Practical Work P1

Camera calibration using OpenCV & Python

This practical work is designed to illustrate the calibration of a digital camera using OpenCV.

Preparation

The first dependency that is needed is OpenCV library. The OpenCV library can be installed with the command:

```
pip install opencv-python
```

or for conda installation:

```
conda install -c conda-forge opencv-python
```

The second dependency is related to image and geometry display. The MatPlotLib library that can be installed with the command:

```
pip install matplotlib
```

or for conda installation:

```
conda install -c conda-forge matplotlib
```

The MatPlot library displays geometric rendering in an independent interactive window. Depending on the Python environment, this window may be rendered as a frozen image, preventing user interaction. To correct this problem, here are a few solutions:

Jupyter Notebook:

Execute following code within the Jupyter Notebook:

```
%matplotlib qt
```

PyCharm

Go to `Settings / Tool / Python Plot` and uncheck the option `Show plots in tool windows`.

Spyder

Go to `Tools / Preferences / IPython console / Graphics / Backend:Inline` and change "`Inline`" to "`Automatic`". Click OK button and restart the IDE.

Julien SEINTURIER
http://www.seinturier.fr / julien.seinturier@univ-tln.fr

Using OpenCV and MatPlot within Python program

OpenCV python binding relies on Numpy for the vector and matrix representation and on MatPlotLib for display. Using OpenCV within a Python program needs to import the three libraries:

```
import numpy as np

import cv2 as cv

import matplotlib.pyplot as plt
```

All Python program that uses OpenCV has to contain these imports.

Exercise 1

Create a python file `calibration_chessboard.py` that contains the following program:

```python
import numpy as np
import cv2 as cv
import glob
import matplotlib.pyplot as plt


def calibrate_chessboard(image_files):
    print('Chessboard calibration')


def main():
    calibrate_chessboard(())


if __name__ == "__main__":
    main()
```

Run the program to ensure that all the dependencies are installed.

Camera calibration

The camera calibration relies on taking images of a rigid frame (the chessboard) and processing it.

Preparation

The first steps before calibrating a camera are to ensure that all the needed dependencies are set up and that it is possible to load images from files.

Exercise 2:

Create a folder named **`data`** in the same directory as **`calibration_chessboard.py`** and put some images within the data folder. Modify then the **`calibration_chessboard.py`** program as follows:

```
import numpy as np
import cv2 as cv
import glob
import matplotlib.pyplot as plt


def calibrate_chessboard(image_files):

    print('Image list:')
    for image_file in image_files:
        print('   -'+image_file)


def main():

    # Load images from data folder
    image_files = glob.glob('data/images/*.jpg')

    calibrate_chessboard(image_files)


if __name__ == "__main__":
    main()
```

Run the program and ensure that are the images within **`data`** directory are listed.

Before calibrating, listed images has to be loaded within OpenCV format. The load of an image from a file is possible with the instructions:

```
img = cv.imread(image_file)
```

```
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
```

The first one load an image within OpenCV format, the second one convert the images to grayscale (that is needed by calibration algorithm).

When an image is loaded, it is possible to display it using MatPlotLib.

Exercise 3:

Modify the **calibration_chessboard.py** program as follows:

```
import numpy as np
import cv2 as cv
import glob
import matplotlib.pyplot as plt


def calibrate_chessboard(image_files):

    for image_file in image_files:

        # Load image
        img = cv.imread(image_file)

        # Convert image to gray
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

        # Display image
        plt.imshow(img, cmap='gray')
        plt.show()


def main():

    # Load images from data folder
    image_files = glob.glob('data/images/*.jpg')

    calibrate_chessboard(image_files)


if __name__ == "__main__":
    main()
```

Ensure that when running the program, images are showing within MatPlotLib window.

Chessboard configuration

Camera calibration process aims at optimizing the chessboard seen on images with its theoretical description. Program has to know the characteristics of the chessboard.

Exercise 4:

Modify the program **calibration_chessboard.py** in order to integrate chessboard characteristics. The function calibrate_chessboard(image_files) has to be added with:

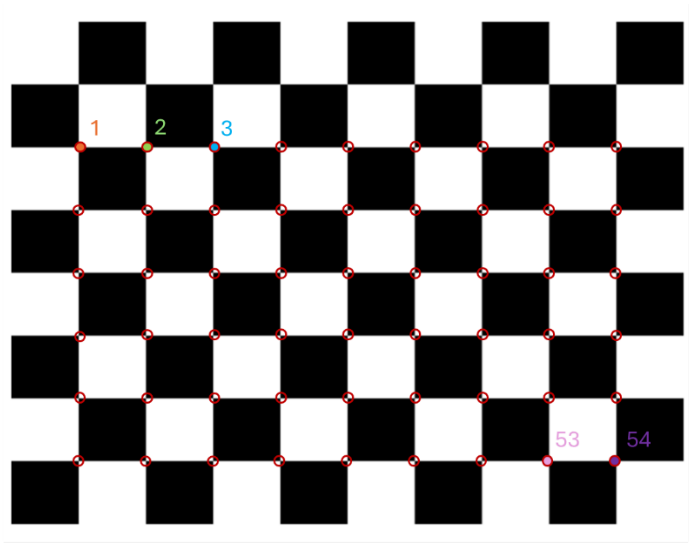
```
def calibrate_chessboard(image_files):
    # Chessboard dimension
    rows = 9
    cols = 6
    size = 1.0

    objp = np.zeros((rows * cols, 3), np.float32)

    for x in range(0, cols):
        for y in range(0, rows):
            objp[y * cols + x] = [x * size, y * size, 0]
```

Chessboard detection

Performing calibration relies on finding the chessboard corners on all images. OpenCV provide a function that enable to detect corners of a chessboard:

```
ret, corners = cv.findChessboardCorners()
```

This function returns the list of detected corners from upper right to bottom left, line by line.



```
corners = [[x, y], [x, y], [x, y], …, [x, y], [x, y]]
             1       2       3          53      54
```

Exercise 5:

Modify the program **`calibration_chessboard.py`** in order to integrate chessboard detection on images. The function calibrate_chessboard(image_files) has to be added with:

```
# Find the chess board corners
ret, corners = cv.findChessboardCorners(gray, (cols, rows), None)

# If found, add object points, image points (after refining them)
if ret is True:

    # Display image with corners
    plt.imshow(img, cmap='gray')

    for corner in corners:
        plt.plot(corner[0][0], corner[0][1], marker='+', color='red')

    plt.show()

else:
    print('No chessboard corner found on image ' + image_file)
    plt.imshow(img, cmap='gray')

    plt.show()
    print('No chessboard corner found on image ' + image_file)
    plt.imshow(img, cmap='gray')

plt.show()
```

Data preparation

Chessboard detection can be optimized by enabling sub-pixel precision. The refinement of detected corners can be done using the OpenCV function:

`cv.cornerSubPix()`

Thin function requires in entre the result from corner detection and a an optimization criteria.

Exercise 6:

Modify the program **`calibration_chessboard.py`** in order to integrate chessboard detection refinement. Within The function calibrate_chessboard(image_files), the following code:

```
# If found, add object points, image points (after refining them)
if ret is True:

    # Display image with corners
    plt.imshow(img, cmap='gray')

    for corner in corners:
        plt.plot(corner[0][0], corner[0][1], marker='+', color='red')

    plt.show()

else:
    print('No chessboard corner found on image ' + image_file)
    plt.imshow(img, cmap='gray')

plt.show()
```

has to me modified with:

```
# If found, add object points, image points (after refining them)
if ret is True:

    # termination criteria
    criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30,
0.001)

    corners2 = cv.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)

    # Display image with corners
    plt.imshow(img, cmap='gray')

    for corner in corners2:
        plt.plot(corner[0][0], corner[0][1], marker='+', color='red')

    plt.show()

else:
    print('No chessboard corner found on image ' + image_file)
    plt.imshow(img, cmap='gray')

plt.show()
```

Ensure that the program still detect correctly the chessboard

Calibration computation

Camera calibration work with determining the parameters to pass from corners position on an image to a corner position in the 3D worlds. For this purpose, it is needed to express the correspondences between 2D and 3D points. For that, two arrays have to be created, containing for a same index I, the 3D coordinates in one array and the 2D coordinates of the corner in the second array.

Exercise 7:

Modify the program **`calibration_chessboard.py`** in order to integrate chessboard detection refinement. Within The function calibrate_chessboard(image_files), two arrays have to be declared at the beginning of the function:

```
# Arrays to store object points and image points from all the images.
points_2d = []  # 2d points in image plane
points_3d = []  # 3d point in real world space
```

These arrays have to be updated after the corner detection of an image. For that, the code:

```
    corners2 = cv.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)

    # Display image with corners
    plt.imshow(img, cmap='gray')

    for corner in corners2:
        plt.plot(corner[0][0], corner[0][1], marker='+', color='red')

    plt.show()
```

has to be modified into:

```
    corners2 = cv.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)

    # Update 2D / 3D corner coordinates
    points_3d.append(objp)
    points_2d.append(corners2)

    # Display image with corners
    plt.imshow(img, cmap='gray')

    for corner in corners2:
        plt.plot(corner[0][0], corner[0][1], marker='+', color='red')

    plt.show()
```

The last information needed to perform the calibration is the size of the input images. This size can be obtained using OpenCV integrated functions.

Exercise 8:

Modify the program **calibration_chessboard.py** in order to perform image size retrieval. Within The function calibrate_chessboard(image_files), two arrays have to be declared at the beginning of the function:

```
image_width = -1
image_height = -1
```

For each loaded image, its size can be obtained by modifying the code:

```
for image_file in image_files:

    # Load image
    img = cv.imread(image_file)
```

with:

```
for image_file in image_files:

    # Load image
    img = cv.imread(image_file)

    # Get the image dimension
    (h, w) = img.shape[:2]

    if image_width == -1 and image_height == -1:
        image_width = w
        image_height = h
    elif w != image_width or h != image_height:
        print('All calibration images have to got same dimension, ignoring
' + image_file)
        continue
```

With all these information, camera calibration can be performed.

Exercise 9:

Modify the program **calibration_chessboard.py** by appending to the function:

```
print('Calibrating camera...', end='')
camera_matrix = np.zeros((3, 3, 1), np.float32)
dist_coefs = np.zeros((5, 1, 1), np.float32)

retval, camera_matrix, dist_coefs, rvecs, tvecs =
cv.calibrateCamera(points_3d, points_2d, (image_height, image_width),
camera_matrix, dist_coefs)
```

Execute the whole program and try to calibrate your own camera.

Julien SEINTURIER
http://www.seinturier.fr / julien.seinturier@univ-tln.fr