NGUYEN, Réal **COMP 499 SEC. GG**
ID# 27566263 **DUE 12/02/2019**

ASSIGNMENT 2 REPORT

Feature detection

First, get both the images' horizontal and vertical gradients (Ix and Iy, respectively). Afterwards, apply the Harris corner detector by getting Ix^2, Iy^2, and IxIy, then getting the corner response function R at every pixel of each image. If R is not a number, then set it to 0; if it is above the threshold (default value = 0.00025) then keep it; else do nothing. The features here are represented by dots with their respective intensities preserved on a black background.

Get the local maxima around a 3x3 window. If the window is at the image's edge, then pad it with 0's. Only keep the pixel with the highest intensity in this window; all other pixels in the window are suppressed to 0.

Finally, use OpenCV's built-in drawKeypoints() method to illustrate the features on the original colour image. Images of the keypoints can be viewed in the "keypoints" folder of this deliverable.

Feature description

This assignment uses the SIFT descriptor. The window size is 16x16 and the cell size is 4x4.

First, get the coordinates for all the features. These will be used to anchor the descriptor windows.

Second, get the orientations for every pixel in the images with the get_orientations() method. The return value of this method is a 2D matrix with the same dimensions as the image that's passed as a parameter, with each element in the matrix being an orientation in radians. Because arctan's codomain is -pi/2 to pi/2 radians, all elements in the matrix lie between these two values. Note that some values had to be set to 0 by the program to avoid divisions by 0.

Third, create and centre the descriptor windows around the feature coordinates. If the windows go beyond the boundaries of the image, do not create the descriptor for this feature and move on to the next one. Otherwise, get all the orientations within this window.

Fourth, get the cells from this window with the get_cells() method. Assuming the window and cell sizes are left to their default values, then the return value of this method is a list of lists, with each sublist being 1D arrays with 4x4 = 16 orientations and the entire list representing the descriptor window. This makes it easier to iterate through in the next step.

Finally, create the descriptor in the get_descriptor() method. For each cell in the window, create an orientations histogram. The histogram itself is a simple 1D array with 8 elements. Elements 0 through 3 are positive angles (0 to pi/2 rads) and elements 4 through 7 are negative angles (0 through -pi/2 rads). If an angle is within a certain bracket, then increment its respective bin. The return value is a 1D array with 128 elements, with each element representing the count of an orientation within a certain bracket, for a certain cell. Note that these values are neither thresholded nor normalized, as I could not find a reliable way to do so on a count-based histogram in time.

Feature matching

In the match_features() method, the descriptors for the first and second images are taken as parameters. For every feature in the first image, calculate the squared sum difference (SSD) with every feature in the second image. If the SSD is under the threshold (the default value is 350), then add it to a list of distances for this feature. Keep the lowest and second lowest SSDs for the ratio test. This method is predictably quite slow, taking a few minutes to get through ~2700 features in the first image.

Unfortunately, there was no time to properly test feature matching with OpenCV's built-in drawMatches() method.