MyAutoPano
RBE 549: P1 Phase 1

Pau Alcolea
Department of Robotics
Worcester Polytechnic Institute
palcolea@wpi.edu

Ben Orr
Department of Robotics
Worcester Polytechnic Institute
bcorr@wpi.edu

***Abstract*—The first phase of this project consists of developing a panorama image algorithm with traditional computer vision and image processing methods that is capable of combining multiple sets of images into a single cohesive picture. Different features are extracted from the separate images, which are matched and then cleaned with RANSAC. The homography between two images is estimated through the different feature matches, and it is used to finally warp the images together. This combination of steps can be used over sets of many images.**

I. Problem Statement

The goal of this project is to stitch two or more images in order to create one seamless panorama image. Different sets of overlapping images with different perspectives can be combined using traditional and deep learning approaches. For Phase 1 of this project, the focus is on the traditional approach and some methods to get corners and features out of an image and how to use those to then stitch the pictures the right way.

II. Phase 1: Traditional Approach

The traditional pipeline consists of detecting the different corners in an image, using Adaptive Non-Maximal Suppression to equally distribute the corners based on the other corners around them and the distance to the next strongest corner. From the corners, features are extracted, which are used for matching with other images. Out of all of the matches, the best ones are picked by using RANSAC, which brings us to the estimation of the homography that is the transformation matrix that can map an image onto another.

A. Data Collection

In addition to the available training datasets, we collected two custom datasets of six and five images, as seen in Figure 1.

B. Corner Detection

Corners were detected using the `cv2.cornerHarris()` method with parameters $blockSize = 2$, $ksize = 3$, and $k = 0.06$ [1]. A corner is seen as a region of an image with a lot of variation in all directions, which are detected with the Harris Corner Detector by finding that difference in intensity over a displacement using a rectangular window [1]. See Figures 2 to 6 for corner detection results on the provided and custom training sets. For test set results, see Figures 7 to 9. Corner outputs for test set 4 are not shown due to overlap with images in test sets 2 and 3.

Fig. 2. Corner detection output for training set 1

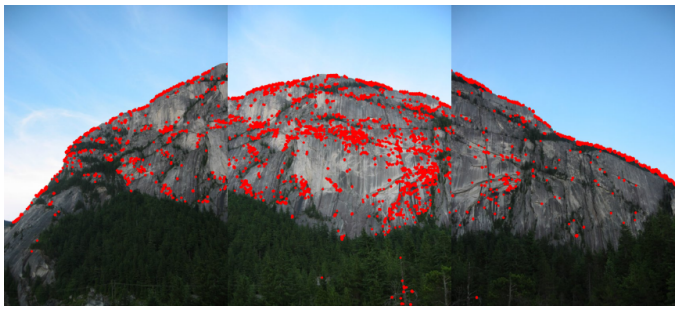

Fig. 3. Corner detection output for training set 2

Fig. 4. Corner detection output for training set 3

Fig. 1. Custom datasets collected. Left: dataset #1, Right: dataset #2

Fig. 5. Corner detection output for custom dataset 1

Fig. 6. Corner detection output for custom dataset 2

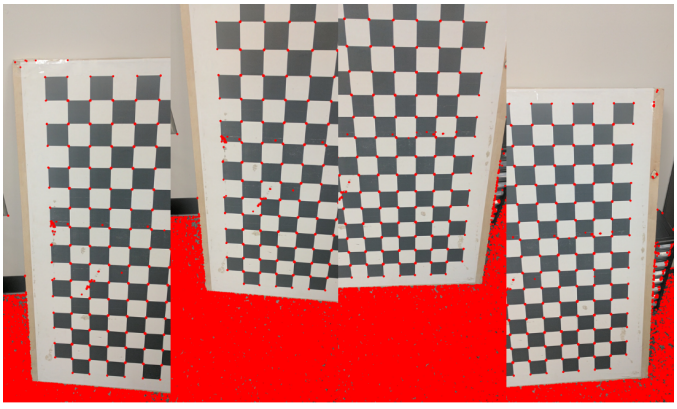

Fig. 7. Corner detection output for test set 1

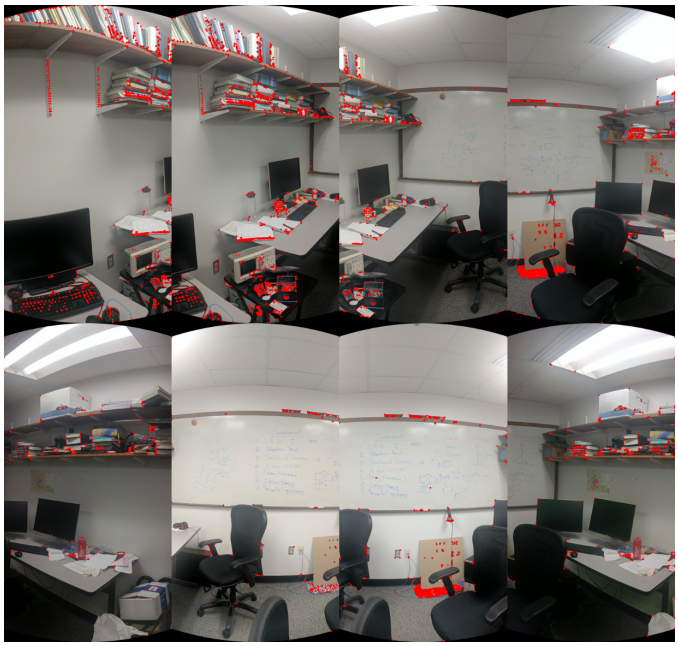

Fig. 8. Corner detection output for test set 2

Fig. 9. Corner detection output for test set 3

C. Adaptive Non-Maximal Suppression (ANMS)

We used Adaptive Non-Maximal Suppression (ANMS) to filter the corner locations as obtained in the previous section. ANMS selects a subset of N_{best} strong and spatially

distributed corners by finding the distance from each corner to the nearest stronger corner. See Algorithm 1 for a closer look at our implementation. We used $N_{\text{best}} = 1000$ and `skimage.feature.corner_peaks()` with $min\_distance = 5$ for detection of local maxima [2].

Algorithm 1 Adaptive Non-Maximal Suppression

Require: Corner score image C_{img}
Ensure: Set of local maxima corner locations $\mathcal{C}_{\text{anms}}$

Find local maxima in C_{img} to get candidate corners $\mathcal{P}$
Extract corner strengths $s_i = C_{img}(p_i)$ for each $p_i \in \mathcal{P}$
Sort $\mathcal{P}$ in descending order of corner strength s_i
Initialize suppression radii $r_i \leftarrow \infty$ for all i
for $i = 2$ to N **do**
    Let $p_i = (x_i, y_i)$
    for $j = 1$ to $i - 1$ **do**
        Let $p_j = (x_j, y_j)$
        Compute distance from p_i to stronger corner p_j:

$$d_{ij} = (x_i - x_j)^2 + (y_i - y_j)^2$$

        Update suppression radius:

$$r_i \leftarrow \min(r_i, d_{ij})$$

    end for
end for
Let $\mathcal{C}_{\text{anms}} \leftarrow N_{\text{best}}$ items in $\mathcal{P}$ with largest radii r_i
return $\mathcal{C}_{\text{anms}}$

Refer to Figures 10 to 14 for ANMS outputs on the training and custom datasets. For test set results, see Figures 15 to 17. ANMS outputs for test set 4 are not shown due to overlap with images in test sets 2 and 3. As you can see, the distribution of points is much more uniform than previously (when it was just the corners), as only the corners with the strongest 1000 "corner strength" that were to an extent isolated from other strong corners were kept.

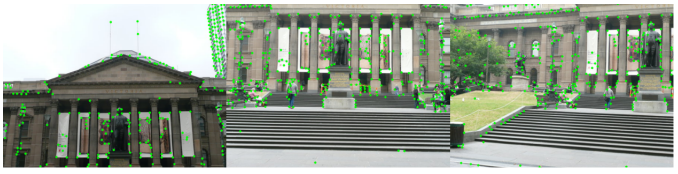

Fig. 10. ANMS output for training set 1

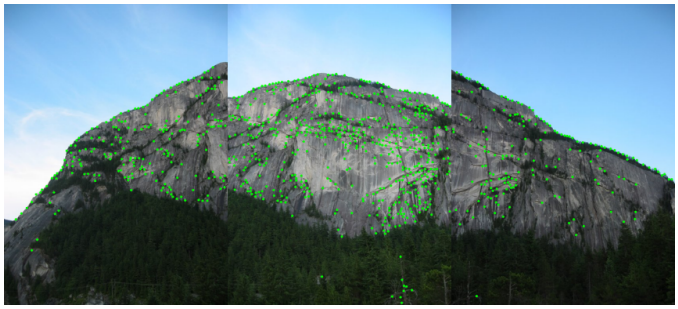

Fig. 11. ANMS output for training set 2

Fig. 12. ANMS output for training set 3

Fig. 13. ANMS output for custom dataset 1

Fig. 14. ANMS output for custom dataset 2

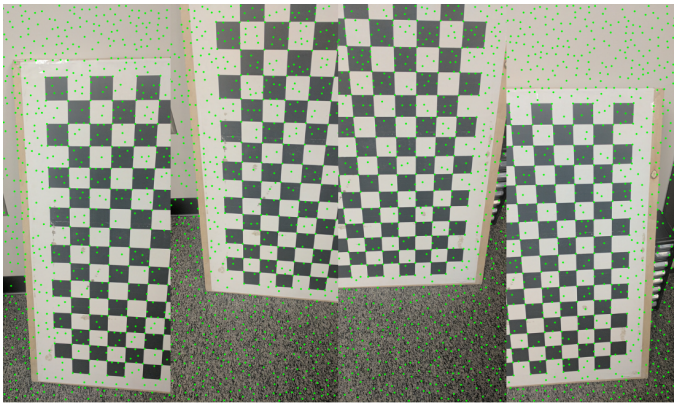

Fig. 15. ANMS output for test set 1

Fig. 16. ANMS output for test set 2

Fig. 17. ANMS output for test set 3

D. Feature Descriptor

The corners outputted by the ANMS algorithm are consider to be relevant points of interests, which have to be encoded in some way for the comparison between images. We encoded them into features using a feature descriptors, see Algorithm 2 for more information. The gist of this algorithm is that it looks at the pixels around a corner by extracting a patch centered at it. It blurs said patch, down-samples it and flattens it so that it is not so sensitive to the slight picture differences when matching features across images and complex.

Algorithm 2 Feature Encoding

Require: Input image I, Corner locations $\mathcal{C} = \{(x_i, y_i)\}$
Ensure: Feature descriptors $\mathbf{D}$, Valid corner locations $\mathcal{C}_{\text{valid}}$

Initialize empty descriptor list $\mathbf{D} \leftarrow [\,]$
Initialize empty corner list $\mathcal{C}_{\text{valid}} \leftarrow [\,]$
Convert I to grayscale image G
for each corner $(x, y) \in \mathcal{C}$ **do**
    if $(x - r, y - r)$ or $(x + r, y + r)$ invalid **then**
        Discard $\mathcal{C}_i$
        continue
    end if
    Extract patch $P \in \mathbb{R}^{41\times41}$ centered at (x, y)
    Apply Gaussian blur to P with $\sigma = 2$
    Downsample P to $P \in \mathbb{R}^{8\times8}$ using area interpolation
    Normalize P
    Flatten P into a vector descriptor $\mathbf{d} \in \mathbb{R}^{64}$
    Append $\mathbf{d}$ to $\mathbf{D}$
    Append (x, y) to $\mathcal{C}_{\text{valid}}$
end for
    return Descriptor matrix $\mathbf{D}$, Corner locations $\mathcal{C}_{\text{valid}}$

The following figures show the visualization of the encoded feature descriptors for the training and custom datasets: Figures 18 to 22. The relatively horizontal and near-zero nature of these curves means that no single pixel dominates the descriptor. The similarities between the mean curves between images can also somewhat represent the similarities between the images. A more similar descriptor means that there are

more features in common in the images in question. Figures 23 to 26 contain the results for all test datasets.

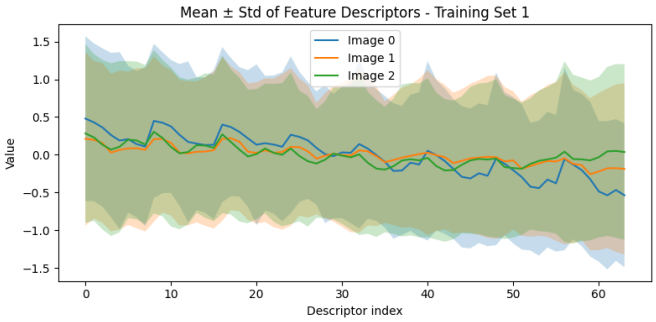

Fig. 18. Mean, standard deviation of feature descriptors for training set 1

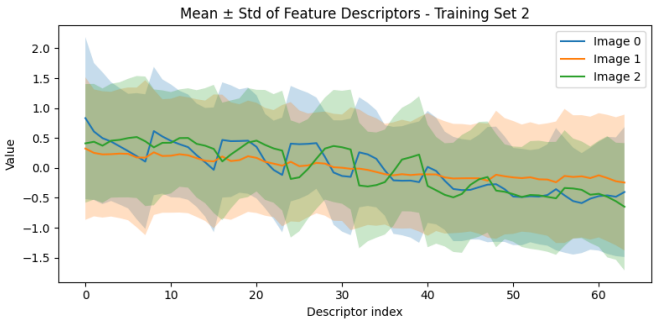

Fig. 19. Mean, standard deviation of feature descriptors for training set 2

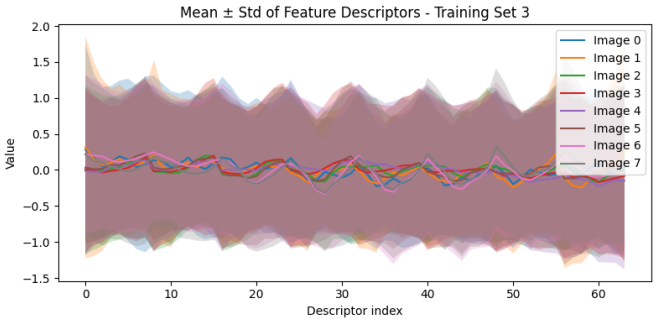

Fig. 20. Mean, standard deviation of feature descriptors for training set 3

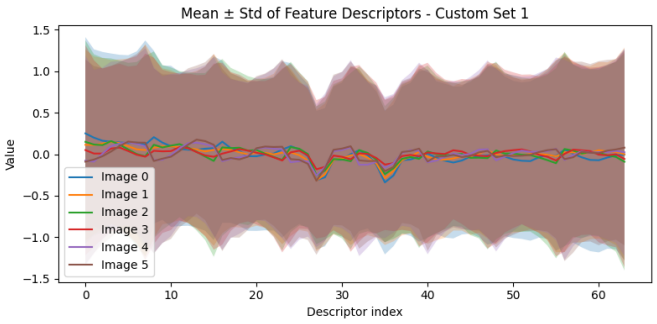

Fig. 21. Mean, standard deviation of feature descriptors for custom dataset 1

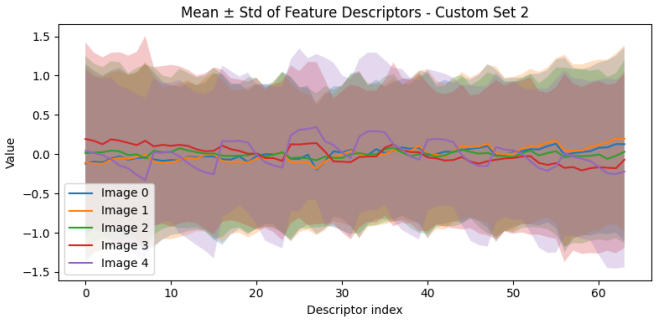

Fig. 22. Mean, standard deviation of feature descriptors for custom dataset 2

Fig. 23. Mean, standard deviation of feature descriptors for test set 1

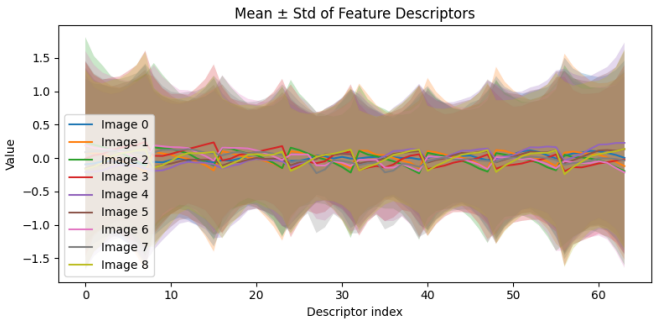

Fig. 24. Mean, standard deviation of feature descriptors for test set 2

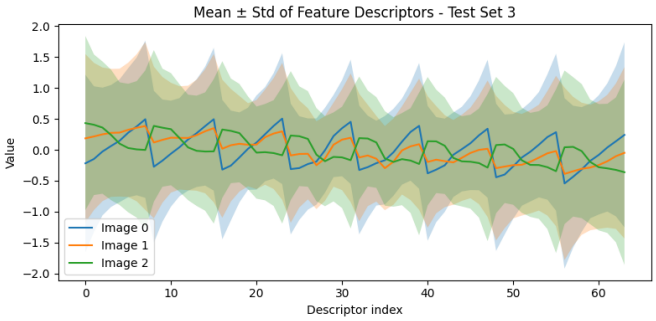

Fig. 25. Mean, standard deviation of feature descriptors for test set 3

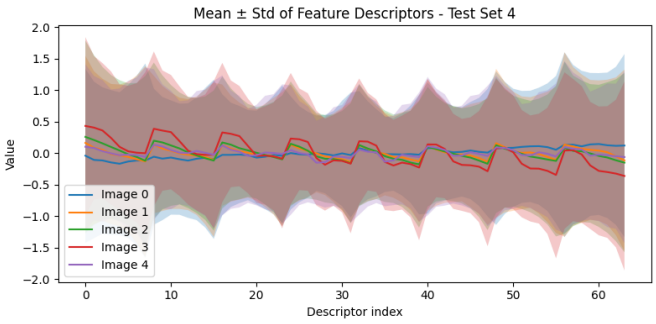

Fig. 26. Mean, standard deviation of feature descriptors for test set 4

E. Feature Matching

After obtaining a set of valid feature descriptors for each image, feature points had to be aligned for each pair of images i and $i+1$. To accomplish this, we implemented Algorithm 3 with ratio threshold $\tau = 0.7$. This algorithm takes a feature descriptor and compares it to the all of the descriptors in the other image in question. The Sum of the Square Distances is taken for all these pairs of descriptors and the best two fits are recorded. The ratio of the best to the second best distance is relevant because we want to confidently match the features. We aim for a small ratio, as that means that the best fit is much better than the second and stands out. A very high ratio could mean that there are a lot of similar features in the compared image, which doesn't provide a lot of confidence for the match. Figures 27 to 34 show the matches, represented by the lines that point from a corner of an image to its match in the next. The vast majority of these matches are correct, but there are some that pair the wrong corners, which is countered by RANSAC in the next step.

Algorithm 3 Feature Matching

Require: Descriptor sets $\mathbf{D}_1 = \{\mathbf{d}_1^i\}$ and $\mathbf{D}_2 = \{\mathbf{d}_2^j\}$
Ensure: List of matched index pairs $\mathcal{M}$

Initialize empty match list $\mathcal{M} \leftarrow [\,]$
for each descriptor $\mathbf{d}_1^i \in \mathbf{D}_1$ **do**
  Compute SSD distances to all descriptors in $\mathbf{D}_2$:

$$\delta_j = \|\mathbf{d}_1^i - \mathbf{d}_2^j\|_2^2$$

  Find index of nearest neighbor:

$$j^* = \arg\min_j \delta_j$$

  Let $d_{\text{best}} = \delta_{j^*}$
  Find second-nearest distance d_{second}
  if $\dfrac{d_{\text{best}}}{d_{\text{second}}} < \tau$ **then**
    Append match (i, j^*) to $\mathcal{M}$
  end if
end for
  return $\mathcal{M}$

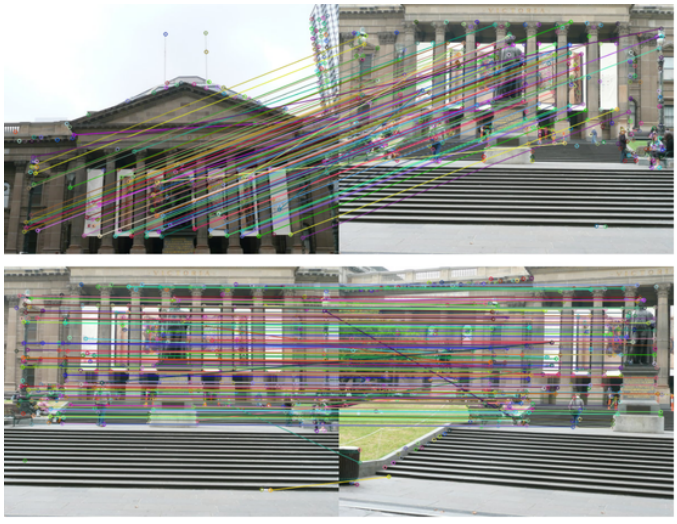

Fig. 27. Feature matching output for training set 1

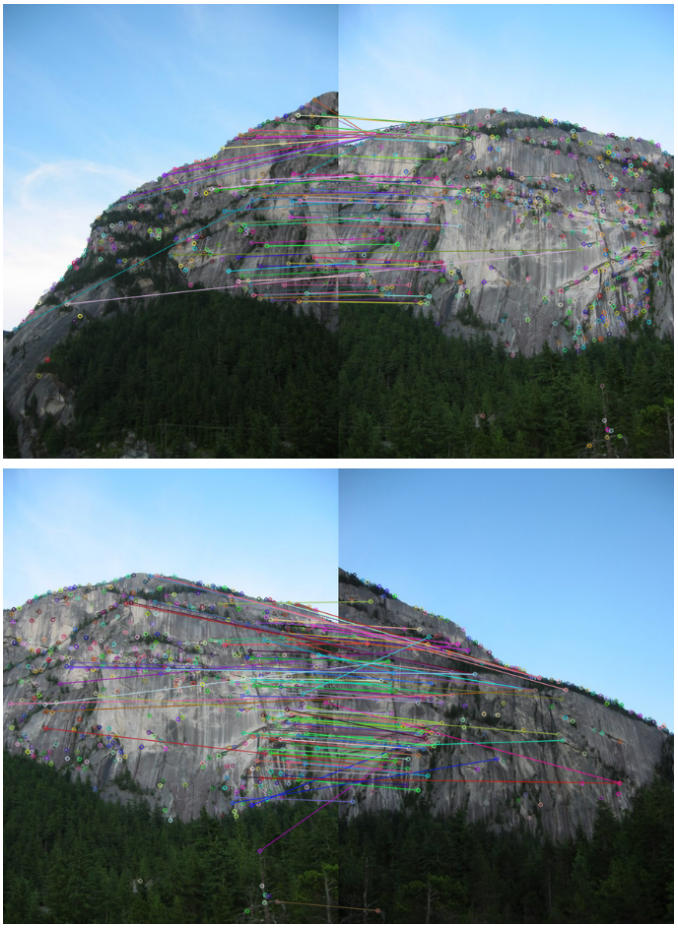

Fig. 28. Feature matching output for training set 2

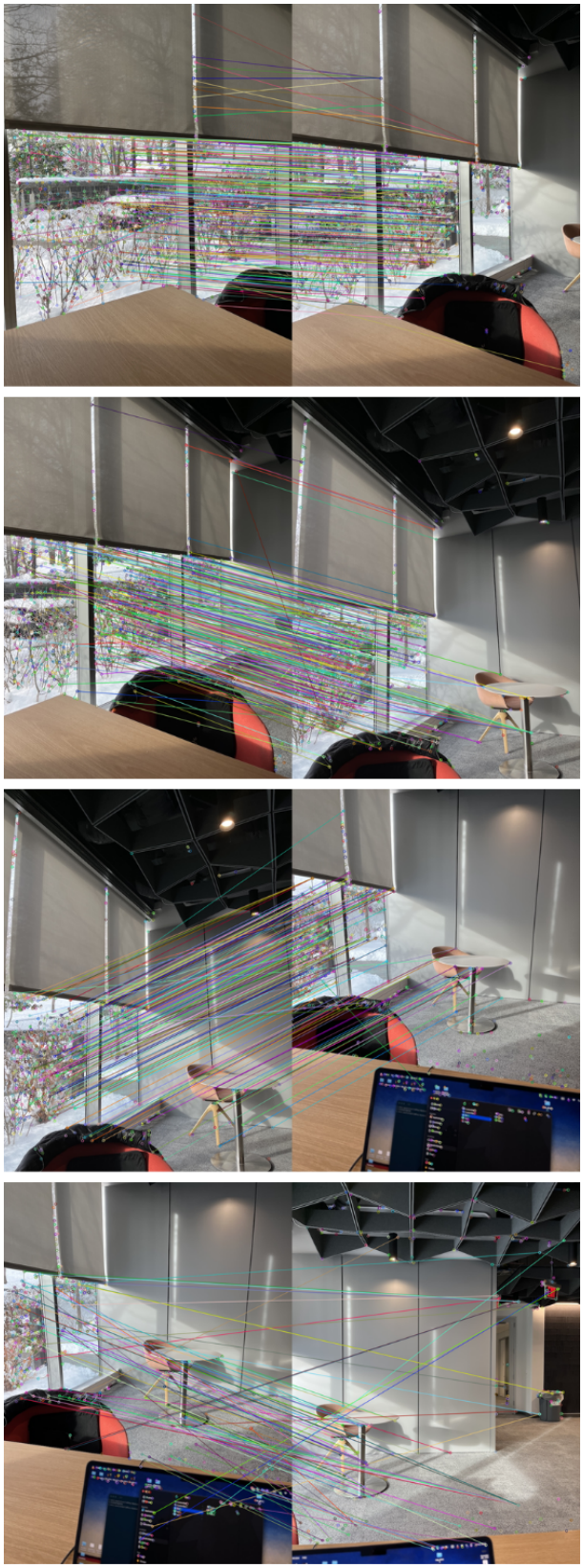

Fig. 31. Feature matching output for custom dataset 2

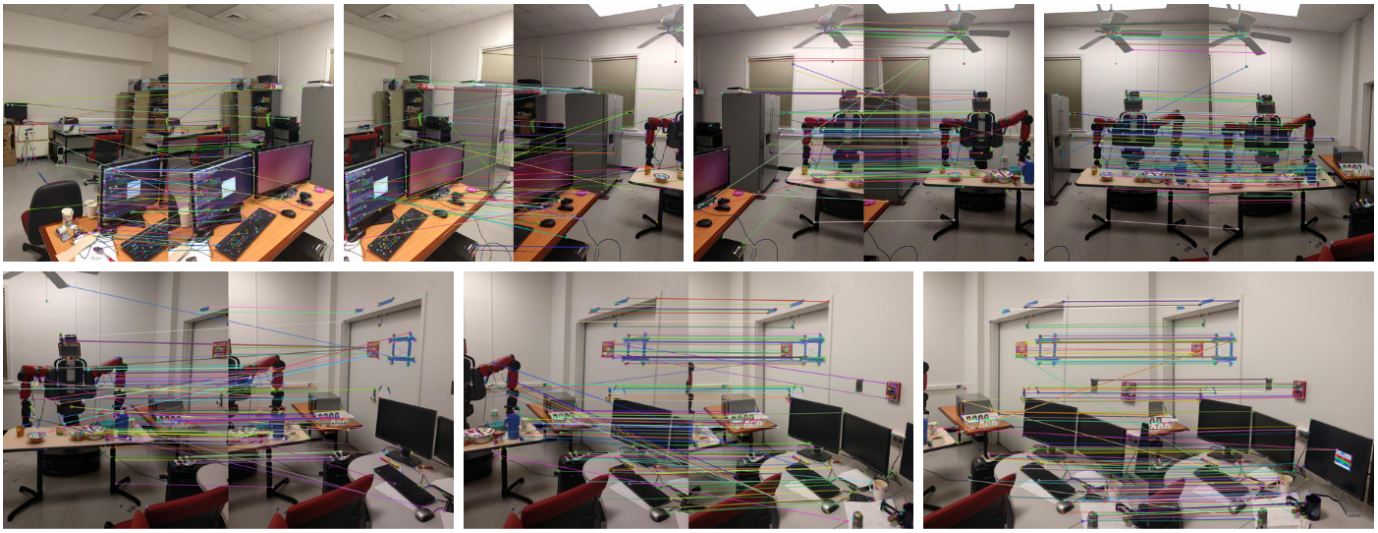

Fig. 29. Feature matching output for training set 3

Fig. 30. Feature matching output for custom dataset 1

Fig. 33. Feature matching output for test set 3

Fig. 32. Feature matching output for test set 2

F. RANSAC for Robust Homography

To produce homographies and resulting panoramas with higher confidence, we adopted the Random Sample Consensus (RANSAC) method. For up to $N = 1000$ iterations—or with enough (i.e. more than $\tau = 85\%$) pairs—we sampled 4 random pairs of matched features, solved for a candidate homography matrix H, and used it to estimate p_2 from p_1. Points used to calculate the maximum number of inliers (i.e. estimates below error threshold $\epsilon = 10$) were retained and used to compute a final H for the two images at hand. Algorithm 4 provides an overview of our implementation.

Algebraically speaking, the homography matrix H maps homogeneous coordinates $p_1 = \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$ to $p_2 = \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$ with Equation (1). Solving for x_2 and y_2 and rearranging gives the linear system of equations found in Equation (2), with the matrix form in Equations (3) and (4).

$$p_2 \sim Hp_1 \tag{1}$$

$$\begin{aligned} x_1h_{11} + y_1h_{12} + h_{13} - x_2x_1h_{31} - x_2y_1h_{32} - x_2h_{33} = 0 \\ x_1h_{21} + y_1h_{22} + h_{23} - y_2x_1h_{31} - y_2y_1h_{32} - y_2h_{33} = 0 \end{aligned} \tag{2}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_2x_1 & -x_2y_1 & -x_2 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_2x_1 & -y_2y_1 & -y_2 \end{bmatrix} h = 0 \tag{3}$$

$$h = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} & h_{33} \end{bmatrix}^T \tag{4}$$

This gives us $Ah = 0$ where $A \in \mathbb{R}^{2n \times 9}$ for $n \geq 4$. We selected $n = 4$ random pairs to satisfy the 8 degrees of freedom in H and continued to solve for h by using singular value decomposition (SVD) (see Equation (5).

$$A = U\Sigma V^T, \quad h = \underset{\|h\|=1}{\operatorname{argmin}} \|Ah\| = \mathrm{v}_9 \tag{5}$$

After up to $N = 1000$ iterations, image pairs with a ratio of inliers to total matches of less than 0.30 were rejected due to insufficient overlap. In our implementation of the algorithm, the homography for these two images was not saved, and the set of images was separated into two at this connection. For example, for Test Set 4, which contained three consecutive images of the lab room scene, one of the hallway and one of the checkerboard, only the homography between the first-second, and the second-third images (which all corresponded to the same scene and had appropriate overlap) were saved. The set was divided into three, separating each scene due to lack of overlap between the pictures. At the time to select a set of images for the panoramic stitching, the longest set was selected, which resulted in Figure 60 of the lab room shown in the 'Final Results' section below.

Examples of satisfactory pairs for all training and custom datasets can be seen in Figures 35 to 39. In these figures one can see the difference that the algorithm makes on the matches, specially when compared to the feature matching results after ANMS seen in Figures 27 to 34 and **??**. Most of the misaligned connections between features are no longer present, which is what makes the homography calculation more accurate for warping the images onto one another. Figures 40 to 42 are for test dataset results. Note that for test sets 2 and 4

Algorithm 4 RANSAC-Based Homography Estimation

Require: Set of point correspondences

$$\mathcal{P} = \{(\mathbf{p}_1^i, \mathbf{p}_2^i)\}, \quad \mathbf{p}_k^i = (x_k^i, y_k^i)$$

Ensure: Estimated homography H, inlier correspondence set $\mathcal{P}_{\text{inliers}}$

Initialize best inlier set $\mathcal{P}_{\text{best}} \leftarrow \emptyset$
for $k = 1$ to N **do**
  Randomly sample four correspondences from $\mathcal{P}$
  Construct linear system $Ah = 0$
  if rank$(A) < 8$ **then**
    Continue
  end if
  Solve for homography matrix H using SVD:

$$h = \arg\min_{\|h\|=1} \|Ah\|, \quad H = \text{vec}^{-1}(h) \in \mathbb{R}^{3\times 3}$$

  Initialize inlier set $\mathcal{P}_{\text{curr}} \leftarrow \emptyset$
  for each correspondence $(\mathbf{p}_1, \mathbf{p}_2) \in \mathcal{P}$ **do**
    Project point using homogeneous coordinates:

$$\hat{\mathbf{p}}_2 = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2^{\text{est}} = \frac{1}{\hat{p}_{2,z}} \begin{bmatrix} \hat{p}_{2,x} \\ \hat{p}_{2,y} \end{bmatrix}$$

    Compute reprojection error:

$$e = \|\mathbf{p}_2^{\text{est}} - \mathbf{p}_2\|_1$$

    if $e < \epsilon$ **then**
      Add correspondence to $\mathcal{P}_{\text{curr}}$
    end if
  end for
  if $|\mathcal{P}_{\text{curr}}| > |\mathcal{P}_{\text{best}}|$ **then**
    $\mathcal{P}_{\text{best}} \leftarrow \mathcal{P}_{\text{curr}}$
  end if
  if $|\mathcal{P}_{\text{curr}}| > \tau \cdot |\mathcal{P}|$ **then**
    Break
  end if
end for
Recompute H using all inliers in $\mathcal{P}_{\text{best}}$
return H, $\mathcal{P}_{\text{best}}$

(Figures 40 and 42), images included in matching results have been excluded due to low inlier count.

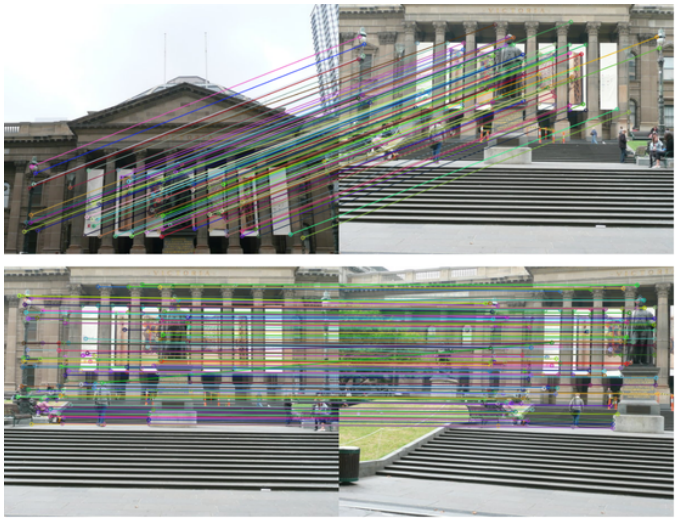

Fig. 35. Feature matching output after RANSAC for training set 1

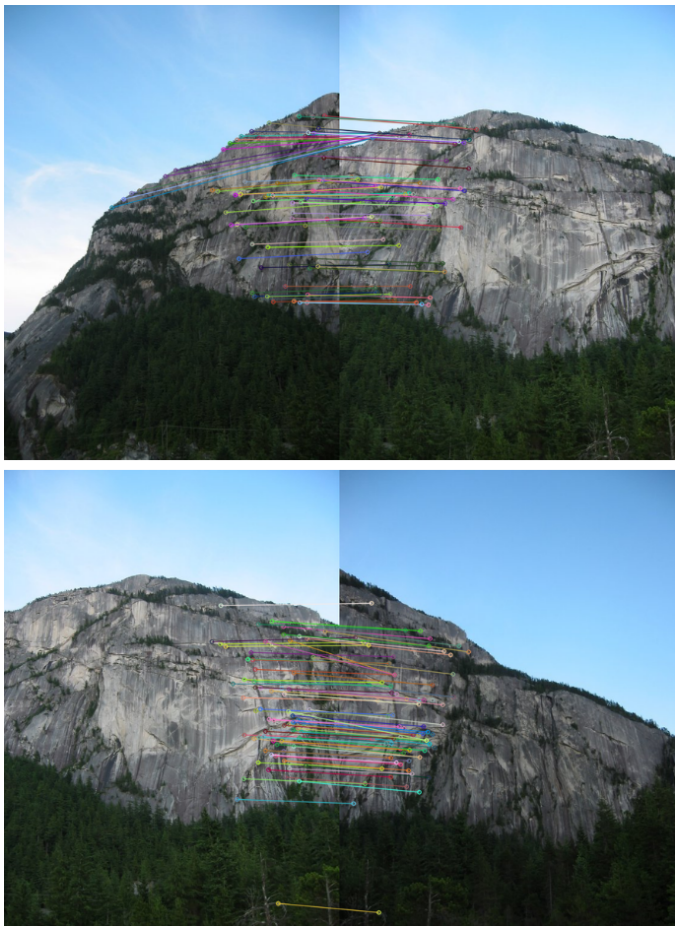

Fig. 36. Feature matching output after RANSAC for training set 2

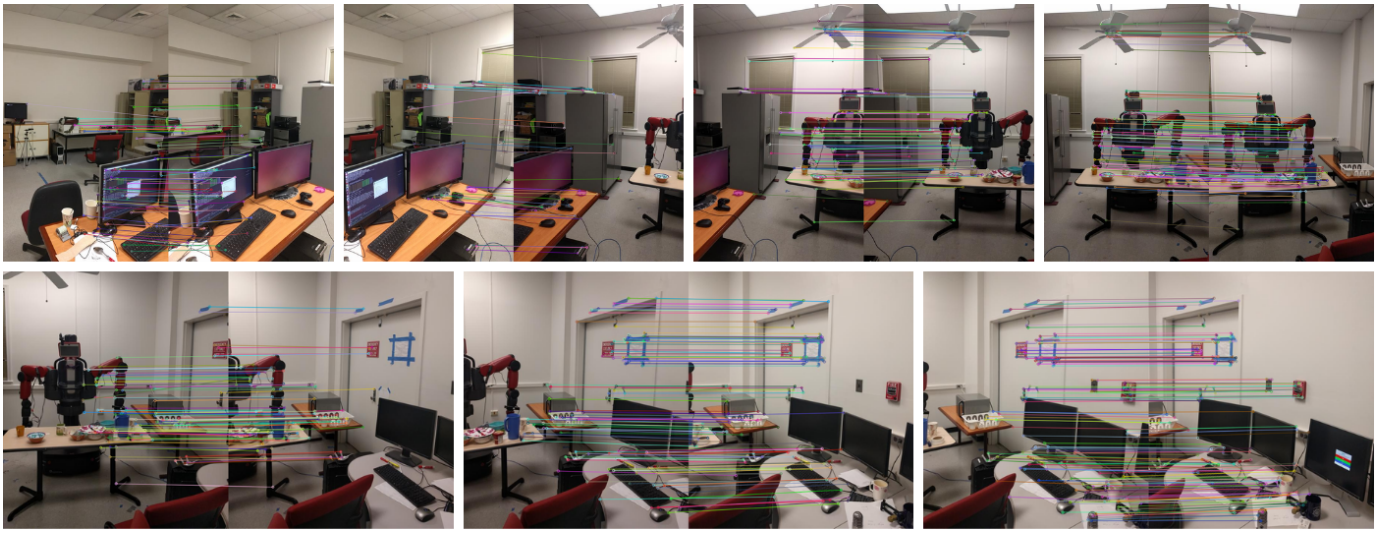

Fig. 37. Feature matching output after RANSAC for training set 3

Fig. 38. Feature matching output after RANSAC for custom dataset 1

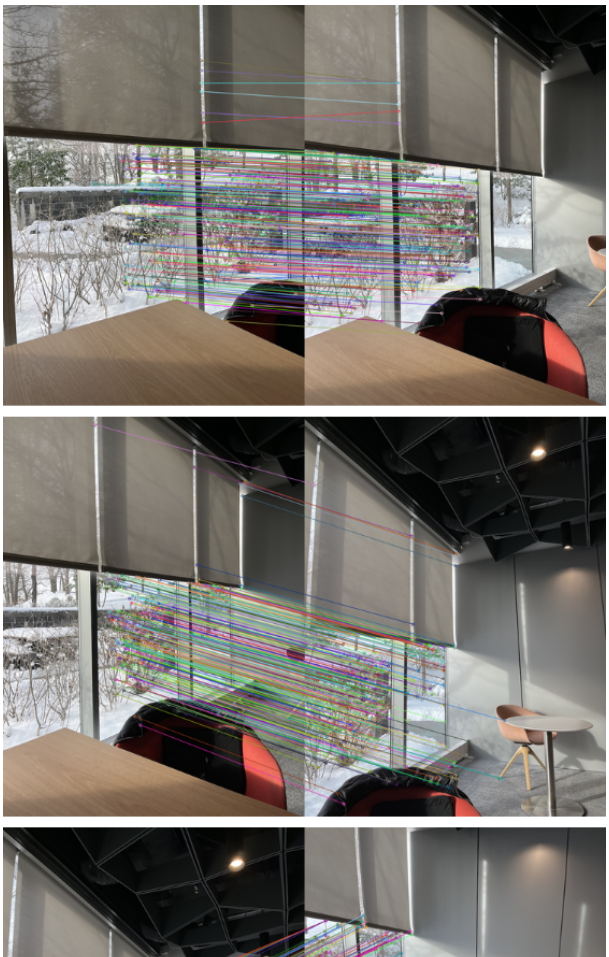

Fig. 41. Feature matching output after RANSAC for test set 3

Fig. 40. Feature matching output after RANSAC for test set 2

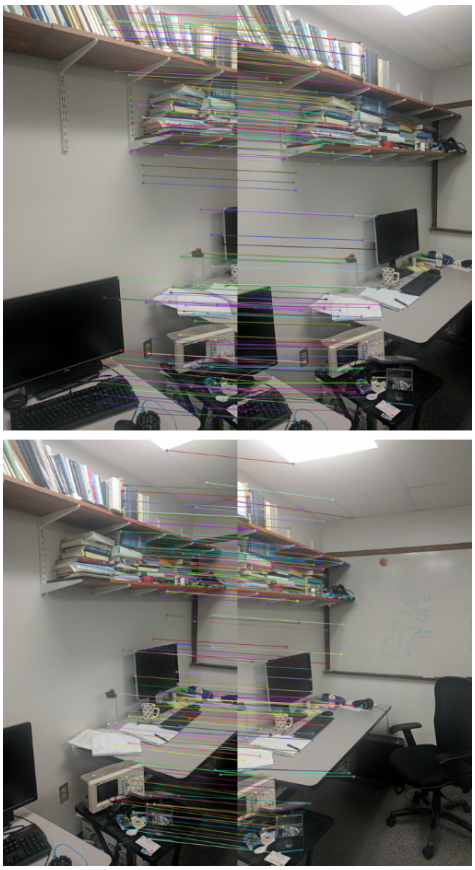

Fig. 42. Feature matching output after RANSAC for test set 4

G. Blending Images

To blend the images together, each image's matrix H was adjusted to be in reference to the panorama's reference frame, i.e. the "middle" image ($idx = n//2$) of the dataset. This was done via chaining H matrices and their inverses in their respective directions, giving H_to_ref. The dimensions of the panorama were derived via obtaining the `min()` and `max()` of the combined corner locations of every frame as transformed in the panorama frame (projected with H_to_ref using `cv2.perspectiveTransform()`. Each image was then individually warped to the panorama frame with `cv2.warpPerspective()` and Lanczos interpolation. Once correctly warped, the images needed to undergo smoothing to remove borders and other artifacts of stitching. To accomplish this, each pair of warped images was blended together using a weighted average, with weights computed using `cv2.distanceTransform()` (L2 distance; $maskSize = 5$), the result of which is a mask representing each pixel's distance to the nearest zero pixel, i.e. the edge of the image [1]. Figures 43 to 51 contain visualizations of each panorama's weighting of images for the training, custom, and test datasets.

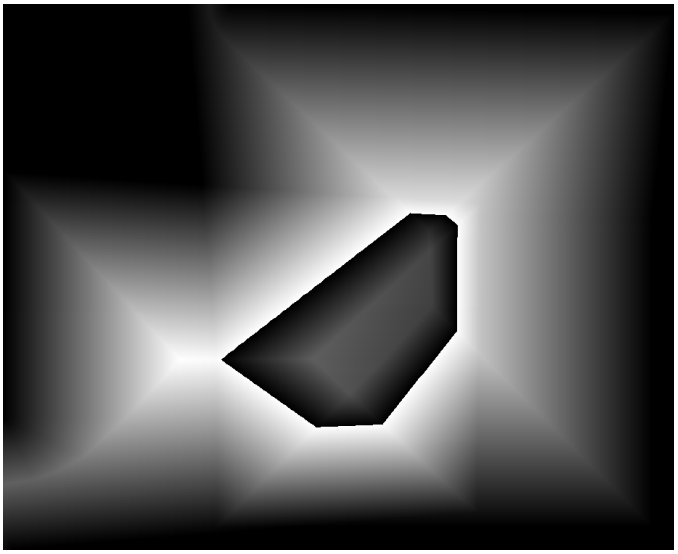

Fig. 43. Mask of weights used to blend images for training set 1

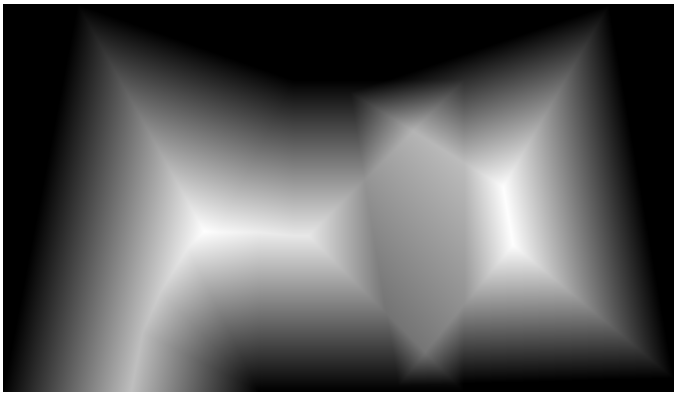

Fig. 44. Mask of weights used to blend images for training set 2

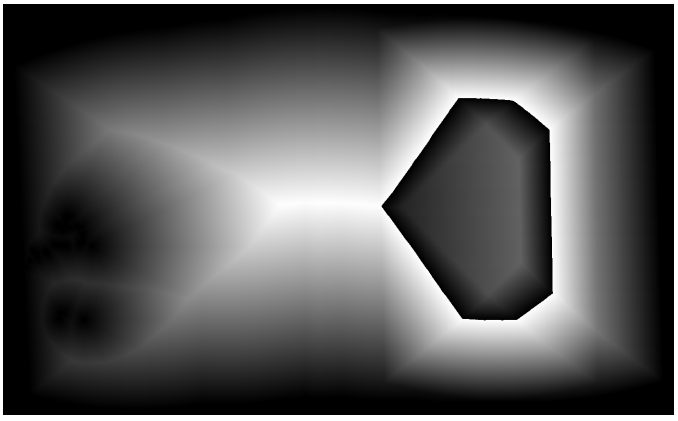

Fig. 45. Mask of weights used to blend images for training set 3

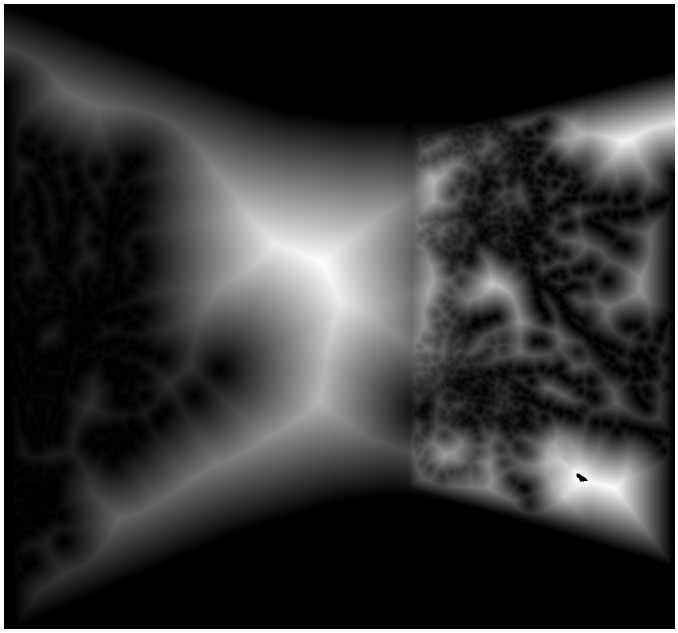

Fig. 46. Mask of weights used to blend images for custom dataset 1

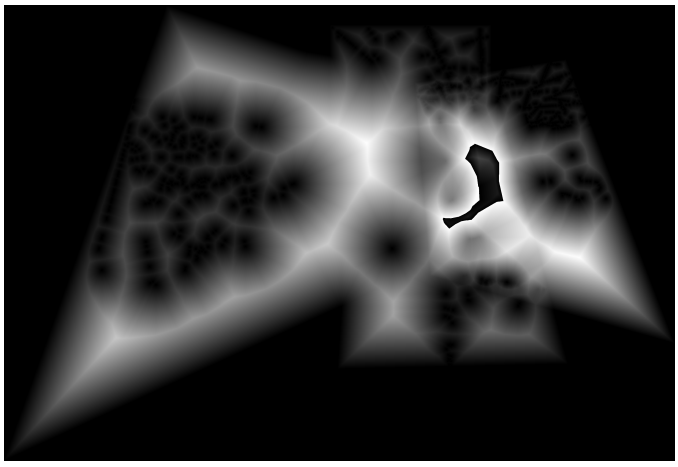

Fig. 47. Mask of weights used to blend images for custom dataset 2

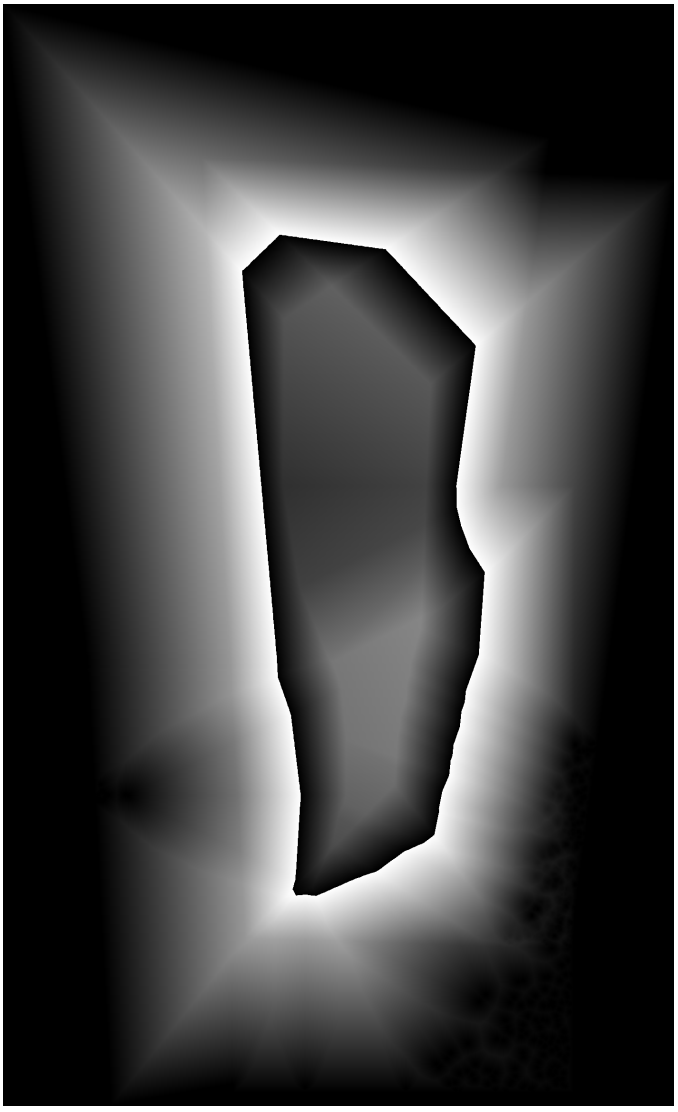

Fig. 48. Mask of weights used to blend images for test set 1

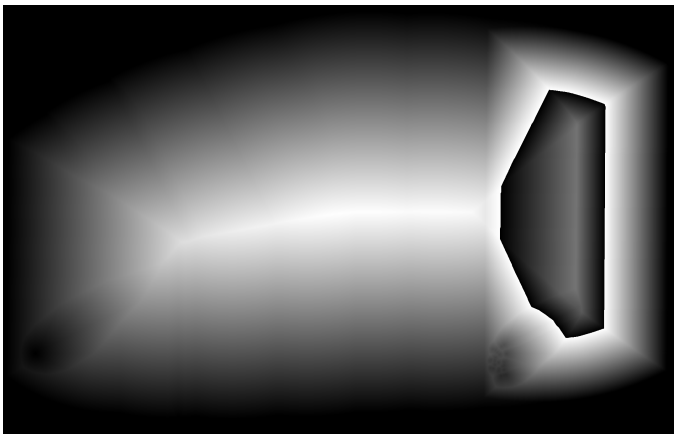

Fig. 49. Mask of weights used to blend images for test set 2

Fig. 50. Mask of weights used to blend images for test set 3

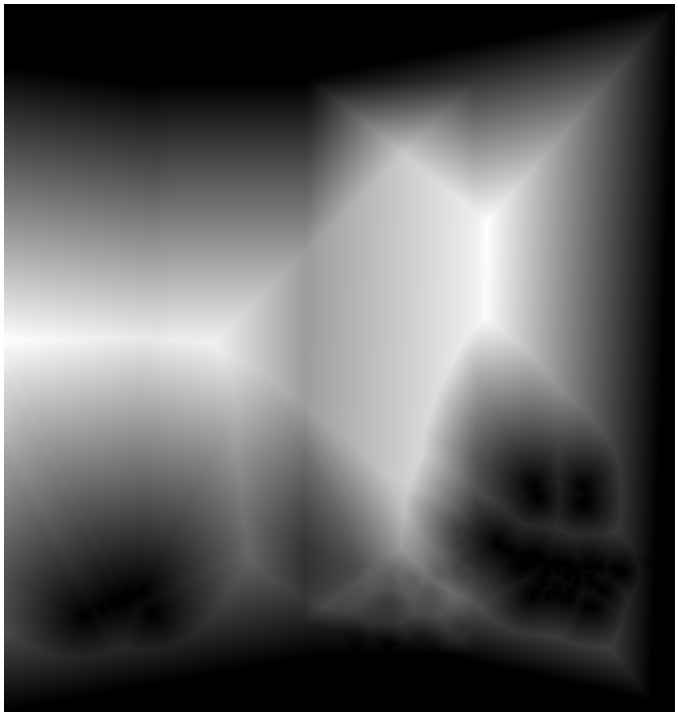

Fig. 51. Mask of weights used to blend images for test set 4

H. Final Results

Final panoramas for training and custom datasets are shown in Figures 52 to 56. The result for training set 3, (Figure 54), was too distorted, which is why we used a cylindrical projec-

tion on the input images. By projecting the original images onto a virtual cylinder, you minimize virtual distortion, the only issue being that this is assuming that the panorama was taken revolving around a circle (no vertical stretch), which in our case for training set 3 and test set 4 is a valid assumption. This is why these images turned out smooth with minimal distortion. The rest of the results for training set results and the custom set results were of very good quality. For some of the larger sets, such as Custom Dataset 2, the intersections between stitched images wasn't perfect, but that is due to the angle at which the images were taken and a lack of corners detected by our algorithm. In the future, this could be improved by more fine-tuning of the parameters to land with something that can identify more corners of interest in each photo. The first test set, resulting in fig. 57, showed a weakness in our algorithm, which struggled to differentiate the features in the images due to the repeating pattern and high resemblance in encoding. Solutions to this problem have been devised and explained in the 'Conclusion' section, but they weren't implemented due to time constraints. For Test Set 2, fig. 58, the panorama was challenging to assemble due to the fact that the images weren't linearly stored, forcing us to detect the order of the images and then rearrange them with the cylindrical projection. The panorama for Test Set 3, fig. 59 turned out beautifully, showcasing the benefits of stitching images in reference to the middle image, as it turned out very symmetrical and minimally distorted. Like briefly mentioned in section II-F, Figure 60 greatly shows the rejection of images with inadequate overlap, as the final panorama doesn't contain traces of the two last images in the set, where were a checkered board image from Test Set 1 and an image of the hallway from Test Set 3.

Fig. 52. Panorama output for training set 1

Fig. 53. Panorama output for training set 2

Fig. 54. Panorama output for training set 3. Due to the large FOV of this dataset, we initially saw distortions (top). These effects were mitigated after performing cylindrical projection on the input images (bottom).

Fig. 55. Panorama output for custom dataset 1

Fig. 56. Panorama output for custom dataset 2

Fig. 57. Panorama output for test set 1

Fig. 58. Panorama output for test set 2

Fig. 59. Panorama output for test set 3

Fig. 60. Panorama output for test set 4. Two of the five images were rejected due to lack of acceptable matches.

III. Conclusion

The algorithm we implemented was successful at seamlessly stitching different images. Slightly erroneous results occurred when the images contained repetitive patterns (such as Test Set 1), as it becomes extremely challenging for the feature descriptor to differentiate the features. This could be combated by modifying the parameters for this step, possibly encoding the features in larger vectors so as to retain more information and hopefully differentiate the features. Another option could be to look at all of the features in an image and remove those that are very similar to each other, which would leave the most unique/descriptive features for matching.

A way to elevate this project in future iterations would be to change our integration by representing the images and their connections as a graph as opposed to linearly with lists. This representation shows the true relationship between images, which is not linear. Interconnections in sets where the images don't follow a line (i.e. horizontal set like Train Set 3), become more intuitive and the pipeline becomes more robust for when an image is missing. The ability to better balance and see what image should be referenced based on the connectivity to other graphs. Finally this could open up the world of search algorithms and trees structure, which could be useful for chaining homographies across images or optimizing stitches.

IV. Contribution Statement

Pau Alcolea: Software, Conceptualization, Methodology, Writing **Ben Orr:** Conceptualization, Methodology, Software, Data Curation, Writing, Visualization

REFERENCES

[1] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[2] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: https://doi.org/10.7717/peerj.453