```
# 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 + '\''
```