Interest Point Detectors and decriptors

1 Introduction

The objective of this session is to practice with interest points detectors and descriptors, following the lecture 3 of Computer Vision course. You can get more information about lectures and associated practical sessions on my website: `http://chateaut.fr`. Essential needed knowledges includes a beginning level of computer science (Linux, Python programming with `Opencv`, `Numpy`, `Matplotlib`, and `Pytorch` (for sessions on Deep Learning)) and applied mathematics.

The listing 1 shows an "Hello Word" python script that loads and convert an image using `opencv`; and displays it with `matplotlib` and illustrates relevant functions.

2 Interest Points detector: Harris

The algorithm of Harris and Stefen corner point detector is:

- Vertical and horizontal gradient computation (using a Sobel Filter): `Ix` et `Iy`
- Computation of the square gradients: $\texttt{Ix}^2$, $\texttt{Iy}^2$ and of the cross-gradient `IxIy`
- windows spatial convolution using a 3×3 gaussian or average kernel:
  Average kernel
  $$\texttt{W} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$
  Gaussian kernel
  $$W(i,j) = C \exp\left[-\frac{i^2+j^2}{2.\sigma^2}\right]$$
- Computation of Harris score R map from the Harris Matrix `M`:
  $$\texttt{M} = \texttt{W} * \begin{pmatrix} (\texttt{Ix})^2 & \texttt{Ixy} \\ \texttt{Ixy} & (\texttt{Iy})^2 \end{pmatrix} = \begin{pmatrix} \texttt{W} * (\texttt{Ix})^2 & \texttt{W} * \texttt{Ixy} \\ \texttt{W} * \texttt{Ixy} & \texttt{W} * (\texttt{Iy})^2 \end{pmatrix}$$
- $\text{Trace}(M) = \lambda_1 + \lambda_2 = M_{1,1} + M_{2,2}$
- $\text{Det}(M) = \lambda_1.\lambda_2 = M_{1,1}M_{2,2} - M_{1,2}M_{2,1}$
- $$R = \text{Det}(\texttt{M}) - k(\text{Trace}(\texttt{M}))^2$$
  k is a setting parameter (usually $k = 0.04$)
- Harris score R threshold.

- Local Maximun Extraction (A local maximum is defined by a point that has only neighbours with lower values)
- Coordinate extraction of the resulting points

The linear spatial filtering of an image `I` is computed by:

$$\mathtt{I_f}(x,y) = \sum_{i=-m}^{m} \sum_{j=-n}^{n} W(i,j).\mathtt{I}(x+i, y+j) \tag{1}$$

Study to realize

You should work from the script `testharris.py`.

1. Complete the Python script `testharris.py` to compute the Harris score map and display a pseudo-image of this map. Test should be done on images `test1.jpg` and `test2.jpg`.
2. Modify the script to extract interest points from Harris score and superimpose them to the original image. Extracting interest points from Harris map is achieved by a two steps algorithm: 1) threshold on the score map (n% from the max. value) and 2) extract local maximum (function `nms` is given to help you but you should explain how it works?).
3. Invariance to luminance offset: Compare the output of the algorithm for images `toy1` and `toy_lum` (that was generated with a luminance offset of `toy1`). For a fair comparison, threshold on harris score should be the same (set it to 300000 for example). Conclude.
4. Invariance to affine transformation of luminance: Compare the output of the algorithm for images `toy1` and `toy_aff` (that was generated by an affine luminance transformation of `toy1`). For a fair comparison, threshold on harris score should be the same (set it to 300000 for example). Conclude.
5. Invariance to image rotation: Compare the output of the algorithm for images `toy1` and `toy_rot` (that was generated by a rotation of `toy1`). Conclude
6. Invariance to image scaling: Compare the output of the algorithm for images `toy1` and `toy_sc` (that was generated by an image scaling of `toy1`). For a fair comparison, thresold on harris score should be the same (set it to 300000 for example). Conclude.
7. Opencv proposes an implementation of Harris Detector in the function:

```
Pts = cv2.goodFeaturesToTrack(imgref,maxCorners = 300,
      qualityLevel = 0.01, minDistance = 4,
      useHarrisDetector=True, k=0.04)}
```

3 Descriptors and Matching using Opencv

In this section, hand-crafted descriptors (hand-crafted means that it has been designed without machine learning) are illustrated in an object detection application. The algorithm takes a reference and a query image as input, computes interest points and descriptors on the two images and matches its in order to detect roughly where is the object in the query image (by computing the mean of the positive matches). The core of the algorithm is given is the script `test_harris_recog.py`.

1. Analyse the code and find the functions corresponding to: the definition of the detector and descriptor, the detection and description step, the matching step.
2. Analyse for the SIFT, SURF, KAZE and ORB descriptors, the performances of the feature matching step (invariance from rotation, scale and illumination variation)
3. modify the script to display, in the query image, roughly, the position of the reference object

Listing 1: HelloWord Python script: load an image and convert it using opencv then display using matplotlib

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Aug  6 08:46:29 2018
hello_wold simply loads an image with opencv, converts it to RGB and GRAYSCALE
and displays it using matplotlib
@author: thierrychateau
"""
import cv2
from matplotlib import pyplot as plt
import numpy as np
# This is an example of function with a string argument that returns another
# string
def my_func(my_arg):
    string2 = my_arg + ' toto ';
    return string2
# This is the main function that is launched when this file is run as a script
def main():
    # call function my_func
    st = my_func("my name is")
    print(st)
    # Read an image file (im is a numpy matrix
    # (nblines,nbcolums, BGR 3 channels))
    filename = "billard_large.jpg"
    img = cv2.imread(filename)
    # Convert to RGB
    imgrgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    # Convert to grayscale image
```

```
    imggray=cv2.cvtColor( imgrgb, cv2.COLOR_RGB2GRAY )
    ## Display the RGB and gray images
    # divide plot screen into 1 line, 2 columns and select first subscreen as
    # current screen
    plt.subplot(121)
    plt.imshow(imgrgb)
    plt.title('Color RGB image')
    plt.subplot(122)
    plt.imshow(imggray, 'gray',vmin=0, vmax=255)
    plt.title('Grayscale image')
#
if __name__ == "__main__":
    # execute only if run as a script
    main()
```