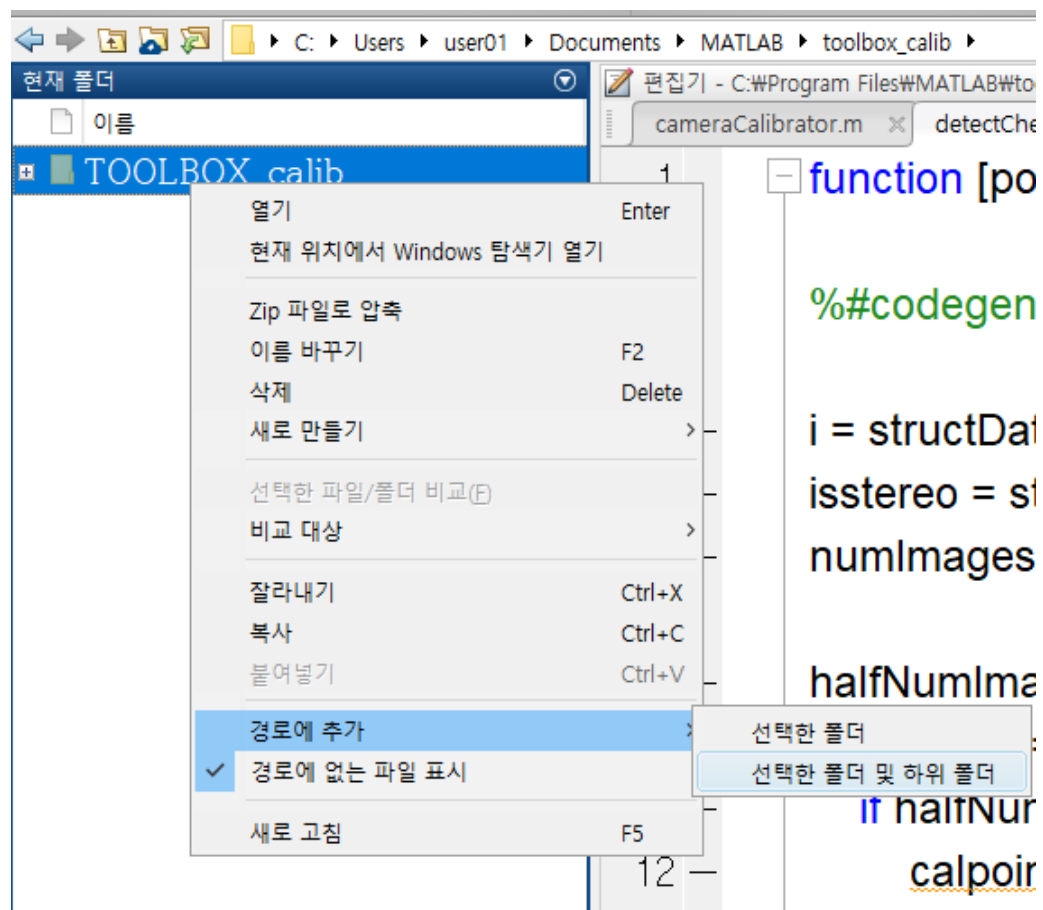


Matlab 을 실행하고 좌측에 보이는 현재폴더 영역에 다운 받은 파일을 드래그해서 넣습니다.





## 2. Matlab for calibration



TOOLBOX\_calib 폴더를 우측클릭->경로에 추가->선택한 폴더 및 하위폴더 선택

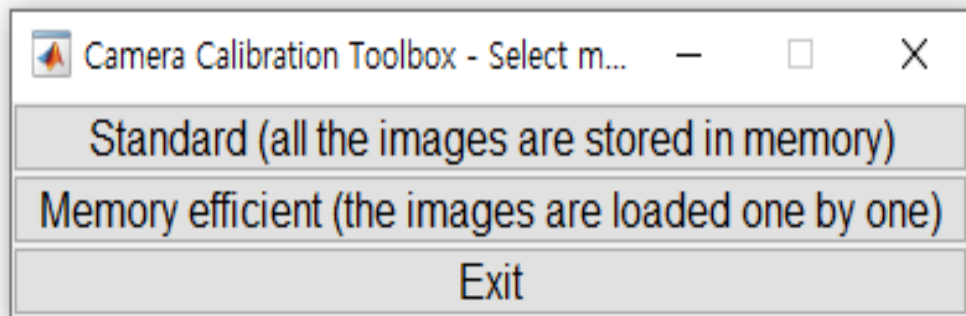


TOOLBOX\_calib 폴더 내부에 새로운 폴더를 하나 형성하여 보정용 사진들을 저장합니다.

Caltech toolbox의 경우 ras, bmp, tif, pgm, jpg, ppm형식의 이미지만을 읽을 수 있으니 보정용 사진들의 포맷을 위에 형식으로 맞춰야만 합니다. 주로 tif 형식을 사용합니다.

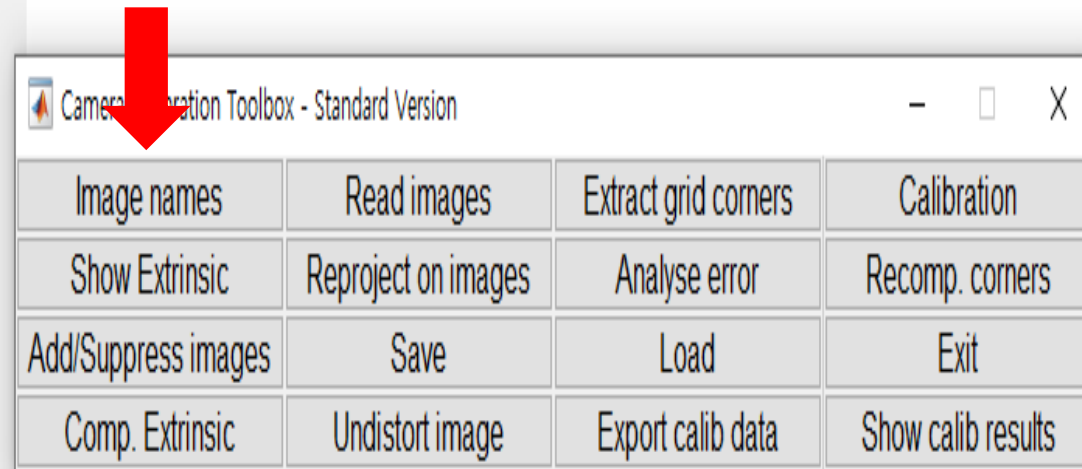
또한 이미지의 이름은 동일하게, 순서는 1부터 차례대로 저장하는 것이 좋습니다.

## 2.1 Camera Calibration ToolBox for Matlab



새 폴더에 보정용 사진들까지 저장을 했다면, 이제 TOOLBOX\_calib 폴더 내부에 들어간 후, 명령창에 calib\_gui를 입력합니다. 잠시 후 함수가 실행되어 위와 같은 창이 나타납니다.

▶ Memory efficient창은 이미지를 하나씩 읽어들이기 때문에 저사양 컴퓨터일때 사용하면 좋지만, 대부분은 Standard 모드를 사용합니다.



먼저 toolbox에 사용하기 위한 이미지들을 읽어들이기 위해 화살표 방향에 버튼을 클릭합니다.

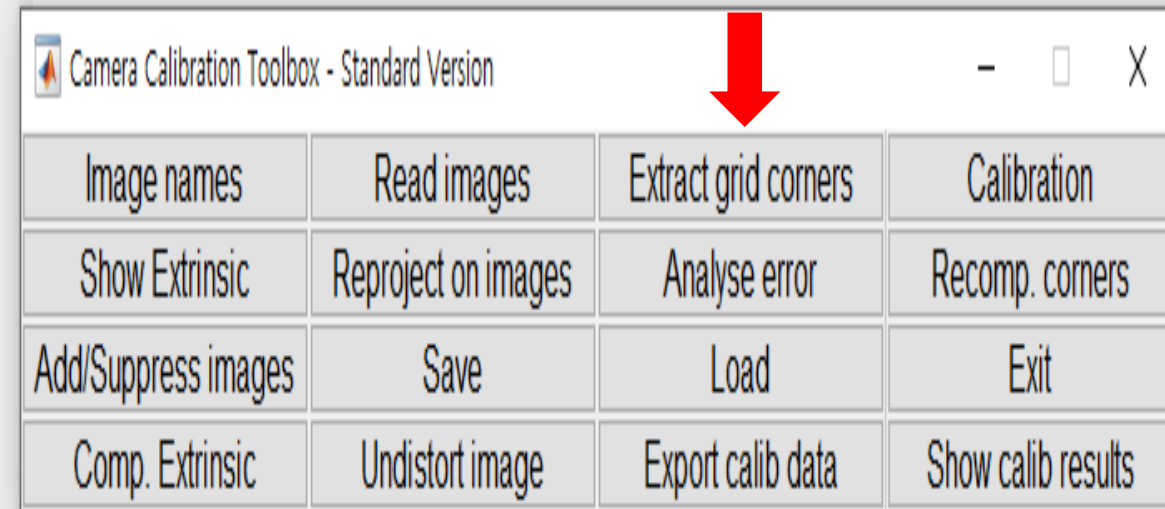
※ 현재경로에 대한 이미지를 찾아서 읽기 때문에 Image names 버튼을 클릭하기 전, 매트랩 좌측에 현재폴더 경로가 보정용 사진이 저장된 폴더여야만 합니다.

## 2.1 Camera Calibration ToolBox for Matlab

Basename camera calibration images (without number nor suffix): LEFT  
Image format: ([]='r'='ras', 'b'='bmp', 't'='tif', 'p'='pgm', 'j'='jpg', 'm'='ppm') t

버튼을 눌렀다면 명령창에 다음과 같이 이미지의 이름과 형식을 물어보는데, 보정용 이미지의 이름과 포맷형식을 입력해주면 이미지를 읽어들이니다.

Ex) 위에 경우에는 이미지의 이름이 LEFT####.tif  
이므로, 이미지의 이름(LEFT)와 format(t='tif')를 입력한 모습.



영상이 잘 읽혔다면 이제 다시 GUI 창으로 돌아가, Extract grid corners 버튼을 클릭합니다.

## 2.2 Extract grid corners

Extraction of the grid corners on the images

Number(s) of image(s) to process ([] = all images) =

Window size for corner finder (wintx and winty):

wintx ([] = 5) = 1

winty ([] = 5) = 1

Window size = 3x3

Do you want to use the automatic square counting mechanism (0=[]=default)  
or do you always want to enter the number of squares manually (1,other)? 1

- ▶ 위에서부터 차례대로, 몇장의 영상을 사용할 것인지, window의 크기 조정, square 개수 자동카운팅 기능 설정을 물어봅니다.
- ▶ 모든 이미지를 사용하기 위해서는 여백을 입력하며, 구간을 입력하여 사용자가 원하는 영상만을 사용할 수도 있습니다.
- ▶ 저희가 사용한 패턴판에서는 윈도우의 크기가 클수록 점이 코너에 정확히 위치하지 않으므로, 제일 작은 값인 1을 입력해주는 것이 좋습니다.
- ▶ 자동카운팅 기능도 역시, 저희 패턴판에는 적합하지 않으므로 꺼주시는 것이 좋습니다.



## 2.2 Extract grid corners

이제 패턴판에 각 코너에 좌표값을 할당해줘야합니다.

▶ 이때 주의해야할 점은 **Stereo pair** 이미지에서 동일한 코너에 동일한 좌표값이 할당되어야 하므로, 처음 영상에서 원점(0,0)으로 잡은 코너는 그 뒤에 모든 영상에서 원점을 정할때 처음 원점으로 잡았던 코너와 같은 코너로 두어야합니다.

▶ 제일 무난한 방식은 원점인 코너(0,0)를 기준으로  
(0,0) -> (x\_max,0) -> (x\_max,y\_max) -> (0,y\_max) 순서(시계방향)로  
마우스클릭을 통해 할당하는 것입니다.

## 2.2 Extract grid corners

Number of squares along the X direction ([]) = 5

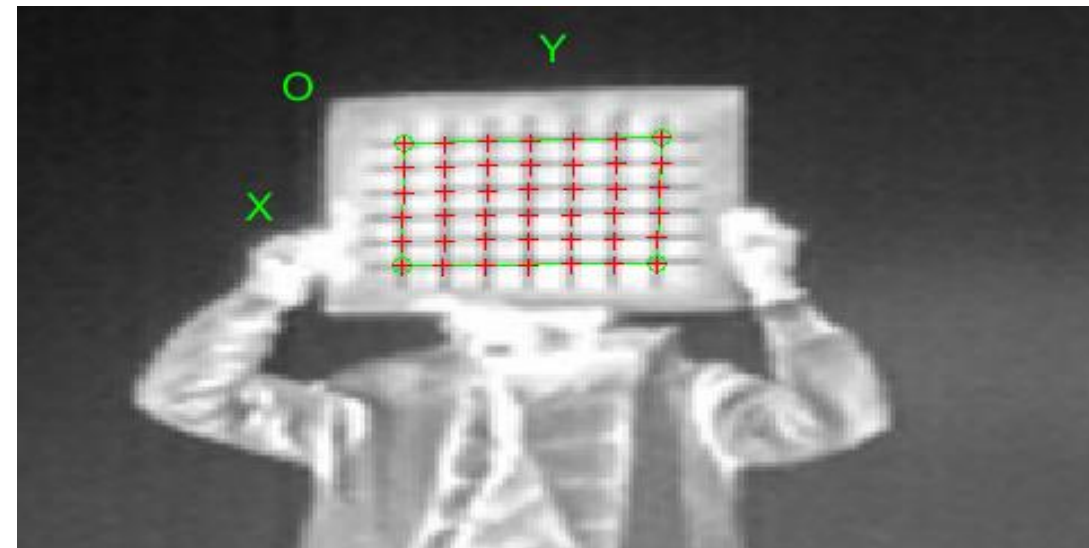
Number of squares along the Y direction ([]) = 6

Size dX of each square along the X direction ([]) = 100mm = 41

Size dY of each square along the Y direction ([]) = 100mm = 41

패턴판에서 4개의 모서리부분에 좌표를 할당했으면, x방향, y방향에 사각형이 몇 개인지와 각 사각형의 가로세로 길이를 입력해줍니다.

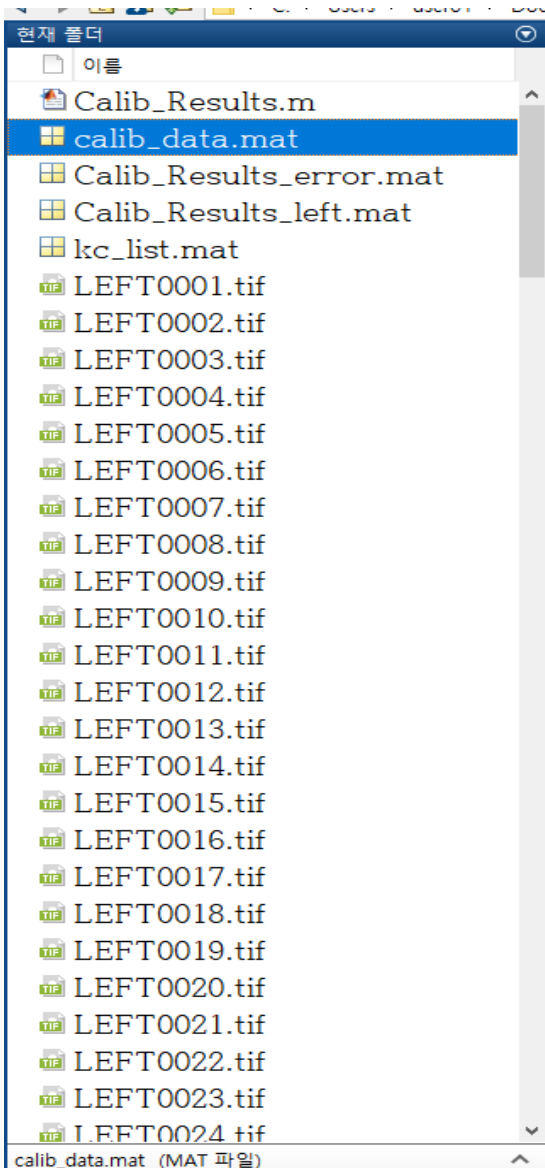
※ 사각형의 가로세로 길이는 첫 이미지에서만 물어보고, 2번째 이미지부터는 길이를 입력할 필요 없이 처음 입력한 길이에 맞추어 자동 할당됩니다.



왼쪽의 과정을 거치면 위에 사진과 같이 각 코너에 빨간 십자가가 할당됩니다.

빨간 십자가는 코너에 정확하게 위치할수록 좋으며 왜곡계수(kc)값을 실험적으로 조정함으로써, 코너에 정확히 위치할 수 있게끔 맞춰줄 수 있습니다.

## 2.2 Extract grid corners



모든 보정용 사진에 대해 좌표값을 할당하여 코너를 추출하였다면 현재 폴더에 calib\_data.mat이라는 데이터가 자동으로 저장됩니다.

이 데이터는 읽어드린 모든 영상 속 패턴판에 할당된 좌표값이 저장되어있는 데이터입니다.

Caltech\_toolbox를 통해 패턴판의 좌표값을 구하여 저장했다면 이제 Matlab에서 제공하는 cameraCalibration app을 사용합니다.

Matlab에서 제공하는 cameraCalibrator app은 vision toolbox를 다운받아야합니다.

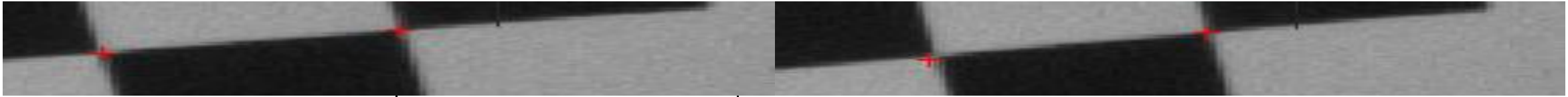
※이때 매트랩 버전은 **반드시 R2019a**를 사용하셔야 합니다.

cameraCalibrator app의 특징은, 보정용 영상과 영상 속 패턴판의 간격(정사각형의 길이)만을 입력하면 일일이 코너에 좌표를 찍어줘야하는 Caltech\_toolbox와는 달리 자동으로 패턴판 속 모든 코너에 대하여 좌표를 할당해줍니다.

하지만 Thermal 카메라로 촬영한 저희의 패턴판은 cameraCalibrator app에서 자동으로 코너를 검출하지 못하므로 사용이 불가능합니다.

이를 해결하고자 기존 Calibrator app속에 코드를 수정하여, Caltech\_toolbox를 통해 얻은 좌표값(좌측 calib\_data.mat)을 load하여 사용합니다.

## 2.2 Extract grid corners



If the guessed grid corners (red crosses on the image) are not close to the actual corners,  
it is necessary to enter an initial guess for the radial distortion factor  $k_c$  (useful for subpixel detection)

Need of an initial guess for distortion? ([]=no, other=yes) 1

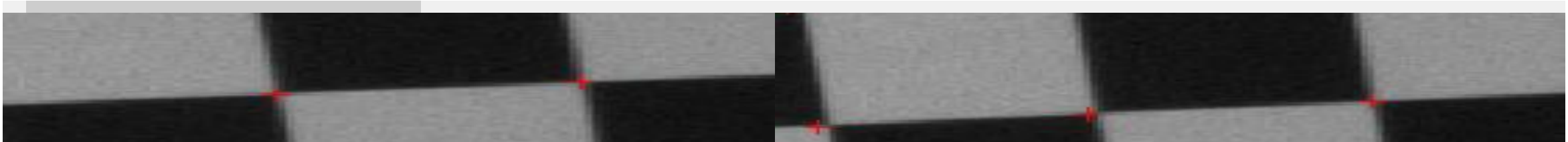
Use number of iterations provided

Use focal provided

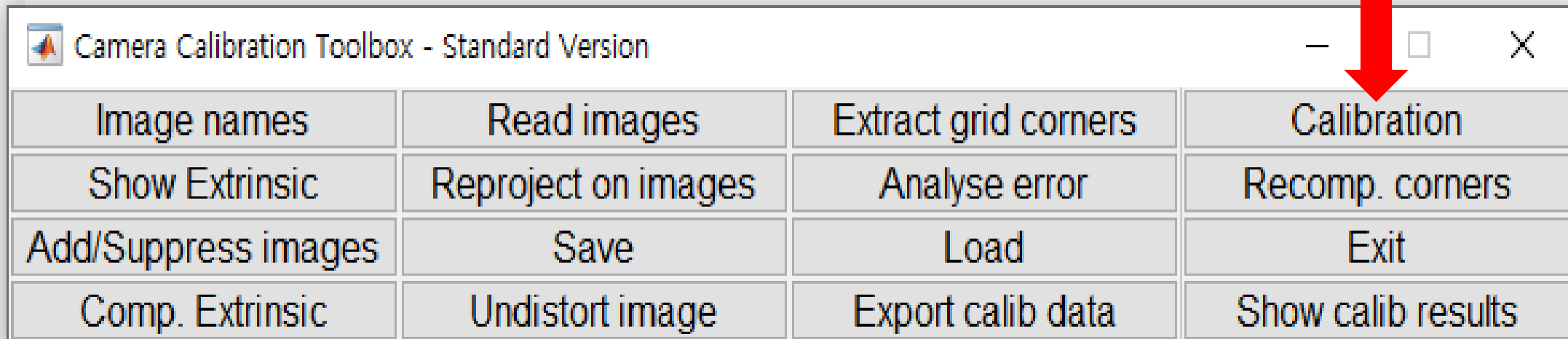
Estimated focal: 1378.5006 pixels

Guess for distortion factor  $k_c$  ([]=0): -.15

Satisfied with distortion? ([]=no, other=yes) 1|



## 2.3 Run Calibration





## 2.3 Run Calibration

Calibration results after optimization (with uncertainties):

Focal Length:  $fc = [ 962.45295 \quad 965.26304 ] \pm [ 1.65621 \quad 1.68497 ]$

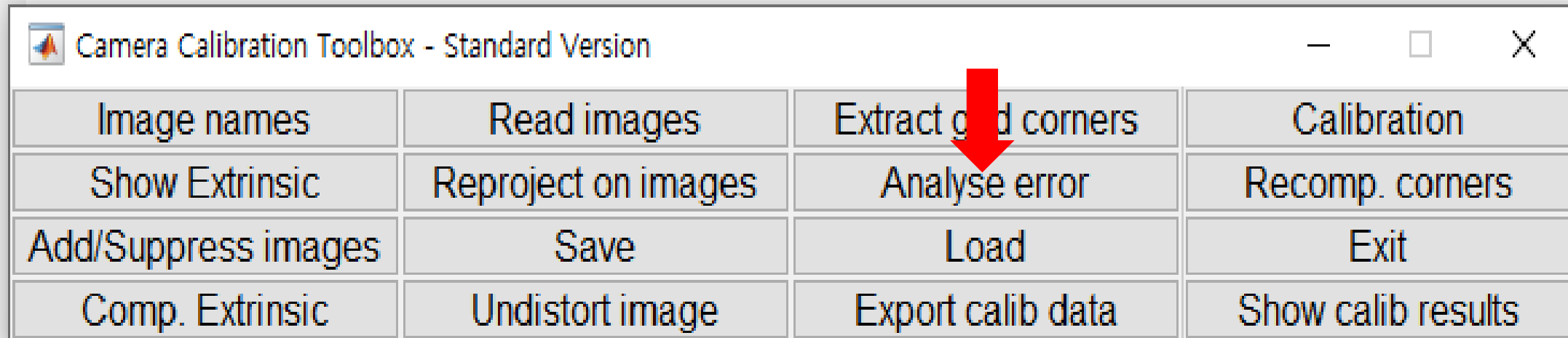
Principal point:  $cc = [ 627.81331 \quad 485.72672 ] \pm [ 2.32586 \quad 1.93282 ]$

Skew:  $\alpha_c = [ 0.00000 ] \pm [ 0.00000 ] \Rightarrow \text{angle of pixel axes} = 90.00000 \pm 0.00000 \text{ degrees}$

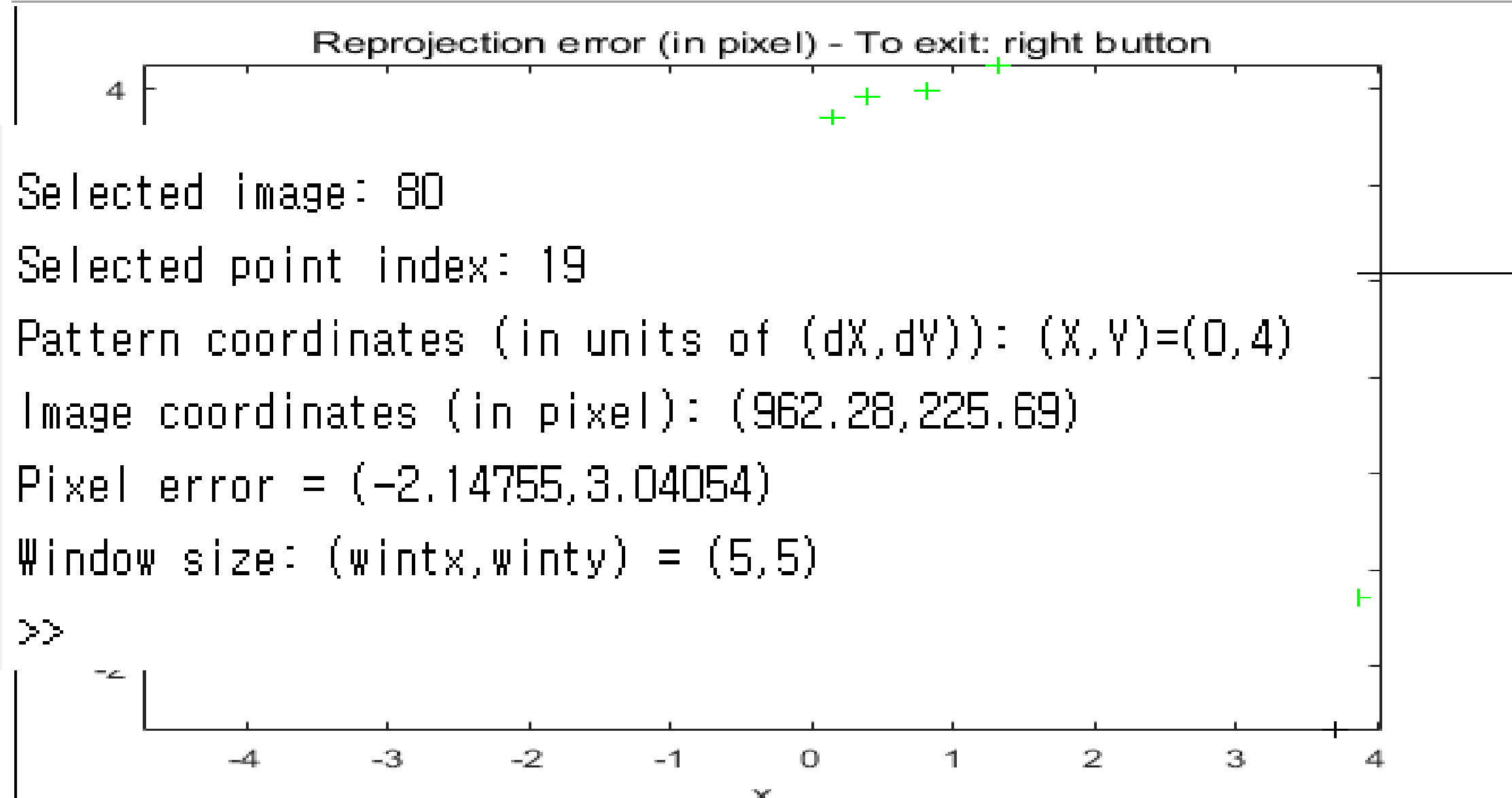
Distortion:  $kc = [ -0.23565 \quad 0.10159 \quad -0.00007 \quad -0.00010 \quad 0.00000 ] \pm [ 0.00295 \quad 0.00383 \quad 0.00039 \quad 0.00038 \quad 0.00000 ]$

Pixel error:  $err = [ 0.35945 \quad 0.26655 ]$

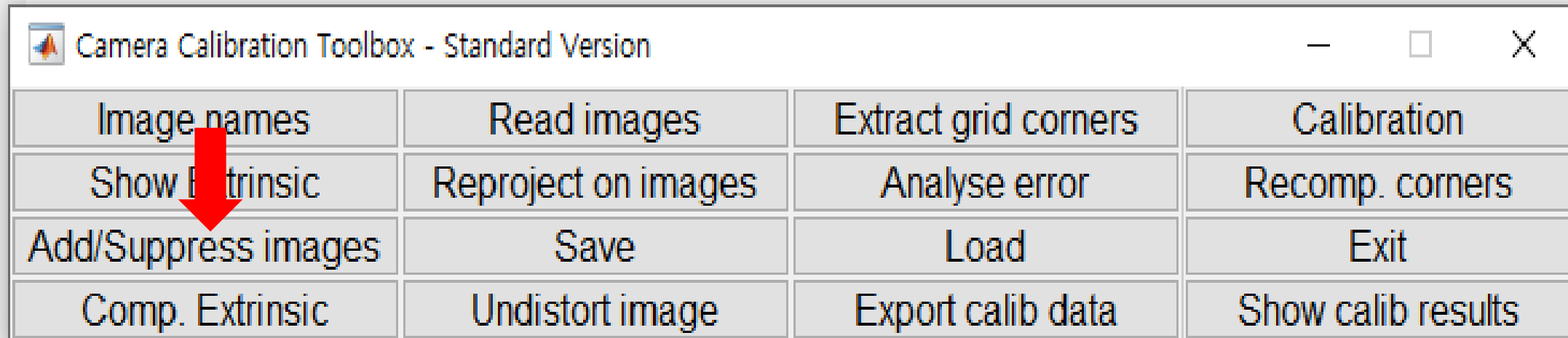
## 2.4 Analyse error



## 2.4 Analyse error




## 2.5 Suppress images



## 2.5 Suppress images

You probably want to suppress images

Number(s) of image(s) to suppress ([ ] = no image) = 80



Suppress 할 이미지 순서

There is now a total of 99 active images for calibration:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,

You may now run 'Calibration' to recalibrate based on this new set of images.





## 2. Camera Calibration ToolBox for Matlab

Main calibration optimization procedure - Number of images: 74

Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...done

Estimation of uncertainties...done

Calibration results after optimization (with uncertainties):

Focal Length:  $f_c = [ 959.56498 \quad 962.12196 ] \pm [ 0.85653 \quad 0.86536 ]$

Principal point:  $c_c = [ 661.46725 \quad 498.10308 ] \pm [ 1.13529 \quad 0.97713 ]$

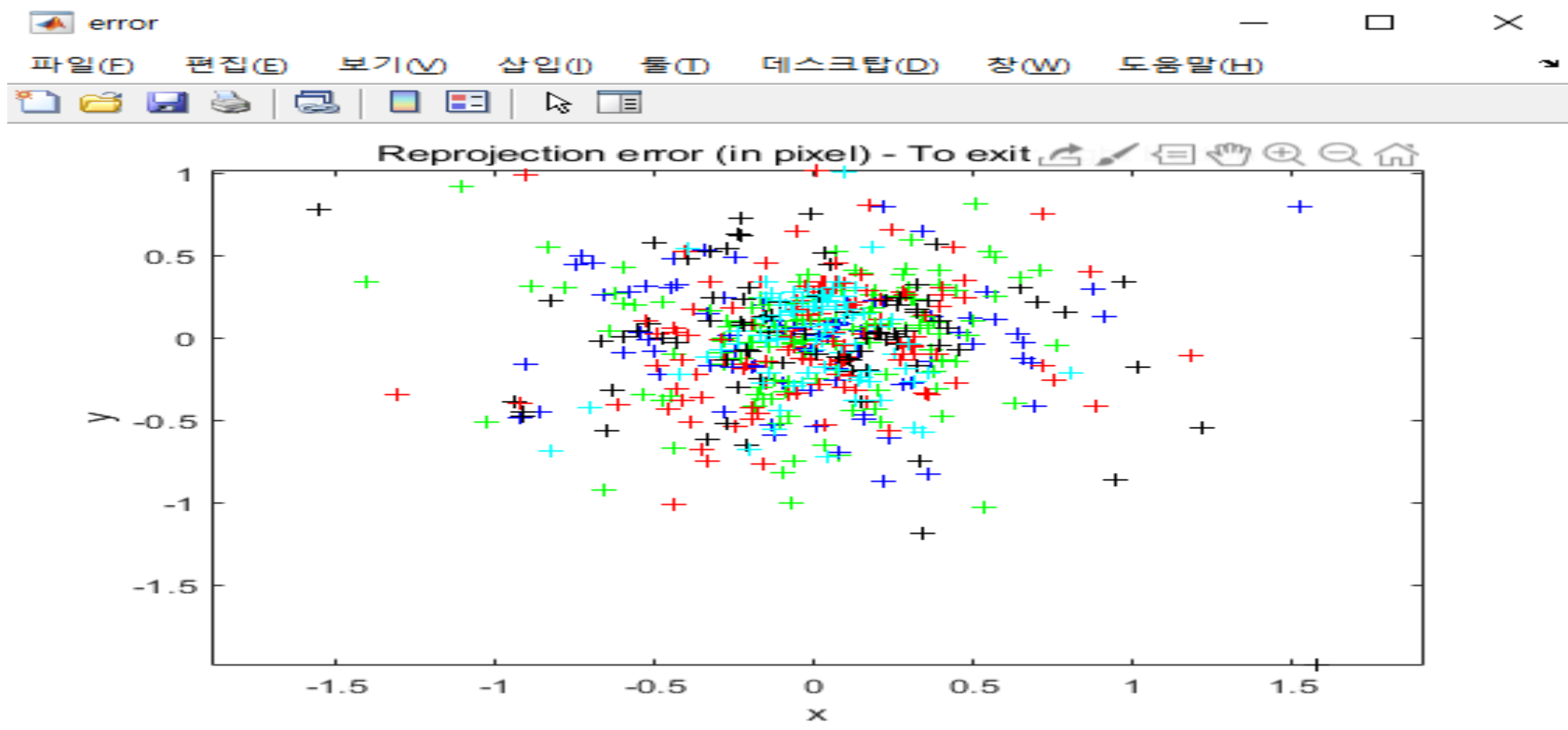
Skew:  $\alpha_c = [ 0.00000 ] \pm [ 0.00000 ] \Rightarrow \text{angle of pixel axes} = 90.00000 \pm 0.00000 \text{ degrees}$

Distortion:  $k_c = [ -0.23937 \quad 0.09825 \quad -0.00097 \quad 0.00054 \quad 0.00000 ] \pm [ 0.00162 \quad 0.00342 \quad 0.00023 \quad 0.0001$

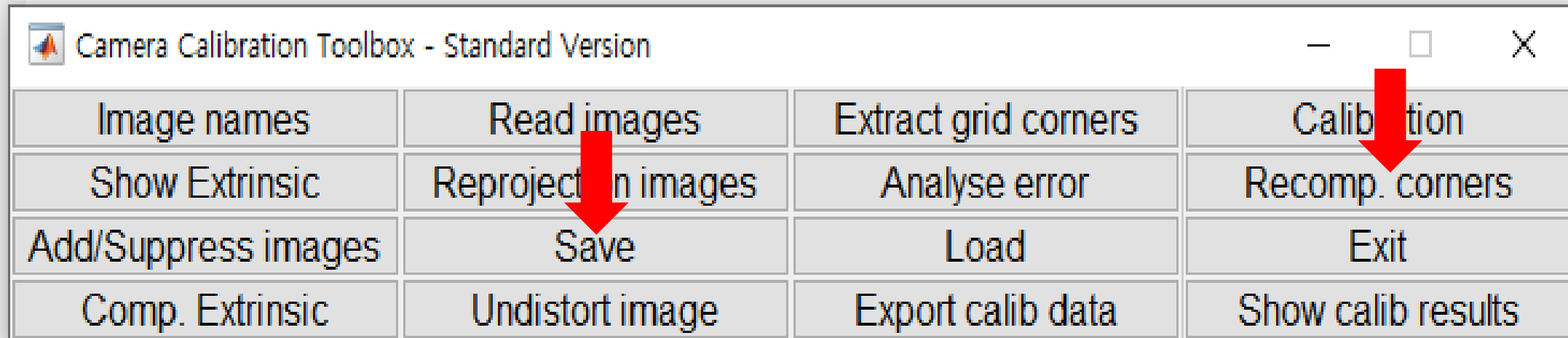
Pixel error:  $err = [ 0.12798 \quad 0.12293 ]$

Note: The numerical errors are approximately three times the standard deviations (for reference).

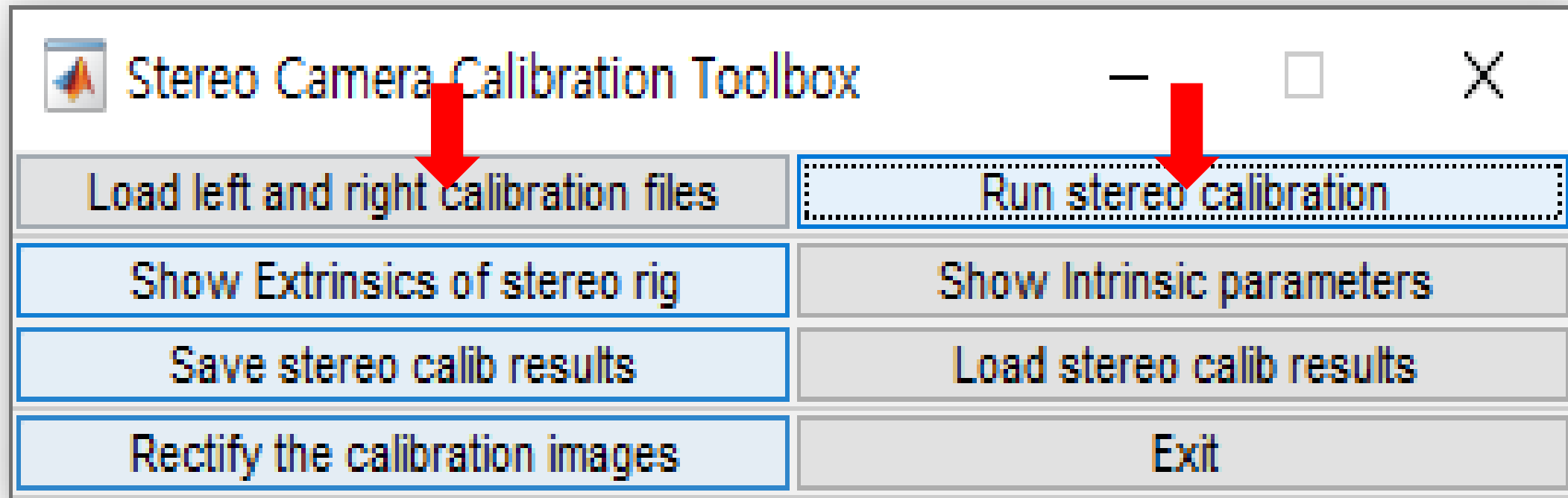
## 2. Camera Calibration ToolBox for Matlab



## 2.6 Recomp. corners



### 3. Stereo Calibration



### 3. Stereo Calibration

Intrinsic parameters of left camera:

```
Focal Length:      fc_left = [ 964.49636  968.22305 ] □ [ 2.70347  2.80450 ]
Principal point:   cc_left = [ 660.68724  502.39105 ] □ [ 4.41358  3.48049 ]
Skew:             alpha_c_left = [ 0.00000 ] □ [ 0.00000 ] => angle of pixel axes = 90.00000 □ 0.00000 degrees
Distortion:       kc_left = [ -0.24216  0.11135  0.00009  0.00063  0.00000 ] □ [ 0.00743  0.01889  0.00083  0.00000 ]
```

Intrinsic parameters of right camera:

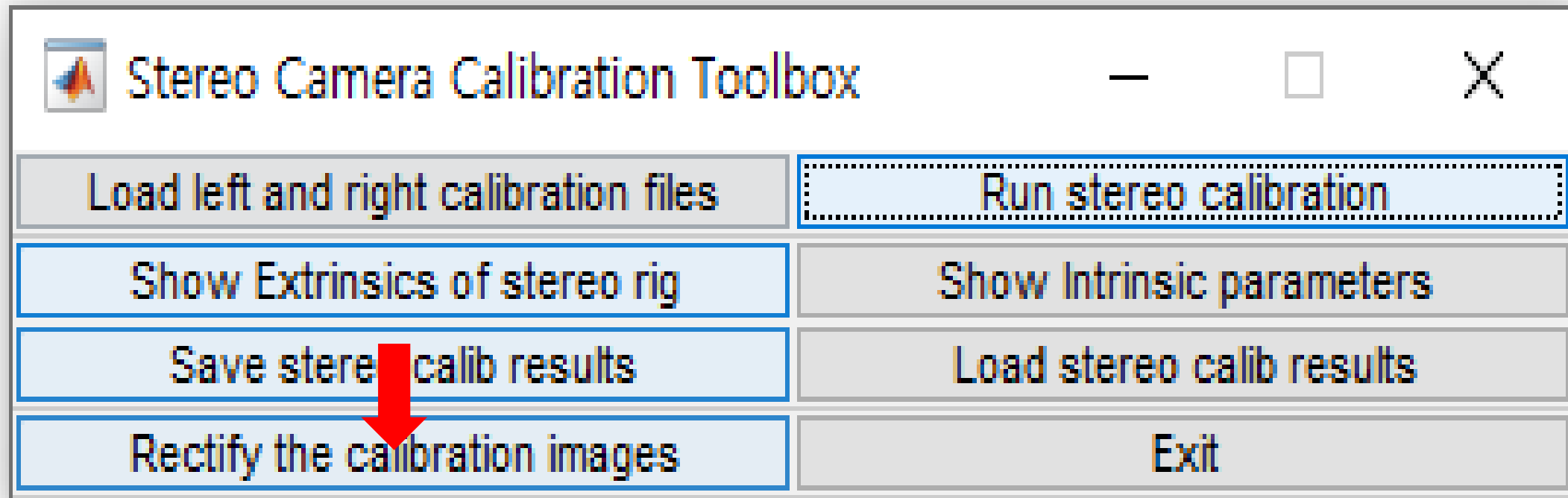
```
Focal Length:      fc_right = [ 961.73661  965.52583 ] □ [ 2.66230  2.68916 ]
Principal point:   cc_right = [ 630.31253  492.30981 ] □ [ 4.50743  3.53759 ]
Skew:             alpha_c_right = [ 0.00000 ] □ [ 0.00000 ] => angle of pixel axes = 90.00000 □ 0.00000 degrees
Distortion:       kc_right = [ -0.22854  0.08585  0.00045  -0.00031  0.00000 ] □ [ 0.00608  0.00800  0.00066  0.00000 ]
```

Extrinsic parameters (position of right camera wrt left camera):

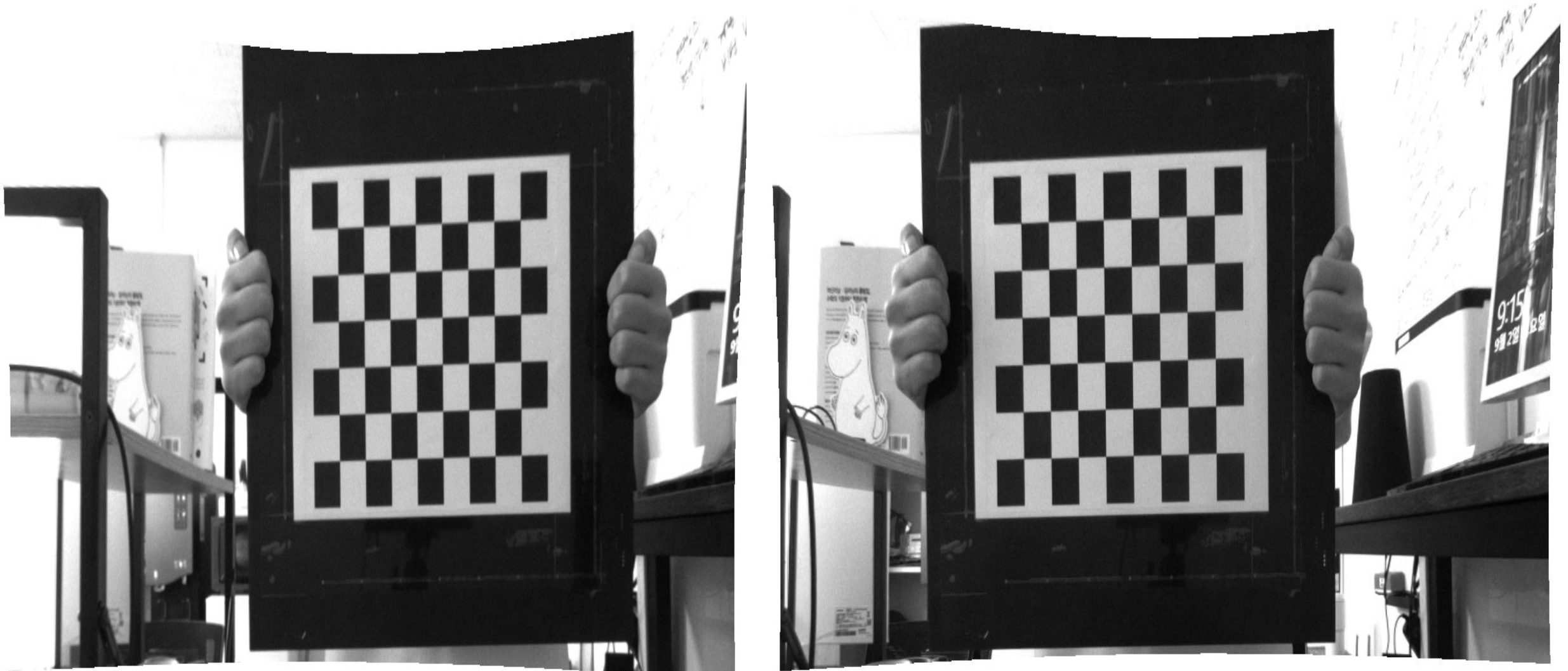
```
Rotation vector:   om = [ -0.00153  0.00031  -0.01135 ] □ [ 0.00373  0.00587  0.00071 ]
Translation vector: T = [ 130.75200  -0.97490  -0.49662 ] □ [ 0.27961  0.30682  1.26205 ]
```



### 3. Stereo Calibration



## 3.1 Rectify the calibration images



## 4. Disparity

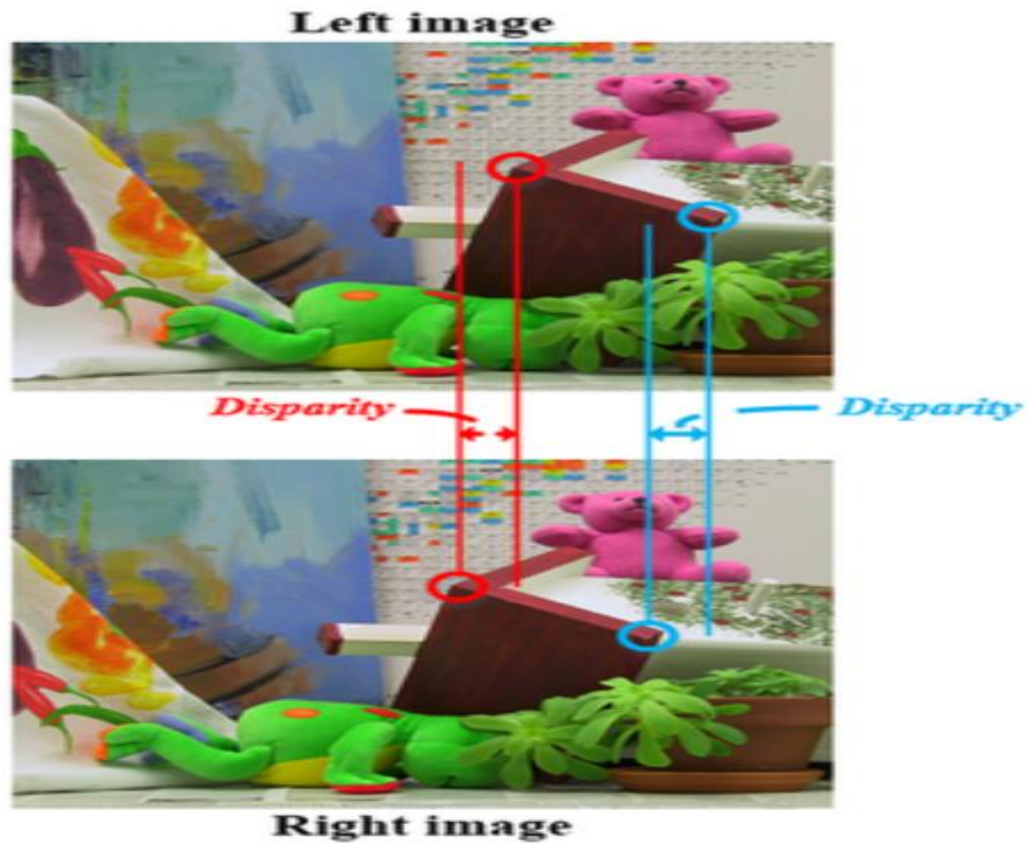


그림 2. disparity

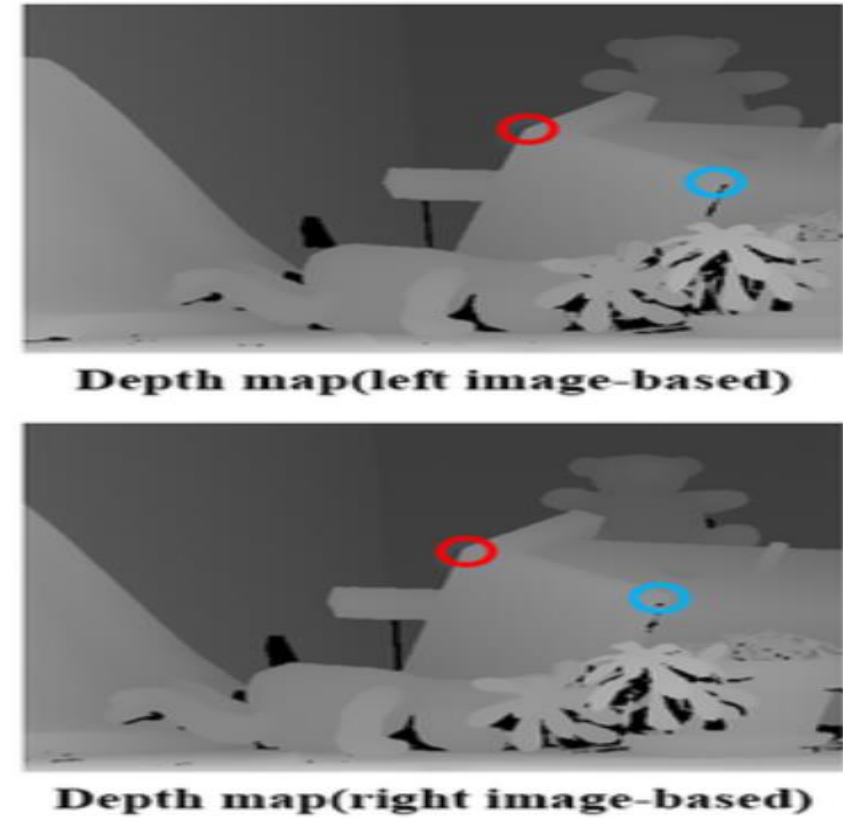
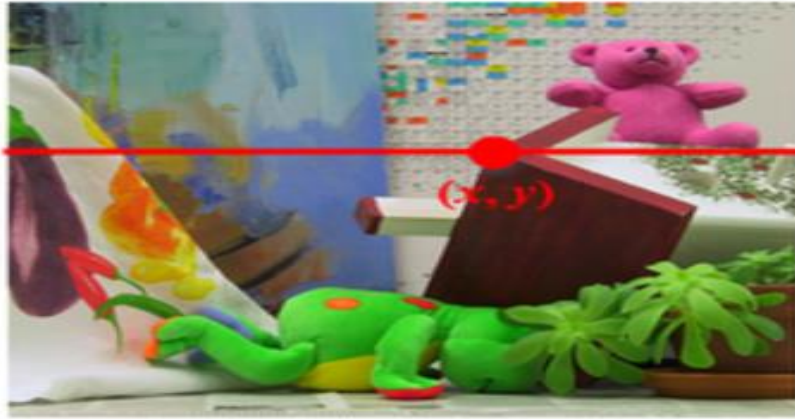
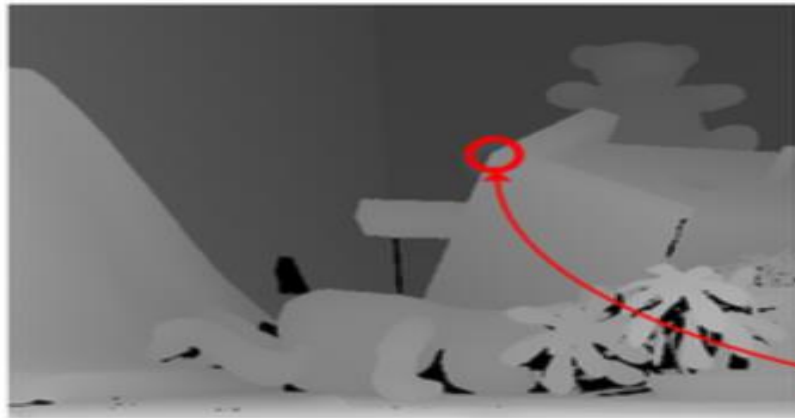


그림 3. Depth Map

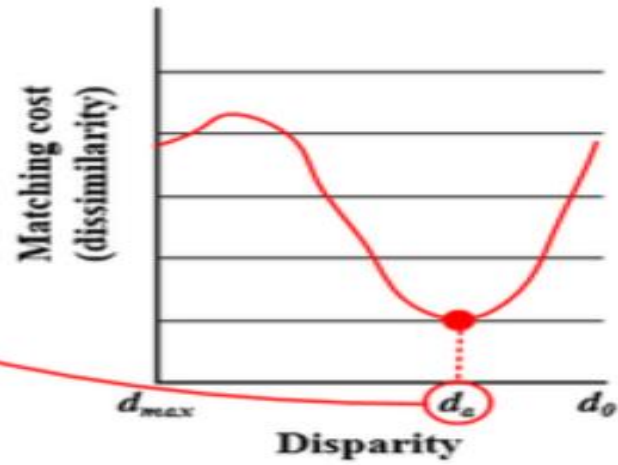
## 4. Disparity



Left image(reference image)

**Right image(target image)**

### Depth map



## 4.1 Disparity Map

Disparity Map과 Depth Map을 구해보자