School of Computing
National University of Singapore
CS4243 Computer Vision and Pattern Recognition
Semester 1, AY 2014/15

Lab 3: Motion Tracking

Objectives:

- Learn to use OpenCV for motion tracking.

Preparation:

- Read the following materials to familiarize yourself **before** you go for the lab session:
    - OpenCV Python Reference Manual:
        - Basic Structures
        - Reading and Writing Images and Video
        - Functions cv2.imread, cv2.imwrite, cv2.goodFeaturesToTrack, cv2.cornerSubPix, cv2.calcOpticalFlowPyrLK.
- Create a folder in the PC with your name, e.g., `d:/myname`. This folder will be used as your working directory.
- Download the file Lab_Motion.zip from IVLE into your working directory. Uncompress the file and you should find the following files: `motionAY1415.pdf`, `LabPhoto1.jpg`, `LabPhoto2.jpg` & `LabVideo.MOV`.

Part 0. Import Modules

1. `Create a python script file and name it motion.py.`
2. `Set` the working directory, e.g., `d:/myname`, and import relevant modules:

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

Part 1. Image and Video Input, Output and Display

Part 1 illustrates useful OpenCV Python functions for reading, writing, and displaying images and video.

1. Read an image and display image information.
   Read and display image information with OpenCV Python:

```
import cv2
import cv2.cv as cv
import numpy as np
import os
os.chdir("d:/myname")

im = cv2.imread("LabPhoto1.jpg", cv2.CV_LOAD_IMAGE_COLOR)
gr = cv2.imread("LabPhoto1.jpg", cv2.CV_LOAD_IMAGE_GRAYSCALE)

# Pops up a window for displaying the image
winname = "imageWin"
win = cv.NamedWindow(winname, cv.CV_WINDOW_AUTOSIZE)
string = 'motion'
cv2.putText(im,string,(20,20),cv2.FONT_HERSHEY_COMPLEX_SMALL,1
,(255,255,255))
cv2.imshow('motion image',im)
cv2.waitKey(1000)
cv.DestroyWindow(winname)
```

Note that reading the image with `cv2.CV_LOAD_IMAGE_GRAYSCALE` forces the loaded image to become a grayscale image.
A grayscale image has only one channel.

2. Saving images

```
cv2.imwrite('colorImage.jpg', im)
cv2.imwrite('grayImage.jpg', gr)
```

3. Read video file and display video information.

```
cap = cv2.VideoCapture("/ --- your directory ---/LabVideo.MOV")
print "frameWidth  = ",  cap.get(cv.CV_CAP_PROP_FRAME_WIDTH)
print "frameHeight = ",  cap.get(cv.CV_CAP_PROP_FRAME_HEIGHT)
print "fps         = ",  cap.get(cv.CV_CAP_PROP_FPS)
print "frameCount  = ",  cap.get(cv.CV_CAP_PROP_FRAME_COUNT)
```

4. Read a video and play back a fast version:

```
# Play video in window
invid = cv2.VideoCapture("/---your directory ---/LabVideo.MOV")

width  = int(invid.get(cv.CV_CAP_PROP_FRAME_WIDTH))
height = int(invid.get(cv.CV_CAP_PROP_FRAME_HEIGHT))
fps    = int(invid.get(cv.CV_CAP_PROP_FPS))
len    = int(invid.get(cv.CV_CAP_PROP_FRAME_COUNT))
```

```
# process the video frame by frame
for i in range(len):
   _,im = invid.read()

   if (i % 3 == 0):
      cv2.imshow('fastForward',im)
      cv2.waitKey(100)

# Close video and window.
del invid
cv2.destroyAllWindows()
```

Part 2. Motion Tracking

Part 2 is to use OpenCV Python functions for motion tracking by Lucas-Kanade tracker.

1. Read two images in colour LabPhoto1.jpg and LabPhoto2.jpg and show them in two separate windows

   Example: im1 = cv2.imread("LabPhoto1.jpg", cv2.CV_LOAD_IMAGE_COLOR)

2. Print the dimension of images

3. Convert the color images to gray scale images

   Example: grImg1 = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY)

   Show the gray scale images in two separate windows and print them.

4. Save the gray scale images

   Example: cv2.imwrite('grayImage1.jpg', grImg1)

5. For the first gray scale image, find the good features for tracking purposes

   Example:
   feat1 = cv2.goodFeaturesToTrack(grImg1, cornerCount, qualityLevel, minDistance)
   feat1 = feat1.reshape((-1, 2))

   where cornerCount = 200 is the maximum number of feasture points to be extracted;
   qualityLevel = 0.001 is the parameter characterizing the minimal accepted
   quality of image corners. Higher number means more stringent.
   minDistance = 9.0 is the minimum acceptable Euclidean distance between

feature points

6. Perform Lucas Kanade Tracking with Image Pyramid

   First, refine the feature (corner) point locations:

   ```
   criteria = (cv.CV_TERMCRIT_ITER | cv.CV_TERMCRIT_EPS, 80, 0.0001)
   win = (3,3)  # actual size is 3*2+1 x 3*2+1
   zero_zone = (-1,-1)   # no dead zone
   cv2.cornerSubPix(grImg1, feat1, win, zero_zone, criteria)
   ```

   Next, perform Lucas Kanade tracking with image pyramid:

   ```
   feat2 = np.copy(feat1)
   feat2, status, err = cv2.calcOpticalFlowPyrLK(grImg1, grImg2, feat1, feat2)
   ```

7. Load the color images again, overlay the feature points and show and print the resultant images.

   Example:

   ```
   im1 = cv2.imread("LabPhoto1.jpg", cv2.CV_LOAD_IMAGE_COLOR)
   cv2.namedWindow("Picture1")
   for (x,y) in feat1:
        cv2.circle(im1, (int(x), int(y)), 3, (255, 255, 255), -1)
   cv2.imshow("Picture1", im1)
   if cv2.waitKey(0) == 27:
      cv2.destroyAllWindows()
   # Save marked images.
   cv2.imwrite('LabPhotoTracking1.jpg', im1)
   ```

   Do the same for image 2.

8. Manually inspect the tracking results. Identify the points that were not tracked accurately i.e. points that were off by more than half a window size. Explain why do you think the tracking was wrong.

Submission Instruction

Submit the following at the end of the lab session:

1. Print-out of your Python program.
2. Print-out of the saved marked images in Q7 and your answer for Q8.
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_Lab3.

Remember to write your name on the hardcopy print-outs.