s160066 & s160071
Available on Stroemskjold & chrwave Github

June 2021

Guide to Hand-Eye Calibration

This guide will help the user obtain the following:

- Camera calibration matrix (focal length and center point in pixel)
- Camera distortion coefficients
- Hand-Eye transformation matrix (rotation and translation of camera relative to end effector)

The guide is based on Python and requires a camera mounted on a robot arm. The guide uses the DTU UR equipment functions. The guide assumes that the UR and camera is connected to the PC.

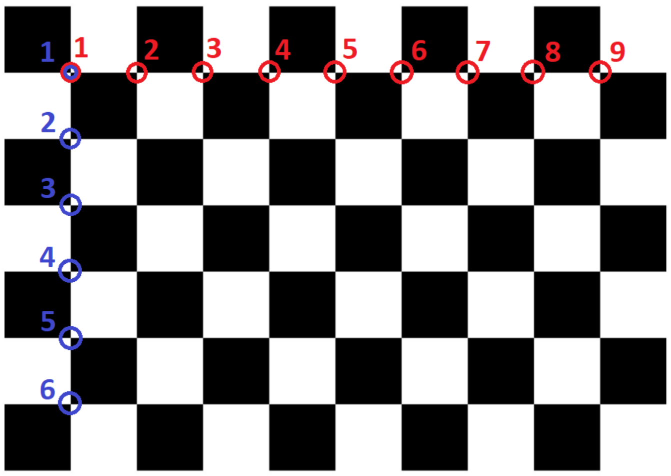
1. Include/install the following packages in the Python environment. The minimum required Python version is 3.8.8 for OpenCV compatibility:

```
pip             version==21.0.1
numpy           version==1.20.2
opencv-python   version==4.5.1.48
pyserial        version==3.5
datetime        version==any version
serial-tool     version==0.0.1
```

2. Download the folder from GitHub and save it in your working directory:

   https://github.com/chrwave/Hand_Eye_Calibration

3. Print the checkerboard from the file "chessboard_pattern.pdf" or other resource. Measure the width of the black squares and count the dimensions of the board as shown below:



The included chessboard has dimensions (9,6) and a width of 25 mm.

4. Edit the script “HandEyeCalibrationGuide.py” and change the variables *directory*, *ImageAmount*, *FactorPictureScaling*, *CheckerboardSquareSize* and *Checkerboard* as shown below. The directory can be found by typing “pwd” in the terminal.

```
###############################################
#Parameters that can be changed
directory = 'D:\Scripts\PythonScripts' #Should be changed to your directory
ImageAmount = 25                # Amount of images
FactorPictureScaling = 1        # Should be in the power of 2. (1, 2, 4, 8, 16)
CheckerboardSquareSize = 0.025 # Size of checkerboard square(s)
Checkerboard = (9,6)            # Checkerboard dimensions
###############################################
```

With the exception of *directory*, all parameters can be left unchanged if using the included chessboard. It is recommended to use 25 images, but the algorithm accepts values ≥ 3.

5. Change the camera device in the variable *cam*. If the PC only has one camera use value 0 otherwise use 1, 2, 3 etc. depending on number of potential inputs. The resolution can also be adjusted here.

```
#Camera initialization
###############################################
cam = cam.DTUCAMERA(1)
cam.SetCameraResolution(1920,1080)
cam.GetCameraResolution()
###############################################
```

6. Save the changes to the script and execute it. A new window will open with a live feedback of the camera. Position the robot such that the entire checkerboard can be seen with the camera. Take an image by pressing “Esc”. Move the robot to a new position and take another image. Continue taking pictures from different positions and rotations until you have images corresponding to the value in the variable *ImageAmount*. It is important to get as many unique poses as possible.
Examples of images can be found in the folder “ExamplePictures”.

7. The user will be prompted with the following message stating the results from the calibration algorithms; *Park-Martin*, *Tsai-Lenz*, *Horaud* and *Daniilidis*. The results should ideally converge. The transformation is from the gripper to camera frame. Rough manual measurements can be used as an estimate to verify the transformation.

```
Park-Martin Calibration Matrix (1):
[[-0.99982509  0.01720372  0.00733605 -0.00054608]
 [-0.0170585  -0.99966593  0.01941765  0.07349137]
 [ 0.00766765  0.01928911  0.99978455  0.06399736]
 [ 0.          0.          0.          1.        ]]

Tsai Calibration Matrix (2):
[[-0.57643875  0.81208626 -0.09074291  0.0253283 ]
 [-0.81712445 -0.57355515  0.057811    0.05352132]
 [-0.00509855  0.10747276  0.99419496  0.21196592]
 [ 0.          0.          0.          1.        ]]

Horaud Calibration Matrix (3):
[[-0.99982465  0.01721861  0.00736105 -0.00054582]
 [-0.01707226 -0.99966403  0.01950282  0.07348795]
 [ 0.00769439  0.01937373  0.9997827   0.06399611]
 [ 0.          0.          0.          1.        ]]
```

```
Daniilidis Calibration Matrix (4):
[[-0.9998424   0.01614164  0.0073912  -0.00049569]
 [-0.01599538 -0.99968312  0.01943731  0.07369336]
 [ 0.00770261  0.01931602  0.99978376  0.06321526]
 [ 0.          0.          0.          1.        ]]


Choose Calibration method (1-4): 1
```

The user must choose one of the methods by typing the corresponding number in the terminal. It is recommended to use *Park-Martin* since *Tsai-Lenz* has rotation problems in OpenCV and *Horaud* and *Daniilidis* requires more images than *Park-Martin*.

8. After typing the desired calibration method, all of the necessary parameters will be printed in the terminal:

```
Park-Martin Method
[[-0.99982509  0.01720372  0.00733605 -0.00054608]
 [-0.0170585  -0.99966593  0.01941765  0.07349137]
 [ 0.00766765  0.01928911  0.99978455  0.06399736]
 [ 0.          0.          0.          1.        ]]

Image Distortion Parameters:
[[ 0.06596595 -0.31947411  0.00207942 -0.00057211  0.484688  ]]

Camera Intrinsic Matrix:
[[1442.51138475    0.          963.98685516]
 [   0.         1442.26829892  535.90506659]
 [   0.            0.            1.        ]]

Optimal Camera Intrinsic Matrix
[[1453.58532715    0.          962.25511949]
 [   0.         1446.32336426  536.97794233]
 [   0.            0.            1.        ]]
```