CS 4476 Project 3

<Albert Xing>
<axing6@gatech.edu>
<axing6>
<903337032>

Part 1: Harris Corner Detector

<insert visualization of Notre Dame interest points from proj3.ipynb here>

< insert visualization of Rushmore interest points from proj3.ipynb here >

Part 1: Harris Corner Detector

< insert visualization of Gaudi interest points
from proj3.ipynb here >

Part 1: Harris Corner Detector

<Describe how your implementation mirrors the original harris corner detector process. (First describe Harris) What does each helper function do? How are the operations we perform equivalent and different?)>

Our implementation is similar to the pseudocode implementation of algorithm 4.1. We begin by finding the Gaussian kernel returning a matrix based on the kernel size and sigma. Taking the provided image array, we use the sobel filters to find the differentiated kernels along the x and y. Using the image gradients, we find the second moments using the Gaussian filter, which is similar to modern approaches while the classical Harris used a [-2 1 0 1 2] filter. With the second moment returns, we create the auto-correlation matrix. Harris uses an empirical constant of alpha = .06 compared to our alpha = .05. Finally, we implement our non-max suppression with a median value and neighborhood size, while they find the local maxima above a certain threshold. We also do not compute multiple images corresponding to the outer product of the gradients.

Part 2: Sift

<Describe how your implementation mirrors the Sift Process. (First describe Sift) What does each helper function do? How are the operations we perform equivalent and different?)>

For the SIFT implementation, we created the helper function to get the magnitudes and orientations of gradients at every pixel location. Using the get_feat_vec function we can return the feature vectors by creating a grid of 4x4 cells around the center. By using the feature_width parameter we create 16 of these cells and find an aggregated histogram from them. The histogram values are normalized and raised to power of .9 to represent the orientation and magnitude more accurately. Using this helper function, we can get the features by passing in the image array and the interest points. We find the image gradients, then use that for the magnitude and orientation, and we can find the features.

Part 3: Feature Matching

<insert feature matching visualization of Notre Dame from proj3.ipynb>

<insert feature matching visualization of Rushmore from proj3.ipynb >

Part 3: Feature Matching

<insert feature matching visualization of Gaudi from proj3.ipynb >

<Describe your implementation of feature matching.>

Results: Ground Truth Comparison

<Insert visualization of ground truth comparison with Notre Dame from proj3.ipynb here>

<Insert visualization of ground truth comparison with Rushmore from proj3.ipynb here>

Results: Ground Truth Comparison

<Insert visualization of ground truth comparison with Gaudi from proj3.ipynb here>

<Insert numerical performances on each image pair here. Also discuss what happens when you change the 4x4 subgrid to 2x2, 5x5, 7x7, 15x15 etc?>

By changing the size of the subgrids, it affects the samples in the histograms and amount of frequencies for bins. Changing the subgrids to large sizes will provide more samples for more accurate results.

Extra Credit: Hyperparameter Tuning part 1

<Insert images of the ground truth correspondence and their corresponding accuracies for varying sigma in the second moments [3, 6, 10, 30] >

When changing the values for large sigma (>20), why are the accuracies generally the same?

Extra Credit: Hyperparameter Tuning part 2

<Insert images of the ground truth correspondence and their corresponding accuracies for varying feature width in the SIFT [8, 16, 24, 32] >

What is the significance of changing the feature width in SIFT?

Extra Credit: Accelerated Matching

<Insert Runtime/Accuracy of your faster matching implementation. What did you try and why is it faster?>