# CS4243 Assignment 2

# ===================

# Name: Tay Yang Shun

# Matric: A0073063M

import sys

import cv2

import cv2.cv as cv

import numpy as np

import math

# Example usage:

# python assg2.py labPhoto.JPG

IMAGE\_FILE\_NAME = sys.argv[1]

# Maximum number of corners to return

TRACKING\_CORNER\_COUNT = 200

# Minimum possible Euclidean distance between the returned corners

TRACKING\_MIN\_DISTANCE = 9.0

# Threshold

THRESHOLD = 0.001

# Half a window size

SIZE = 5

SQUARE\_COLOR = (0, 0, 255, 0)

SQUARE\_THICKNESS = 2

SQUARE\_LINE\_TYPE = 8

SQUARE\_SHIFT = 0

colored\_image = cv2.imread(IMAGE\_FILE\_NAME, cv2.CV\_LOAD\_IMAGE\_COLOR)

image = cv2.imread(IMAGE\_FILE\_NAME, cv2.CV\_LOAD\_IMAGE\_GRAYSCALE)

img\_height, img\_width = image.shape

def pad\_image\_border(img):

# Gives an extra border to the right and bottom of the image

return np.lib.pad(img, ((0, 1), (0, 1)), 'edge').astype(int)

# Calculate dI/dx for each pixel

print 'Calculating dI/dx value for each pixel...'

image\_Ix = pad\_image\_border(np.copy(image))

for y in range(img\_height):

for x in range(img\_width):

image\_Ix[y][x] = image\_Ix[y][x+1] - image\_Ix[y][x]

print 'Done calculating dI/dx values!'

# Calculate dI/dy for each pixel

print 'Calculating dI/dy value for each pixel...'

image\_Iy = pad\_image\_border(np.copy(image))

for i in range(img\_height):

for j in range(img\_width):

image\_Iy[i][j] = image\_Iy[i+1][j] - image\_Iy[i][j]

print 'Done calculating dI/dy values!'

image\_eigenvalues = np.zeros(image.shape)

print 'Calculating minimum eigenvalues...'

count = 1

total\_pixels = len(range(SIZE, img\_height - SIZE, SIZE)) \* \

len(range(SIZE, img\_width - SIZE, SIZE))

for y in range(SIZE, img\_height - SIZE, SIZE):

for x in range(SIZE, img\_width - SIZE, SIZE):

print 'Calculating minimum eigenvalues:', round(float(count) / total\_pixels \* 100, 4), '% complete'

mat = np.zeros((2, 2))

for v in range(y - SIZE, y + SIZE):

for u in range(x - SIZE, x + SIZE):

mat[0][0] += image\_Ix[v][u] \*\* 2

mat[0][1] += image\_Ix[v][u] \* image\_Iy[v][u]

mat[1][0] += image\_Ix[v][u] \* image\_Iy[v][u]

mat[1][1] += image\_Iy[v][u] \*\* 2

mat /= (2 \* SIZE + 1) \*\* 2

eigenvalues, eigenvectors = np.linalg.eig(mat)

min\_eigenvalue = min(eigenvalues)

image\_eigenvalues[y][x] = min\_eigenvalue

count += 1

print 'Done calculating minimum eigenvalues!'

eigenvalues\_data = []

# Get a list of eigenvalues and its pixel coordinates

for y in range(SIZE, img\_height - SIZE, SIZE):

for x in range(SIZE, img\_width - SIZE, SIZE):

if image\_eigenvalues[y][x] > THRESHOLD:

eigenvalues\_data.append([image\_eigenvalues[y][x], (int(x), int(y))])

corners\_list = sorted(eigenvalues\_data, key=lambda e: e[0], reverse=True)

def dist(pt\_a, pt\_b):

return math.hypot(pt\_a[0] - pt\_b[0], pt\_a[1] - pt\_b[1])

top\_corners\_list = []

for i in range(TRACKING\_CORNER\_COUNT):

if len(corners\_list) > 0:

current\_corner = corners\_list[0]

top\_corners\_list.append(current\_corner)

corners\_list.pop(0)

# Remove remaining corners that are too near to the corner just added

corners\_list = [corner for corner in corners\_list if \

dist(corner[1], current\_corner[1]) > TRACKING\_MIN\_DISTANCE]

else:

break

for (eig, pt) in top\_corners\_list:

cv2.rectangle(colored\_image, (pt[0]-SIZE, pt[1]-SIZE), \

(pt[0]+SIZE, pt[1]+SIZE), SQUARE\_COLOR, \

SQUARE\_THICKNESS, SQUARE\_LINE\_TYPE, SQUARE\_SHIFT)

# Save new image with good features indicated

file\_name, file\_extension = IMAGE\_FILE\_NAME.split('.')

new\_file\_name = file\_name + '-features.' + file\_extension

cv2.imwrite(new\_file\_name, colored\_image)

print IMAGE\_FILE\_NAME, ' with good features marked, saved as \'' + new\_file\_name + '\''

