Homework 3 - Segmentation and Homography

Naomi Derel, 325324994 Sagi Ben Lulu, 207031493

08.01.2025

Question 1 - Image Segmentation

1.1 - Load Images

Figure 1: Images of Frogs

Figure 2: Images of Horses

1.2 - Classical & Deep Learning Segmentation

The classic method for segmentation we choose is watershed algorithm, and the deep learning-based method we choose is a pretrained model from torchvision deeplabv3-resnet101.

Classical Method - Watershed Algorithm

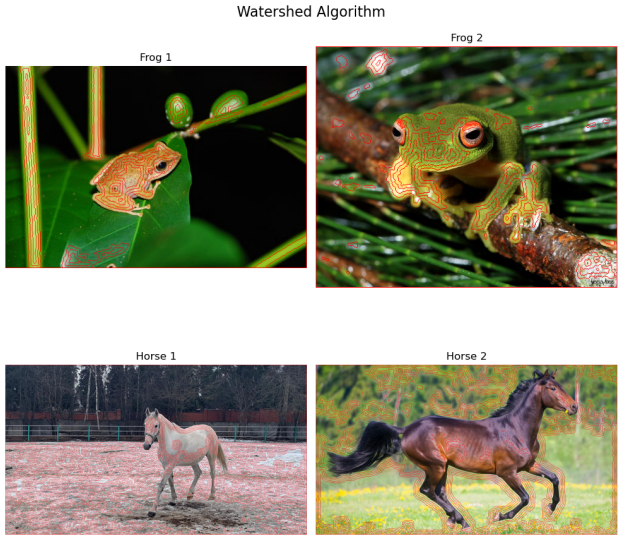

Figure 3: Segmentation Results Using the Watershed Algorithm

The watershed algorithm finds local minimas in the gradient image and then iteratively add neighboring pixels to the same area (segmentation) until reaches a pixel that belongs to a different area and this is the boundary between areas.

The results of the segmentation are in Figure 3.

Deep Learning Method - ResNet

deeplabv3-resnet101 uses a resnet101 network as a backbone for the segmentation to find lower dimension features, resnet101 is a deep convolutional network that uses skip connections. Then uses Atrous Spatial Pyramid Pooling to upsample the output for segmentation.

The results of the segmentation are in the following 4 Figures 4, 5, 6, 7.

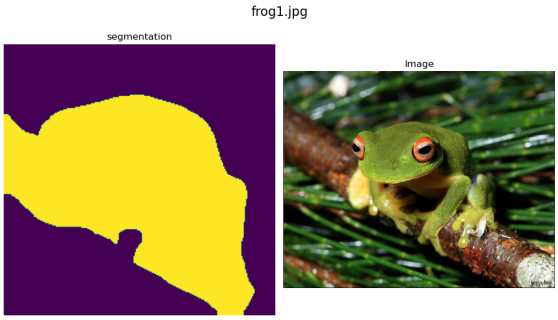

Figure 4: Segmentation of Frog 1

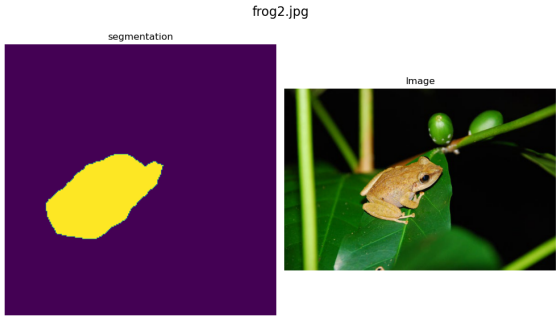

Figure 5: Segmentation of Frog 2

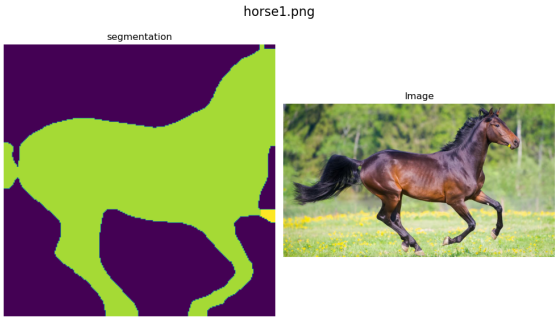

Figure 6: Segmentation of Horse 1

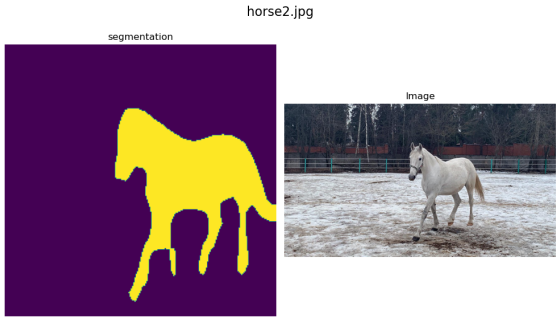

Figure 7: Segmentation of Horse 2

Advantages and Disadvantages

From the results, we can see that the deep learning method is far more accurate than the classical method and provides bounding boxes. However, for the orange frog, the deep learning method does not capture exact details that the classical method does. Additionally, the deep learning method is more computationally expensive and requires a pretrained model, while the classical method is fast and is possible to implement from scratch with limited resources.

1.3 - Additional Images

We select an image of a human being (`'lena.jpg'`), an image of a common object - a car, and an image of an uncommon object - a missile. The images are shown in Figures 8.

Figure 8: Additional Images

1.4 - Segmentation of Additional Images

The results of both methods of segmentation are shown in Figures 9 and 10.

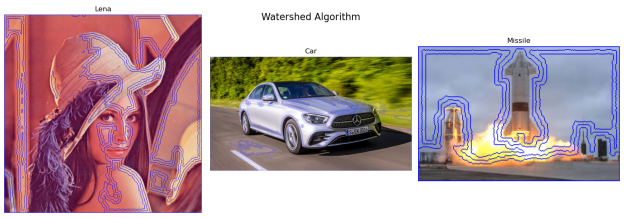

Figure 9: Segmentation Results Using the Watershed Algorithm

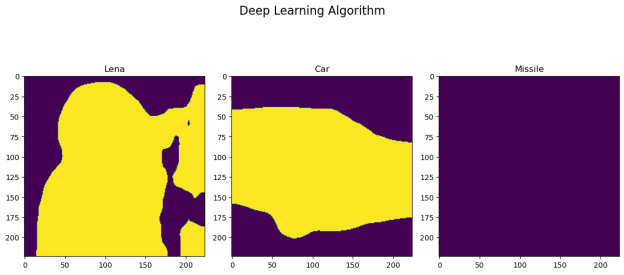

Figure 10: Segmentation Results Using the ResNet Model

The network method got a lot better results for the person and car but wasn't able to segment the missile, probably because it is not similar to the training data and therefore the model didn't learn to classify it. The Watershed algorithm got similar results for all the images and therefore the performance on the missile image is better than the network method got. We can also see that the algorithm performs better on the homogeneous parts and finds larger areas.

1.5 - Improvment Suggestions

We can try to apply a smoothing filter before the segmentation, this might help on regions like the horse's tail that are not fully covered but might also get worse results when for example trying to distinguish between objects. Another suggestion is to add an edge image channel, for example to apply a canny edge detector

to get another channel, this might help the model learn the boundaries better and get better results around the edges.

Question 3 - Planar Homographies

3.0 - SIFT

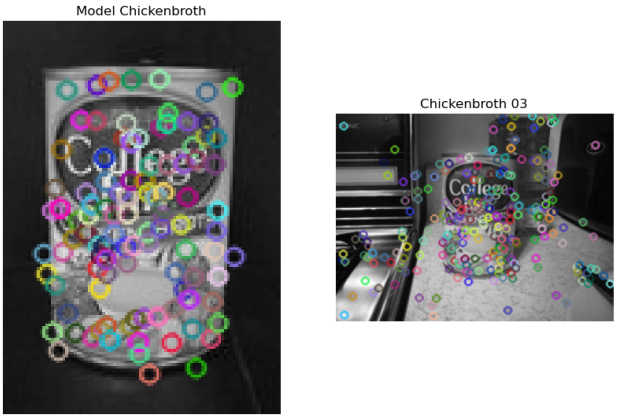

Figure 11: Descriptor Points on Chickenbroth Model and Counter Images

3.1 - Finding Corresponding Points Using SIFT

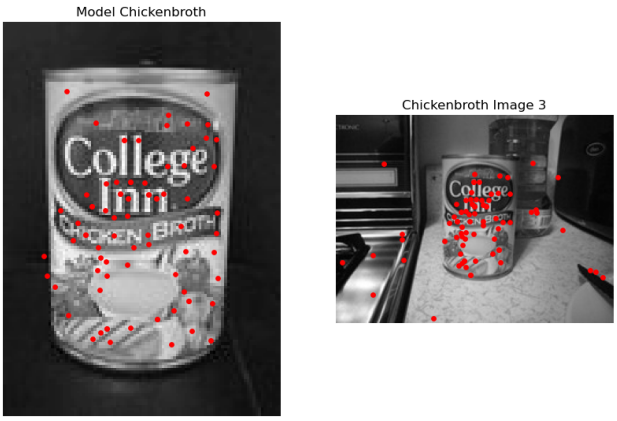

Figure 12: Corresponding Points on Chickenbroth Model and Counter Images

3.2 - Calculating Transformation

The implemented function to find the homography matrix performs the following steps:

1. From given points, constructs the A matrix as seen in lecture. For each point, two rows are defined:

$$A = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & x'x & x'y & x' \\ 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \end{bmatrix}$$

2. Calculate the SVD decomposition of A.
3. Find the minimal singular value and its corresponding column in V.
4. Reshape the column to a 3×3 matrix H and return it.

The result is projected using the matrix, returned to heterogeneous coordinates and displayed in the image. The result is shown in Figure 13.

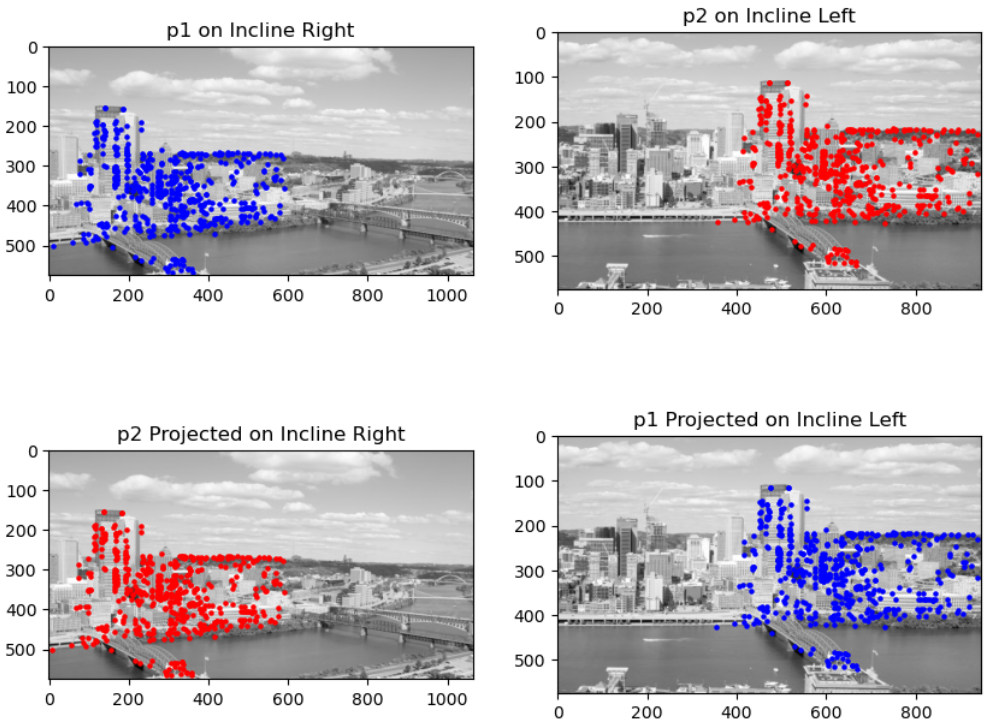

Figure 13: Homography Transformation of Incline Panorama Images

The projection appears to be relatively accurate, with many points matching between the original image (top row) and the relevant projected points (bottom row).

3.3 - Image Warping

We first implement a function that computes the necessary dimensions for the output image, which is done by computing the new coordinates of the corners of the image using the homography matrix. This function also updates the homography matrix to shift the image to the origin.

Then, the parameters can be inputted into the warp function, which uses inverted wrapping. The function is implemented using the suggested `RegularGridInterpolator`, which accepts a method of interpolation. We test the 'linear' method and the 'nearest' method, and the results are shown in Figures 14 and 15 respectively.

The results are very similar for both methods. The linear method interpolates between the pixels to create a smoother image, while the nearest method simply takes the nearest pixel value - however in this case, it is hard to notice the difference, likely due to the distance of the incline or quality of the images.

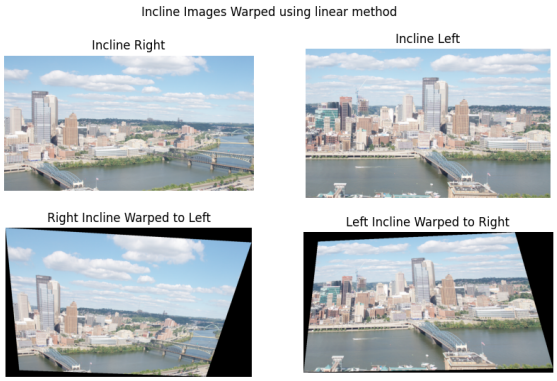

Figure 14: Linear Interpolation of Incline Panorama Images

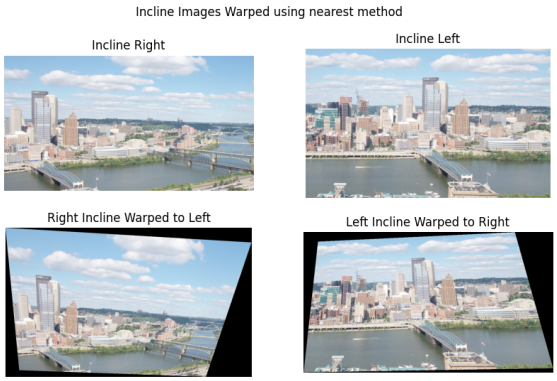

Figure 15: Nearest Interpolation of Incline Panorama Images

3.4 - Panorama Stitching

Following the instructions in the question, we assume the input images consist of two images, one of which is wrapped using the `warpPerspective` function. The images are then overlapped by padding and layering the images on top of each other. The result is shown in Figure 16.

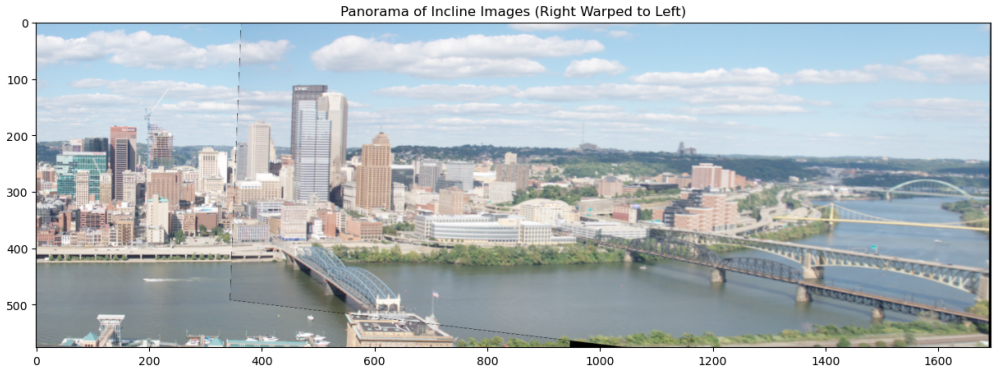

Figure 16: Panorama Stitching of Incline Panorama Images

3.5 - Several Image Stitching

First, we show the series of images and the resulting panramas on two domains: the Sintra castle and the beach photos.

We note that for both panoramas, since we are not using any selective point algorithm such as RANSAC, we must carefully utilize only some of the points recieved by the SIFT algorithms, specifically those with the highest confidence. This is because not all matches are correct, and using those to compute the homography matrix results in incomprehensible panoramas with "smeared" images. In practice, we used between 30-100 points to compute the panoramas well.

Sintra

The seperate images of the Sintra castle are shown in Figure 17. The resulting panorama is shown in Figure 18.

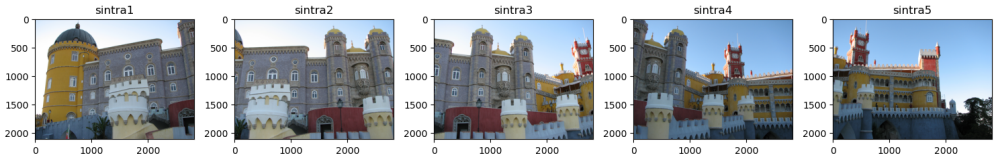

Figure 17: 5 Images of the Sintra Castle

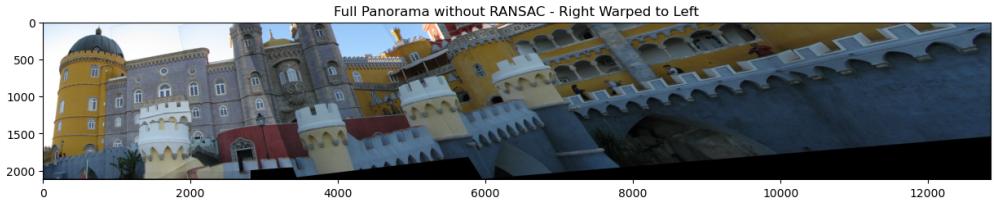

Figure 18: Panorama of the Sintra Castle

Beach

The seperate images of the beach are shown in Figure 19. The resulting panorama is shown in Figure 20.

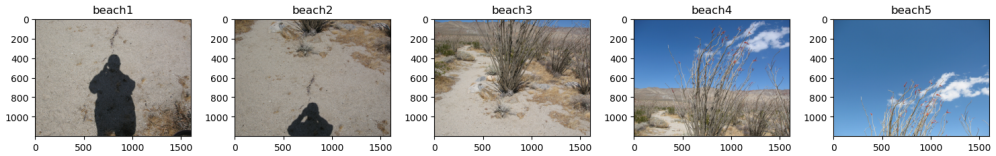

Figure 19: 5 Images of the Beach

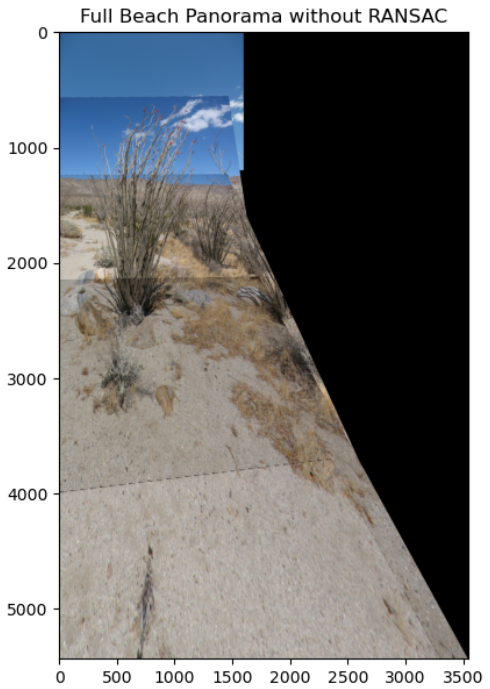

Figure 20: Panorama of the Beach

As the amount of images grow, the panorama becomes more streched out, the points are harder to match, and more information is lost. This is exagerated by each new image attaching to an already warped image which exagerates the angle and makes it harder to match, as the images are not strictly on a single plane and the depth difference interferes.

3.6 - RANSAC

RANSAC allows for a selection of only the best match points and computation of H according to them. This helps reduce the chances of wrong matches which result in an unclear panorama. Especially, it could be needed when objects in the images are on different planes, when there is noise in the images or when there are similar keypoints in the image and errors are more likely.

Figure 21: Panorama of Sintra with RANSAC

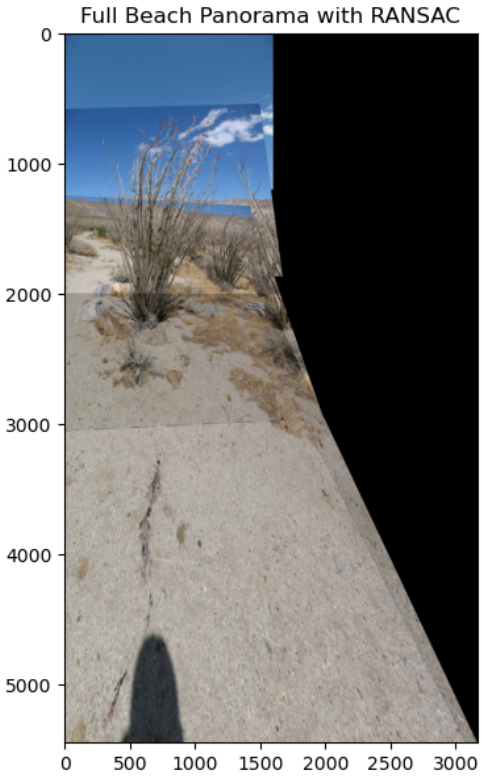

Figure 22: Panorama of Beach with RANSAC

We show the results of the panoramas using the RANSAC algorithm, instead of selecting the first matches with most confidence as done before, in Figures 21, 22.

Compared to not using RANSAC, the results on the Sintra images are similar, although there was no need for the careful selection of correct number of matches and so it is less prone to error. On the beach images,

the RANSAC algorithm with appropriate parameters allowed for more of the image space to show as the homography matrix became more accurate. This improves the results, and makes them more robust similarly to the Sintra panorama.

These results could be improved further by increasing the iteration number even more and fine tuning the parameters to reach the best results, as trial and error on the beach images shows this can lead to improvement. Further improvement might be achieved through trying different ordering of the panoramic construction to avoid as much loss as possible in the images - this can be seen with the number of matches going down with the number of images stacked on top of each other, and sometimes going out of balance.

3.7 - Creating Our Own Panorama

We took 3 pictures of the Zisapel building with only camera rotation differences. Then resized the images to a smaller size (for shorter computation) and created their combined panorama image.

The seperate images of the Zisapel building are shown in Figure 23. The resulting panorama, computed with RANSAC, is shown in Figure 24.

Figure 23: 3 Images of the Zisapel Building

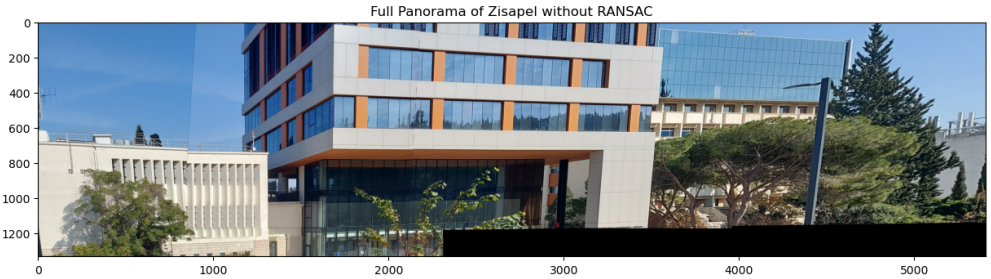

Figure 24: Panorama of the Zisapel Building

This is the best result we achieved, through using the RANSAC algorithm and as many as 2,000 iterations, which allowed for a near perfect alignment of the images, while using the matches without RANSAC didn't work at all and created an incomprehensible image.