Phase 3: Image transformation

Harris Corners Pipeline

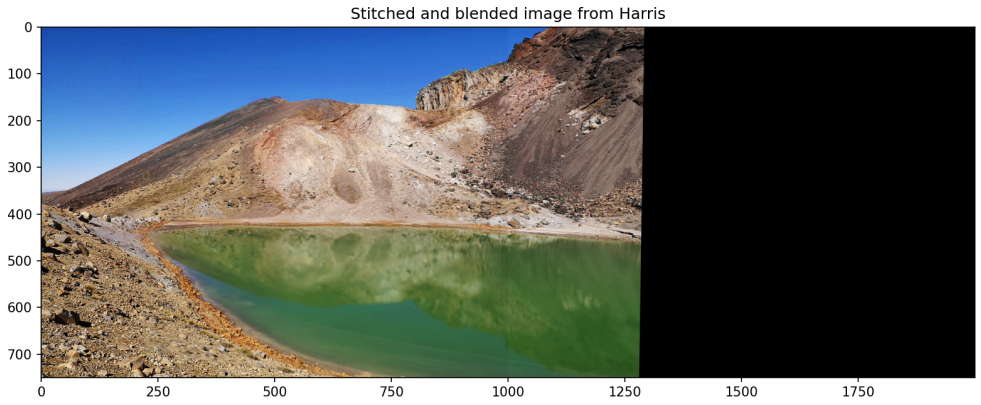


Figure 1: Image stitching results using the Harris corner pipeline

Direct Linear Transformation (DLT):
In order to identify the best fitting homography between the matching points, we implemented DLT in order to fit the homography to a given set of matches chosen via RANSAC.

RANSAC:
Two key parameters of the RANSAC method are the number of iterations to perform and the inlier distance threshold. From experimenting with different parameters, we found that 1000 iterations were a suitable number and that searching for any more potential transforms was unlikely to yield significantly better results. We set the inlier distance threshold to 3 (pixels) and found that it worked quite well. Small increases to this threshold did not have much of an effect beyond the final homography being computed on a few more matches. This is likely due to the fact that most of the matches are quite good and consistent, meaning most of the outliers are at a large distance. When we set the threshold to a very large value (in the hundreds), then many bad matches are included as inliers, leading to very clearly wrong transformations being produced.

Stitching:
The inverse warping algorithm is used to stitch images together once the transformation is found. Pixel colours are blended by taking the average where the two images overlap. Although this blending method is very simple, it leads to very visible, sharp borders between the two images. This can be seen in Figure 1 at approximately x = 1000. To rectify this, we would need to experiment with more complicated, alternative blending functions such as blending gradually rather than by zone-based blending.

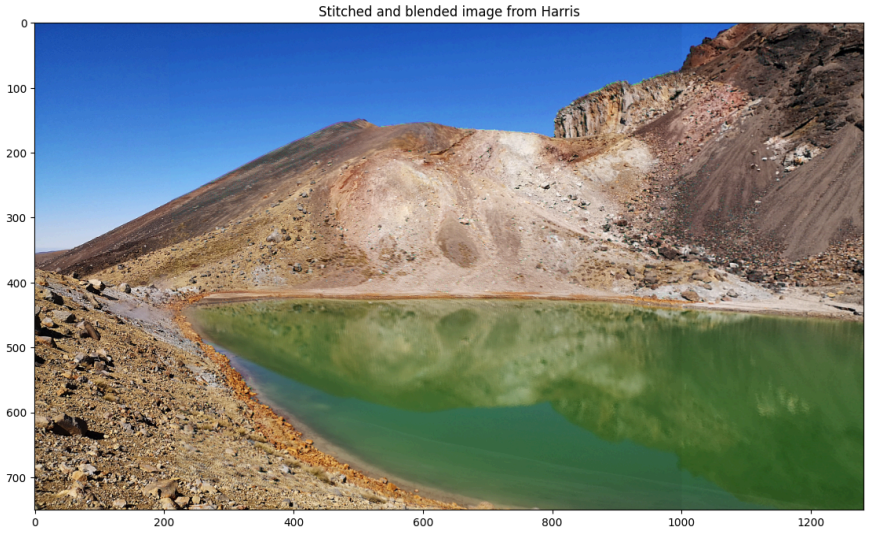


Figure 2: Cropped stitched image

We also included the option to crop stitched photos to remove the excess black area from the canvas (See Figure 2). This works by scanning the image inwards from each side until a line with no perfectly black pixels is found. This means we have hit the start/end of the image and the point where we need to cut off the rest of the canvas. This method works reasonably well, although it can fail if the image itself contains pixels with colour (0, 0, 0). This is, however, quite unlikely. This can be enabled by setting the crop parameter of the function warp_image() to true in CS773StitchingSkeleton.py.

The approximate computational time from start to end of the Harris image stitching pipeline on a low-mid level laptop = 161 seconds.

Extension 1: DoG Pipeline

Given the successful results we saw from phases 1 and 2 with our extensions for DoG feature extraction and matching algorithms, we decided to complete the image stitching process using the DoG features. The RANSAC algorithm was able to be adjusted to take in the paired blobs (as opposed to paired corners), extract the coordinates from the blobs and find the best-fit transform in the same way it does for corners.

Because our other extensions (adaptive thresholding for Harris corner detection and HoG feature descriptors) were incompatible with the two DoG extensions, they were not used in this pipeline in any way.

With no scale change between the left and right images, the resulting image is practically identical to the stitched image from the Harris corners, even after multiple completions of the full pipeline. It is, however, considerably slower due to the DoG detection and matching methods taking a significant amount of time, especially when compared to Harris Corners performing the same methods.

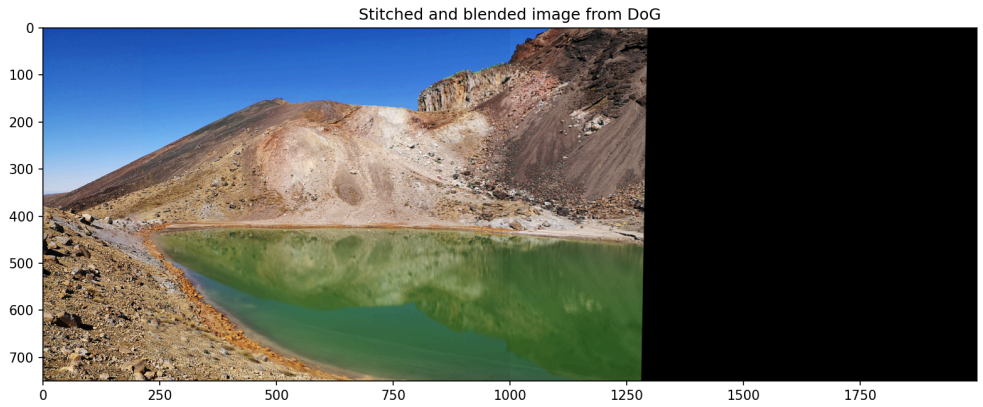


Figure 3: Stitched image (with no scale change between left and right image) using DoG pipeline

The approximate computational time from start to end of the DoG pipeline on a low-mid level laptop = 296 seconds. Although DoG takes much longer to compute, it has the key advantage of being scale invariant. When we experimented with DoG featuring matching for our extension in Phase 2, we found that the Harris detector performed very poorly with zoomed-version and was unable to find many good matches at all. This caused RANSAC and DLT to completely fail at producing a reasonable homography (Figure 4). On the other hand, the DoG pipeline was able to detect features perfectly fine after the scale change, meaning the matching and transformation were able to proceed normally, leading to a good result (Figure 5).

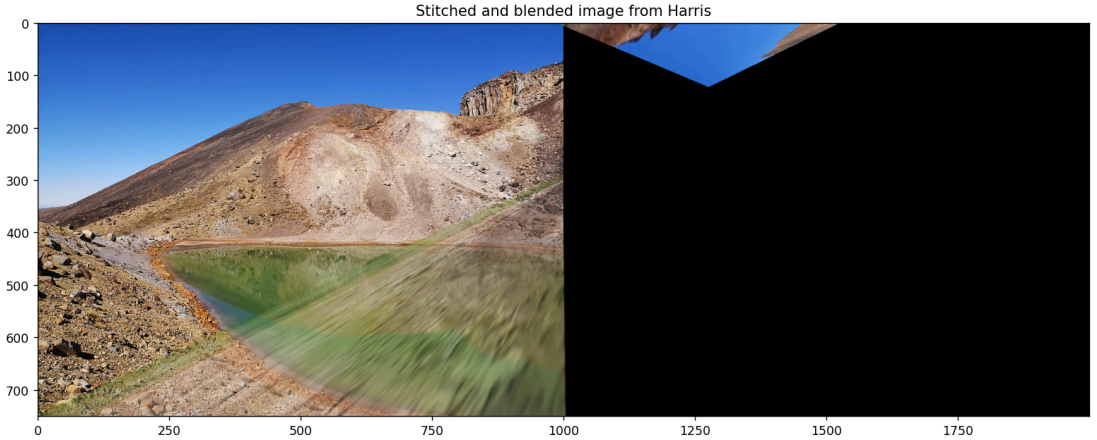


Figure 4: Stitched image (with a zoomed-in version of right image) using Harris pipeline. The homography completely fails.

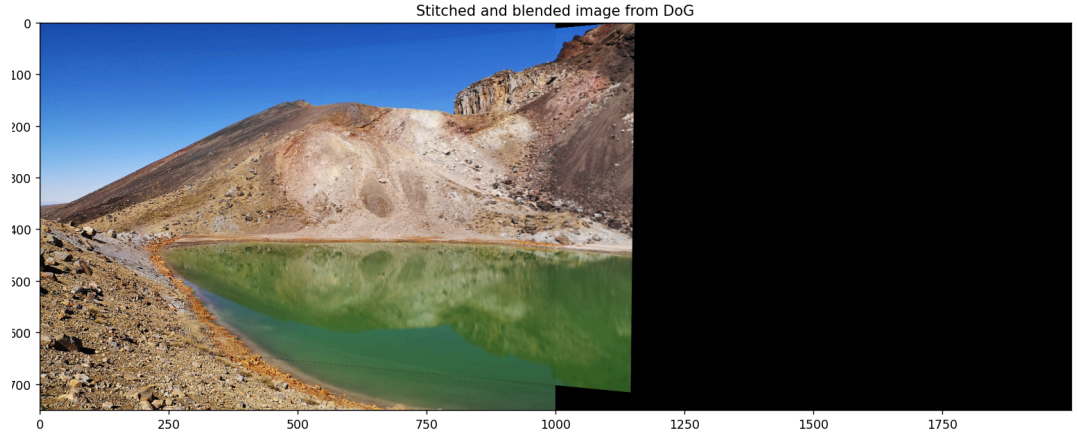


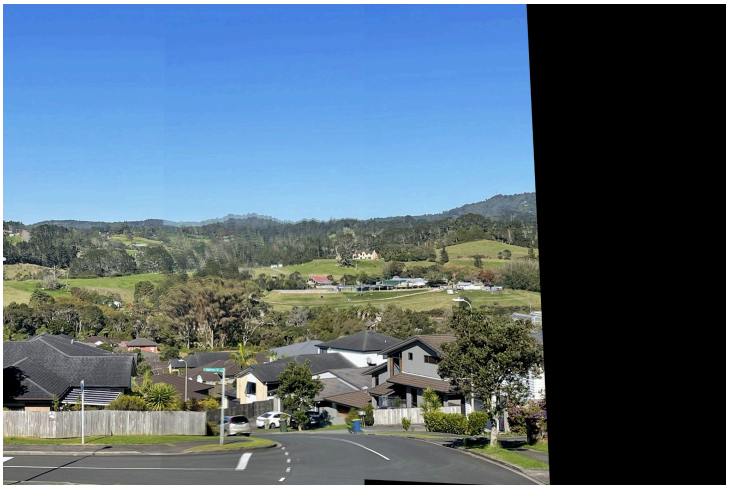
Figure 5: Stitched image (with a zoomed-in version of right image) using the DoG pipeline. Here, we can see the transform scales the images appropriately and perform a good perspective transformation.

Extension 2: Generalisation to other images

In order to assess how well our Harris corner detection image stitching method generalised to other environments, we took some extra photos to test on. We decided to experiment with some photos of the West Auckland landscape in order to see how the stitching pipeline performed in a setting where there was no one clear landmark as there was in the Tongariro images.

Figure 6: Three adjacent images of West Auckland (image 1, 2 & 3)

In Figure 6, we can see that the images have enough overlap such that there should be sufficient features for corner matching and stitching. However, we theorised before processing the images that there could be some issues with lining up the background landscape; given that most of the background is full of trees and similar features. Additionally since less than half of the image were foreground, we expected to see some additional issues, which we did find in our results.



Figures 7 & 8: Image 1 stitched with Image 2 (left) & Image 2 stitched with image 3 (right)

We found that our suspicions were correct since we managed to have two images that stitched together quite well, and another two which performed horribly. We suspect that images 1 & 2 (left) from Figure 7 stitched together so well because there were consistent landmarks across both images such as road markings, a street sign and a house. These features work extremely well with Harris corner detection since they are often straight, and have a lot of contrast in the background behind them. This also works very well with the sobel filter we apply. Images 2 and 3 on the other hand performed extremely poorly, likely since they lacked these same landmarks.

While there was a shared house between the two images, with an overall lack of landmarks, the corner matching algorithm had to rely on more generic features such as leaves on trees which resulted in a poor performance. This stark difference in results is very telling about the conditions in which Harris corner matching performs very well and very poorly. We found that the Harris corner detection works best for image stitching when there are a sufficient number of quality corners such as those on road markings and street signs, and performs much worse when there are a high quantity of low quality corner matches such as those with trees and grass. While the results weren't particularly satisfying, they do give a strong indication of the scenarios where Harris corner matching performs very well, and where it leaves a lot to be desired.

Stephen:

- Direct linear transformation (computing homography)
- Taking photos and applying stitching pipelines to the additional photos

Zak:

- RANSAC
- Completed DoG pipeline

Kelham:

- Inverse image warping
- RANSAC
- Debugging and testing full image transformation processes