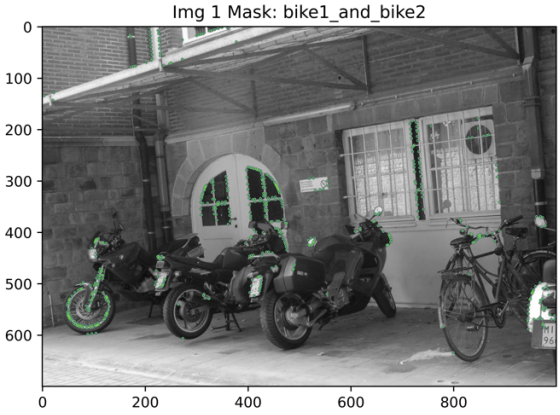
CMP_SC 8690: Computer Vision
Homework 2: Feature detection, description, and matching
By: Mikey Joyce
Due: 2/20/2024

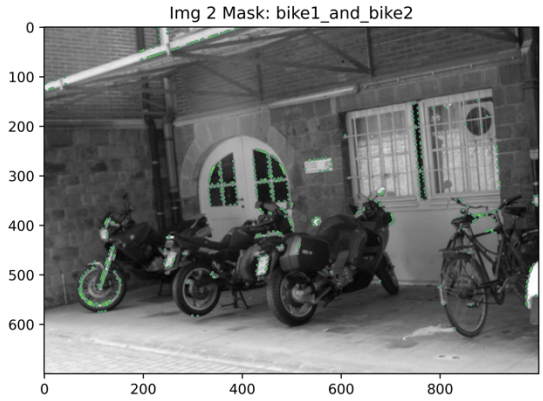
Abstract:

In this assignment, we are to implement a full feature detection, description, and matching pipeline. Much of this assignment is ambiguous to allow us to take our own direction with the implementation. I conducted two different official experiments in this submission, comparing the matching between bike1 and bike2 is the first experiment and the second experiment compares bike1 and bike5.

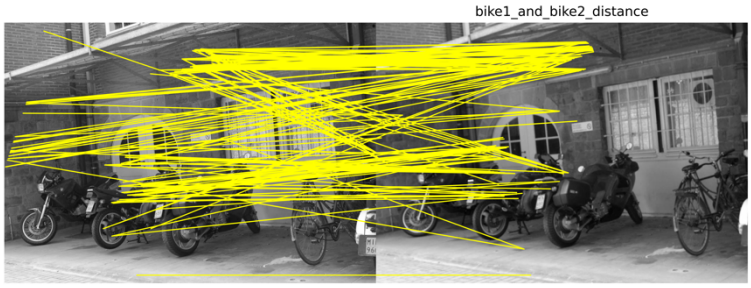
Experiments and Results:

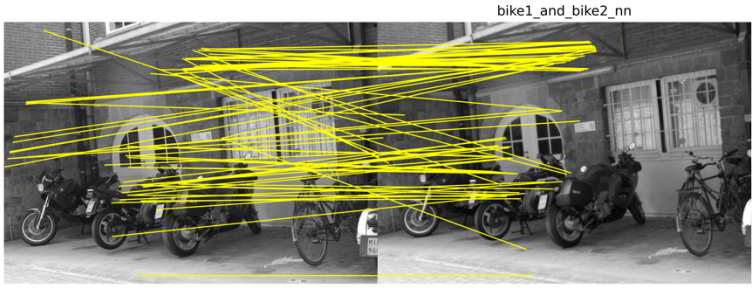
Experiment #1: Matching Bike1 and Bike2

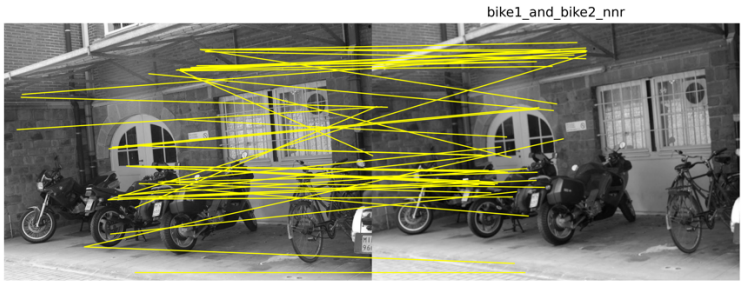




Above shows the detected features on the bikes, it may be hard to see on this paper, refer to the results folder in the project directory to see the detected features in a larger picture.

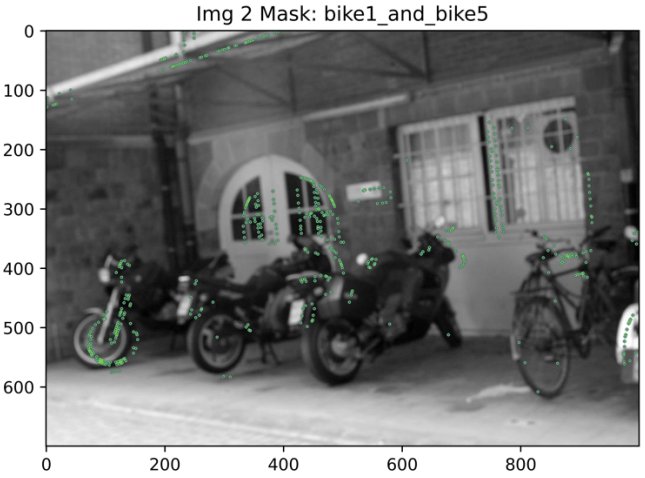






	Distance Thresh	Nearest Neighbor	Nearest Neighbor Ratio
# detected points	379	379	379
# matched points	188	84	49
TPR	76 → 40%	39 → 46%	29 → 59%
FPR	112 → 60%	45 → 54%	20 → 41%

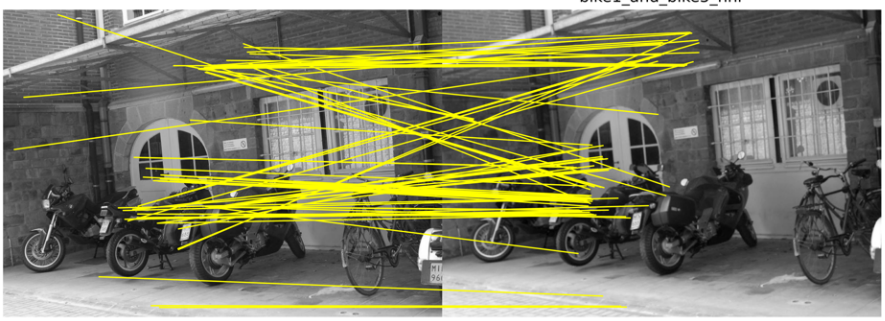
Experiment #2



bike1_and_bike5_distance

bike1_and_bike5_nn

bike1_and_bike5_nnr



	Distance Thresh	Nearest Neighbor	Nearest Neighbor Ratio
# detected points	379	379	379
# matched points	339	95	76
TPR	134 → 40%	38 → 40%	31→ 41%
FPR	205 → 60%	57 → 60%	45→ 59%

Conclusion:

In this assignment we were asked to implement a full feature detection, descriptor, and matching pipeline. For the feature detection I utilized the Harris detector to detect the features. I believe this was a strong feature detection measure and utilized it due to the strong documentation for implementation that was provided in the slides. After this I utilize scikit images peak local max to find the maxima that the Harris detector found while also pruning the number of detected points to the desired amount. Since some of the detected maxima may be on the border, I prune the border points afterwards, so the number of detected points usually isn't the same as the target number of detected points. This concluded the detection part of the assignment. Next is the feature descriptor part. I decided not to opt for the bonus section since this assignment already was a large commitment as is, so I just picked a neighborhood around each detected point to be my feature descriptor. After this we do the matching part of the assignment. We were asked to implement 3 different matchers. The first matcher is just looking at the distances and utilizing a threshold to ensure that the distance is close enough to be a match, if that distance is less than the given threshold, then it is a match. Because of this primitive nature of the distance matching, it is expected that there are many more matched points. The second matcher was the nearest neighbor matcher. In this matcher we just look at the distances and pick the closest distance, given that this distance is less than the given threshold. The last matcher is the nearest neighbor ratio matcher which we essentially look at the closest distance and the next closest distance, if the match of the second closest distance is close to the best match, then we do not consider that a match. As we can see from the performance of the three matchers, the nearest neighbor ratio algorithm performed the best, even getting a positive match rate on the bike1 to bike2 experiment. The next best algorithm was the nearest neighbor algorithm. Lastly, the distance thresholding performed the worst, which was expected because it is not very strict on what it

considers to be a match. I think this pipeline is somewhat decent, the descriptor choice that I decided to implement (Euclidean distance) was definitely not the most robust out of all of the descriptors that I could have picked, but it was the easiest. One way that this pipeline could improve would be to use built in functions for the feature detector like the Harris detector. Another way this could be improved would be to just use something like the SIFT matcher, because that is already quite robust.

References:

- Libraries and tools: PyCharm, OpenCV, NumPy, Matplotlib, skimage, Preview.
- Lec06_FeatureMatching_extended.pdf
- CV2024_HW1B.pdf