Image Concatenation (Panaroma)

Rishabh Biyani

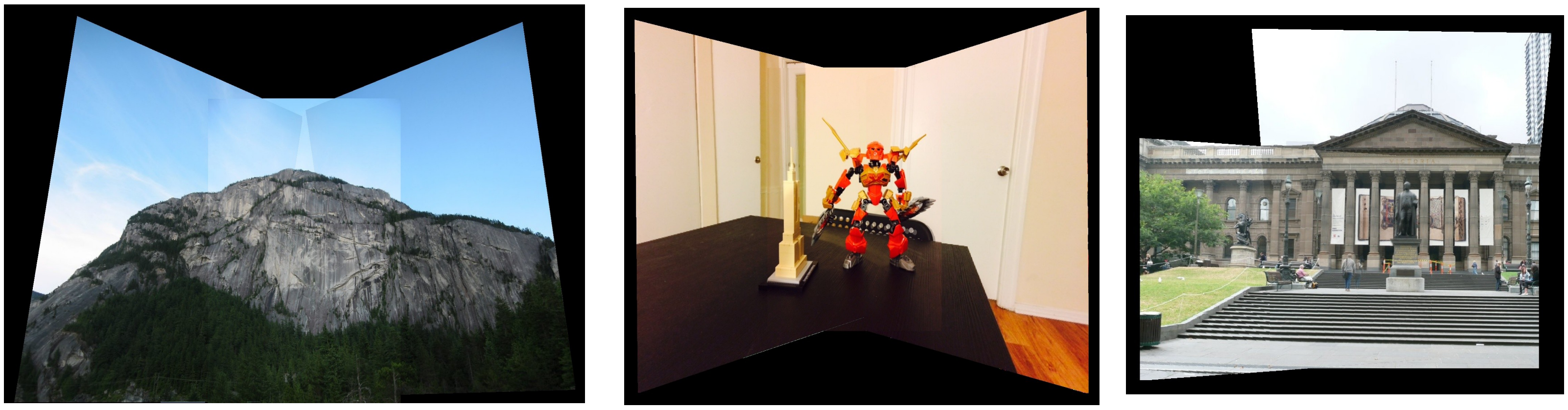

Figure 1: Outputs from the written Image Concatenation algorithm

Approach -
Briefly, Matching Points are extracted using SIFT features and corresponding descriptor. From the Matching Points, Inliers are computed for the Homography using RANSAC. Homography gives the correct transformation between images. Padding is performed using a written logic to make space in the base image for concatenating the other images. For concatenation, inverse homography on the base image points is done and potential co-ordinates of the next image is extracted. The Intersecting parts is recalculated using a average blend between the base and the next image. Parts that go beyond the base image is simply taken from the points in the next image. In the entire implementation, I have written my own functions for Homography Estimation and the Stitching and the Blending procedure. For Feature Extraction I have used the OpenCV Feature Detector Algorithm directly.

1. **Feature Extraction and RANSAC**: I have used the SIFT features for getting robust features across images and also for identifying potential matching points for homography. This extraction of matching point is done inside the function **getMatchingPoints.** Homography is calculated using Direct Linear Transformation technique and using RANSAC for outlier rejection. This is naturally evaluated from the next image to the base image. I have implemented functions for these in the **getHomography** and **getHomographyRANSAC.** The output of this function is verified b comparing with the findhomography function in OpenCV.
2. **Image Combine :** Combining the image needed to have a centre/base image across which we can stitch right and left images. For this project, I have identified these images manually. Once, the base image is identified, first I stitch left image and then the right image. For stitching, initially, I pad the base image using the extremes of the next image points – this gives an idea on how the next image is to occupy space next to base image. Then, inverse homography identifies potential points in the next image. A check is performed to identify pixel points that are in intersection with the base image and the next image. At this intersection, I have used a alpha value of 0.3 for alpha blending. In other words, it will take 0.3 from the next image and 0.7 from the base image while modifying the pixel value. Then, for the pixel points that do not intersect with the base image, I choose these points directly from the next image. This algorithm is implemented in the **concatenate** method.

A word on blending – I can improve my blending by using a distance transform and finding the pixel is closer to the border getting more weight. I have implemented this function in **distance2Border.** However, I am facing some artifacts in blending with this approach. I am working on fixing this.