# E27 Final Project:

# Panoramic Stitching from Videos

Theron Mansilla and Quinn Okabayashi

## Team Responsibilities

Theron implemented the entire UI for navigating our program, and Quinn implemented the code for converting video inputs into key frames useful for panoramic stitching. Both worked equally on the stitcher, downscaling program, and the write up.

## Important Files and Functionalities

**main.py**

- This file brings all the modules together, and is run without any command line args:

  ```
  $ python main.py
  ```

  This brings up a UI that the user can navigate to create panoramas from .mp4 files located in the videos/ folder. It allows them to choose how many key frames to pick, and whether to show the key frames, show the panorama, or save the panorama.

**stitcher.py**

- This file provides utilities for taking a sequence of images in a panoramic scene and stitching them into one big panorama. It uses ORB to automatically detect key points, and then uses our stitching and blending strategy from our project 4.

**video_slicer.py**

- This file provides utilities to automatically extract useful frames for panoramic stitching from a video file. It does this by converting a video into an array of all the frames, and taking the first frame as some offset from the start, and then iteratively finding the number of frames (the “interval”) that it needs to skip to make the keypoint matches align nicely by using ORB. Once it finds the interval, it uses this to find the rest of the images using this frame interval.

**downscale_video.py**

- This file provides utility to downscale .mp4 files to 800x600. This makes the file sizes smaller for the repository, and allows faster point matching and stitching.

**demo.py**

- This file provides the utilities for the lovely UI interface featured all throughout our project.

## Reflecting on our Goals

In our project proposal, we initially set out to achieve the following:

1. Stitch a set of images together that don't all share common points.
2. Automatically detect keypoints and features to find matching points between two images.
3. Provide the user with a friendly user interface that allows them to navigate through our project and easily reproduce results.

Our project accomplishes these goals, and we're particularly proud of the UI because you can build custom panoramas with different parameters all within the UI. We also went above and beyond our goals by making our program build panoramas from videos instead of just still images, which makes our results  far more customizable.

## Limits to Approach

The biggest constraint to our project is getting good data. We originally had multiple sets of still images as datasets, but we found that ORB had a lot of trouble finding key points for them. This motivated us to use videos instead, since we could pick and choose which images to use, but we found that videos that had significant foreground and background subjects do not produce good keypoints, especially in the foreground. This is evident in the Mountains dataset, where a lot of ghosting occurred in the foreground (see results below).

## Improvements and Additions

Given more time, we would like to remove the black lines present at the borders of where an image ends up on top of another image during warping. This is caused by cv2.warpPerspective, since it enlarges parts of the image, and these stretched parts get blurred with the black background. This blur around the edges appears dark but isn't fully black, making it detected as part of the image and not the black background during image blending, making it not masked away.

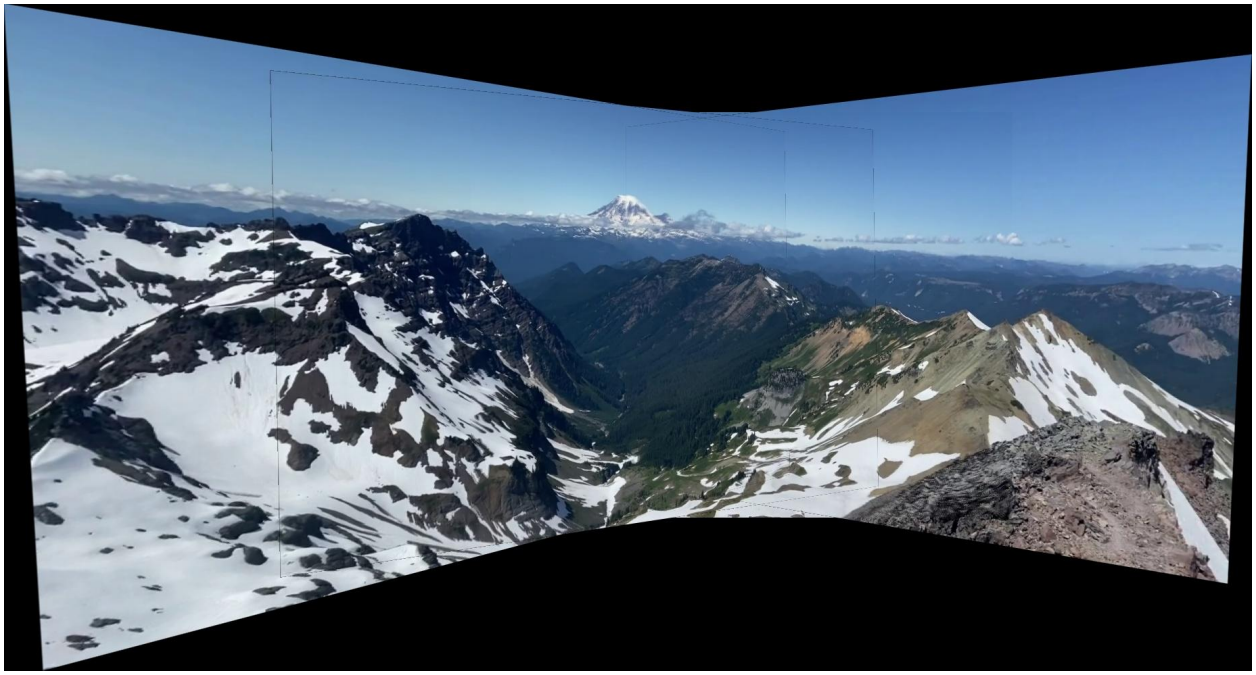
## Datasets Rankings

1. Beach [Using 10 frames]

The Beach dataset worked the best of all our datasets, as we were able to stitch together 10 frames to get a rounded look and still get a very clear image with minimal ghosting. The only issue is that the camera aperture changes throughout the video due to lighting differences, causing noticeable lighting changes near the edges. However, the fact that we were able to get no ghosting with so many frames placed this dataset very high on our rankings.

Overall Ranking: 90

2. Rainier [Using 4 frames]



The Rainier dataset creates an amazing panorama that only suffers from minor black lines caused by our blending algorithm. The colors are consistent and there isn't any noticeable ghosting. The only downside is that the video was very short, only allowing us to capture 4 usable frames.

Overall Ranking: 80

3. Italy [Using 10 frames]

Our Italy dataset worked pretty well, and we were able to stitch together 10 images. The coloring looks really great and the lighting is mostly consistent across the panorama, but there is significant ghosting in the trees. This isn't much of an issue, however, because trees are mostly blurry subjects and therefore it isn't as noticeable.

Overall Ranking: 75

4. Wharton [Using 5 frames]

The Wharton dataset stitched together extremely nicely, and really makes the viewer feel like they're looking up due to the angle. However, we found the upper bound is 5 frames, and any more will make the stitched panorama not visually appealing. Below is Wharton using 6 frames:

Overall Ranking: 50

5. Mountains [Using 3 frames]

Although there is minimal ghosting and consistent coloring in the upper half, the heavy ghosting featured in the foreground of the image weighs the overall rating of this dataset down. Additionally, running with a higher number of frames greatly increases the ghosting and distortion of the foreground. Below is Mountains using 6 frames:

Overall Ranking: 20

## Concluding Statements

Across our datasets, we found the best attributes to be consistent lighting, no foreground, and scenes with defined shapes. These give ORB the best chance of finding good matches, and allows our blending algorithm to make our output look like one continuous image without lighting issues near edges.

## Resources

Using ORB to find keypoints:

- https://medium.com/analytics-vidhya/panorama-formation-using-image-stitching-using-opencv-1068a0e8e47b

Panoramic stitching:

- https://towardsdatascience.com/image-panorama-stitching-with-opencv-2402bde6b46c

Writing video to file:

- https://stackoverflow.com/a/54731615/12401179

Using OpenCV and NumPy for UI:

- https://github.com/swatbotics/mnist_pca_knn/blob/main/mnist_pca_knn.py

Get video frames as NumPy array:

- https://stackoverflow.com/a/42166299/12401179

## Credits

All datasets were collected by Quinn Okabayashi

UI was inspired by Matt Zucker