School of Computing
National University of Singapore
CS4243 Computer Vision and Pattern Recognition
Semester 1, AY 2013/14

Lab 2: Feature Detection

Objectives:

- Continue to learn to write simple Python programs.
- Learn to use OpenCV for feature detection.

Preparation:

- Create a folder in the PC with your name, e.g., `d:/myname`. This folder will be used as your working directory.
- Download the files `featureAY1415.doc` & `scene.jpg` into your working directory.

Part 1. Starting Python

This part illustrates how to start using Python for simple image processing with OpenCV Python.

1. Write codes in a program file.
   Although the interpretative mode of Python is very convenient for doing quick checks, it is often more convenient to write the codes in a program file.
   Create a python program file called feature.py and follow the steps in the rest of this lab instruction to add commands to feature.py.
   To run a python program, do the following

   ```
   >>> import sys
   >>> sys.path.append(".")
   >>> execfile("feature.py")
   ```

   The first two steps are optional. They are required only if the current directory is not searched for feature.py.

2. Set the working directories, e.g., `d:/myname`.

   ```
   import os
   os.chdir("d:/myname")
   ```

3. Read and display image information with OpenCV.
    1. Initialization
    ```
    import cv2
    import cv2.cv as cv
    import numpy as np
    import sys
    ```

    2. Read image.
    ```
    image = cv2.imread("scene.jpg", cv2.CV_LOAD_IMAGE_COLOR)
    ```

    3. Display image information.
    ```
    print image
    ```

4. Apply Gaussian smoothing to remove noise.
    ```
    for i in xrange(1,31,2):
        gaussianImg = cv2.GaussianBlur(image,(i,i),0)
        txt = 'Guassian Filtered Image : kernel size = ' + str(i)
        cv2.putText(gaussianImg, txt, (20,20),
        cv2.FONT_HERSHEY_COMPLEX_SMALL,1,(0,255,255))
        cv2.imshow('GaussianSmoothing',gaussianImg)
        cv2.waitKey(500)
    ```

5. Save image to a file.
    ```
    cv2.imwrite('smooth.jpg', gaussianImg)
    ```

    Open `smooth.jpg` to check that it is indeed a smooth version of `scene.jpg`.

Part 2. Harris Corner Detection

The following code segment illustrates how to apply Harris corner detector to detect good features in an image.
It works in the following steps:

1. Load input image as a grayscale image. This is required by Harris corner detector.
2. Apply Harris corner detector to the image.
3. Reload input image as colour image for painting corner points.
4. Mark the locations with large corner values.
5. Save corner image.

```
# code for Harris corner detection
image = cv2.imread("scene.jpg", cv2.CV_LOAD_IMAGE_GRAYSCALE)

# Apply Harris corner detector
harris = cv2.cornerHarris(image, 2, 3, 0.04)

# Reload image as color image for painting corner points
image = cv2.imread("scene.jpg", cv2.CV_LOAD_IMAGE_COLOR)

width = image.shape[1]
height= image.shape[0]

# Mark locations with large corner values
threshold = 0.001
for x in range(width):
   for y in range(height):
         if harris[y, x] > threshold:
            center = (x,y)
            radius = 5
            thickness = -1  # negative means filled
            lineType  = 8
            shift     = 0
            cv2.circle(image, center, radius, (0, 0, 255, 0), thickness, lineType, shift)

# Save image
cv2.imwrite('harris.jpg', image)
```

There are several parameters that you can adjust to change the corner detection results:

- block size: size of the block in image used to compute corner value.
- threshold: the minimum corner value for a location to be considered as a good corner

Experiment with different parameter values and observe their effects on the corner detection results.

Part 3. Simple Programming Assignment (to be submitted for grading)

Write a Python program to detect Tomasi's good features as follows:

1. Load input image as a grayscale image.
2. Apply cv2.goodFeaturesToTrack with appropriate parameter values:
    - Set maximum number of corners to detect to a sufficiently large number, e.g., 200.

- Quality level should not be too small; otherwise, many bad corners will be detected. It should not be too large also; otherwise, many good corners are rejected. Try 0.001 for a start.
- Set minimum distance to an appropriate value, e.g., 11.0, to space out the detected corners.

3. cv2.goodFeaturesToTrack returns an array of detected corner positions, i.e., [(x1, y1), (x2, y2), …].
4. Reload the input image as a color image.
5. Mark the locations of the detected corners in the image. Suppose the array that cv2.goodFeaturesToTrack returns is called `feature_list`. Then, the locations in `feature_list` can be accessed as follows:

```
feature_list = feature_list.reshape((-1,2))
for (x, y) in feature_list:
    ...
```

   The color value is given as (blue, green, red, alpha). So, red color is specified as (0, 0, 255, 0).
6. Save the image marked with corners.

To edit your program, you can use the editor that comes with IDLE, the Python GUI. Alternatively, you can use NotePad++, Vim or NotePad in Windows, and and Vim, GNU Emacs or any other editors in Linux.

Submit the following at the end of the lab session:

1. Print-out of your Python program.
2. Print-out of the image marked with corners.
3. Submit the softcopy of your Python program to IVLE
   Please put your python program in a folder and submit the folder. Use the following convention to name your folder:
   MatriculationNumber_yourName_Lab#. For example, if your matriculation number is A1234567B, and your name is Chow Yuen Fatt, for this lab, your file name should be A1234567B_ChowYuenFatt_Lab2.

Remember to write your name, matriculation number, and lab group number on the hardcopy print-outs.