COMPSCI 773 Assignment 1

Stephen Hallett, Kelham Rawlinson, Zak Quor

Chapter 1

Harris Corner Detection

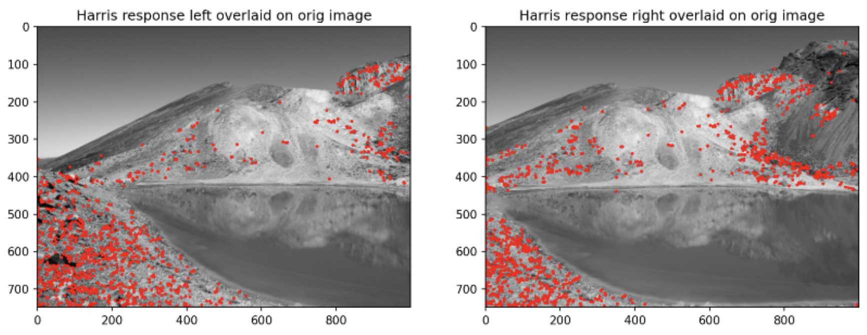

Harris response left overlaid on orig image
Harris response right overlaid on orig image

Figure 1.1: Output of our Harris corner implementation on the Tongariro images.

While implementing the Harris corner detection algorithm, we experimented with several parameters, including the Gaussian kernel size, alpha, and non-maximal suppression window size. The overall sum of our best decisions being Figure 1.

On the topic of experimenting with parameters, we found that slightly increasing the size of the Gaussian kernel, from what seems to be the standard 3x3, up to 5x5, helped to improve performance as well as further smoothing the image. By increasing the kernel width, the derivatives were blurred more. This led to less noise around the detected edges. For corner detection, this helped to reduce the number of false corners that were likely detected due to slight variations around edges and actual corners. This can be seen in the following two images, Figure 1.2 and Figure 1.3. In the right image, we can see that fewer corners are detected between other corners (i.e., along edges).

In our experiments, we found almost no positive difference when varying the alpha parameter between 0.04 and 0.06. Because of this, we keep it at 0.04.

We also found that increasing the non-maximal suppression window size led to multiple benefits. Namely, it helped to further prevent very dense clusters of corner points (clusters are not useful for image stitching due to local ambiguity). Secondly,

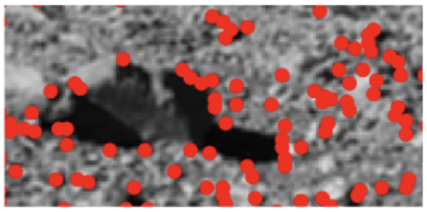


Figure 1.2: Output of our Harris corner implementation on the Tongariro images.

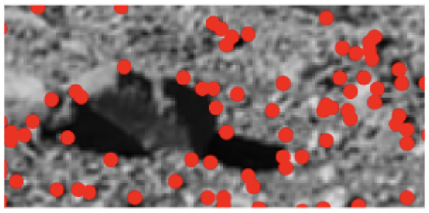


Figure 1.3: Output of our Harris corner implementation on the Tongariro images.

by increasing the NMS window size, we could remove some overall strong corners that are not useful for feature matching. Some of these strong corners belong to clusters of strong points, and because our implementation outputs the top 1000 points (based on cornerness score), removing these strong points had the side effect of allowing weaker but more useful/varied features to be detected across the image. This may lead to better matching in the long run, even though the features are not as strong. Together, these two decisions caused the detected corners to be more evenly distributed across the entire image. This is critical with panorama stitching, as if dense regions of corners were to be eliminated between photos (due to movement/translations between image captures), it could become difficult to perform image alignment due to the low number of corresponding features.

Chapter 2

Assignment Extensions

2.1 Difference of Gaussian Blob Detection

For an alternative feature detection method, we implemented a blob detector using the difference of Gaussian (DoG) method. The output of DoG can be demonstrated by uncommenting the line *show_DoG(px_array_left, px_array_right)* in CS773StitchingSkeleton.py and then running that file.

Compared to the Harris corner detector, there were some significant changes which we observed from the DoG method. Particularly, we found that the DoG method is much more sensitive to changes in the image size, image detail, and hyperparameters of the DoG implementation. This is best demonstrated by the images in Figures 2.1 and 2.2 below, which show example effects of these sensitivities.

The first image demonstrates how sensitive the DoG blob detector is to different values of sigma - the parameter which handles the strength of the Gaussian blur at each level of the image pyramid. We can see as the sigma increases, we see larger blobs in general, and also we see that the blobs focus on larger areas of the image, particularly the central pattern on the butterfly is covered in significant blobs when the sigma value is 2, but with a smaller sigma, the DoG detector acts closer to a corner detector.

With the same parameters but differently detailed images, we see that the blob detector is most visually effective with these parameters when the image is smaller, and scaling up to a larger image has the effect of acting more or less like a corner detector, creating only extremely small blobs since there is so much detail to be represented through the blobs. In order to avoid getting tiny blobs on more detailed images, the sigma and nms window sizes would need to be scaled up drastically, further demonstrating how sensitive the detector is to its parameters.

This gives a strong indication of how each detector can be extremely useful in different situations, and that there is no blanket optimal detector for every image. For instance, when trying to find key features of a patterned butterfly, with the right param-

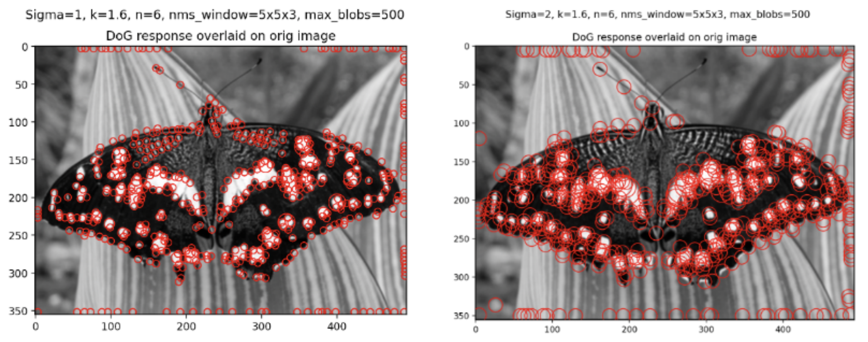

Sigma=1, k=1.6, n=6, nms_window=5x5x3, max_blobs=500
DoG response overlaid on orig image
Sigma=2, k=1.6, n=6, nms_window=5x5x3, max_blobs=500
DoG response overlaid on orig image

Figure 2.1: DoG blob detection on the same image with differing values for the sigma parameter.

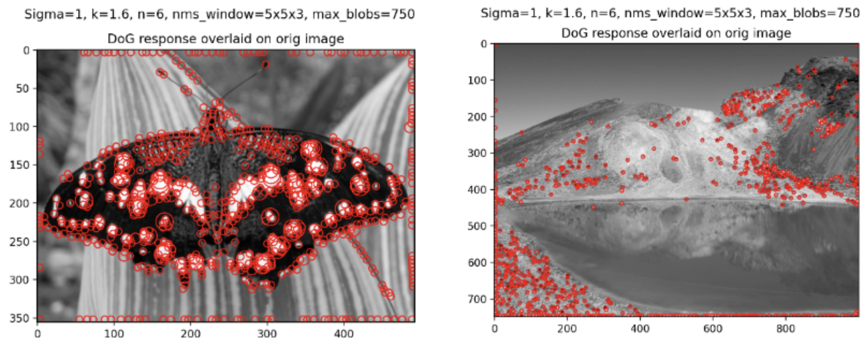

Sigma=1, k=1.6, n=6, nms_window=5x5x3, max_blobs=750
DoG response overlaid on orig image
Sigma=1, k=1.6, n=6, nms_window=5x5x3, max_blobs=750
DoG response overlaid on orig image

Figure 2.2: DoG blob detection on images with different levels of detail and identical DoG parameters.

eters, the DoG filter performs extremely well and has a clear visual advantage over a corner detector. However, in more detailed images like that of Tongariro, for example, there seems to be no meaningful advantage of using DoG over a corner detector.

2.2 Gradual Thresholding

When the Harris corner detector was tested on different images, we found that a ‘good’ cornerness threshold varied significantly between images. With a single, fixed threshold, some images had very few features eliminated and some had many features eliminated. This meant that finding a way to change the threshold value based on the

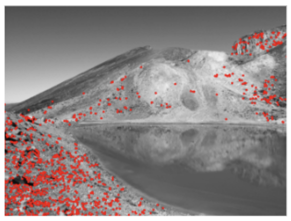
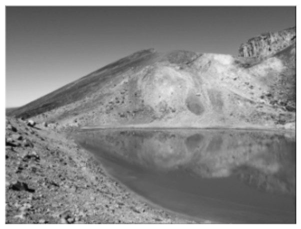
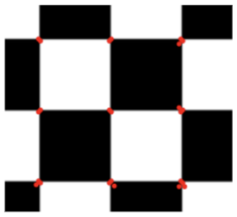
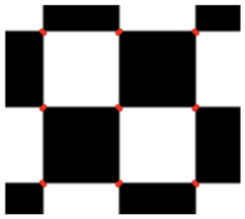
Threshold: 1x10^2	Threshold: 1x10^9
	
	

Figure 2.3: Effect of setting a fixed threshold across two different image sets.

particular image characteristics seemed very necessary.

A research paper by Vino, G., and Sappa, A. D. (2013)[1] developed an adaptive thresholding method based on the idea of comparing each candidate corner point to a threshold determined by an equation including itself and its immediate neighbours rather than a global threshold score. It attempts to enable the concept of considering how distinct or strong a feature is in its region, which is useful for feature detection for image stitching.

This method is enabled in our code by setting the adaptive_threshold parameter to True in the get_harris_corners function in CS773StitchingSkeleton.py.

The high level concept of how the code works is as follows: First accept any corners that are so strong that they are obviously great features. The second is to cut off any features that are obviously weak, leaving a set of features somewhere in the middle. These features are iterated through, and have a kernel applied to them which values their orthogonal neighbours more highly than their diagonal. For each feature, it computes a strength score, normalises it, and then sets an adaptive threshold based on the original cornerness score (the equation to set the threshold this is contained in Vino and Sappa's research paper and based on empirical testing results). Finally, if the normalised score is above the adaptive threshold, then the feature is considered strong

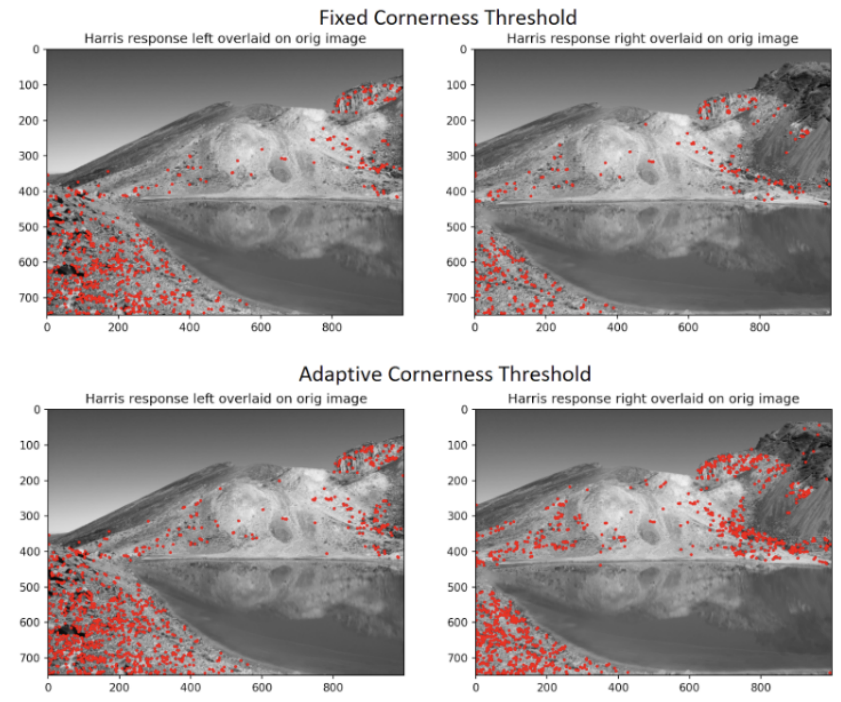

Fixed Cornerness Threshold
Harris response left overlaid on orig image
Harris response right overlaid on orig image
Adaptive Cornerness Threshold
Harris response left overlaid on orig image
Harris response right overlaid on orig image

Figure 2.4: Features when using a fixed vs adaptive cornerness score threshold.

enough and it is stored in the feature array.

The difference in results produced between fixed and adaptive thresholds for the same image reveal themselves most significantly when observing that features are slightly more spread out. This is due to features which would previously have been suppressed being given a chance to appear and demonstrate that they are strong for their region, as shown in Figure 2.4.

Chapter 3

Contributions

Zak:

- Gradual/adaptive threshold interpretation and implementation
- Simple Thresholding
- Non-max suppression
- Harris Corner final results sorting and listing
- Report writing Gradual Thresholding section
- Report editing - minor formatting, proof-reading

Kelham:

- Gaussian filter implementation and blurring of image derivatives
- Cornerness score calculation
- Researched various Adaptive Harris threshold methods
- General testing and debugging of Harris, DoG and Adaptive thresholding
- Harris parameter experimenting and testing
- Report writing (excluding the DoG section and gradual thresholding details)

Stephen:

- DoG implementation
- Sobel filter implementation
- General debugging of Harris corner detector
- Report writing DoG section and LaTeX report formatting

Bibliography

[1] G. Vino and A. Sappa, “Revisiting harris corner detector algorithm: A gradual thresholding approach,” vol. 7950, pp. 354–363, 06 2013.