HW2 Writeup

Ryan D'Mello

Interest Point Operator

I used the Harris Corner Detector, and used Szeliski as a reference. First, we compute the Sobel Gradients, and then take the terms of the Hessian. We then convolve the 3 different terms of the Hessian. In order to calculate the Harris Response, we use the Hessian terms to determine the trace and determinant. We perform nonmax suppression in order to get our response. We then aim to find local maxima as the detected interest points.

Feature Descriptor

I followed the 3x3 grid design provided in the assignment description, ultimately resulting in a feature vector of size 72 being returned by combining the 8 orientation bins. We have 9 windows from the grid (3x3). I had a helper function update_feats that updated feats in each iteration of counting.

Feature Matching

The KDnode class attributes are data, indices, left, and right.The implementation has the following functions:

match_features()

naive_linear_search()

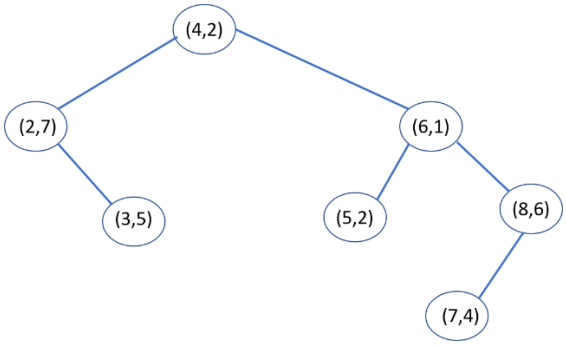
create_KDtree()

Nearest_Neighbor_KD()

Nearest_Neighbor_Feature()

The function that builds the KD tree is create_KDtree(). It is a recursive function that finds the first left or right spot at which to insert. It first determines whether to insert into the left subtree or right subtree of the root node. Once this is determined, it recursively inserts into the left/right subtree. A very simple example would be nodes with values (4,2), (6,1), (2,7), (3,5), (8,6), (5,2), (7,4) (with the x-coordinate compared first). This would yield the KD tree shown

below because (4,2) becomes the root, then (6,1) is to the right of the root (because 6 > 4). To insert (5,2) we simply do a recursive insertion, this time starting with the node (6,1), to insert (3,5) we would recursively insert starting at (2,7), and so on.



The match_features() function has a parameter “mode” and calls naive_linear_search() if the mode is “naïve” and Nearest_Neighbor_KD() if the mode is not naïve.

The Nearest_Neighbor_Feature() function essentially takes one feature and a set of other features, determining which of the other features is closest to that one, and also computes a score reflecting how “unique” that nearest neighbor is.

naive_linear_search() takes two feature sets and for each one in the first it finds the closest one in the second (by calling Nearest_Neighbor_Feature()). It is not as efficient as Nearest_Neighbor_KD().

Finally, the Nearest_Neighbor_KD() function takes two sets of features, creates the KD tree for the second set and then uses that tree (using a split across the median at each stage) to find the nearest neighbor (in the second set) for each of the features in the first set.

Hough Transform

In this problem of the assignment, we aim to estimate a translation vextor t – we return each component of the vector tx and ty. Moreover, we take in the x and y coordinates of the interest points for two images and subsequently count votes. We use the matches input to determine the difference between the two images’ respective interest point coordinates. We then find the minimum and maximum, and create the votes array. Ultimately, we find the max_score by iterating through the array, and creating the returned vectors tx and ty.