Vrinda Subhash
Student #1009454626
Username: subhashv

(a) Comparison with Reference Implementation

When comparing the results of my implementation to the reference implementation, the most significant differences are in the accuracy and smoothness of the morphed images. My implementation generated outputs with visible distortions and jagged edges, particularly in areas on the letter "F" and in the background/side of the F. The images appear to be mirrored/inverted or “loopy”. In contrast, the reference implementation produced smoother and more cohesive results, keeping the original shape and details.

The main causes for these discrepancies likely are:

Interpolation Accuracy: My implementation struggled with achieving smooth warp transitions. This could be a result of incomplete or incorrect implementation of bilinear interpolation. It could also be caused due to problems in normalizing/un-normalizing points.

Lack of Vectorization: The reference solution uses optimized, vectorized operations to perform calculations more precisely and efficiently, which could’ve led to better output quality. I did not implement vectorization in my code, potentially leading to inefficiencies and less accurate computations.

Errors in helper functions: I could have incorrectly implemented the single_line_pair_algorithm. This error would be propagated through the rest of the functions, potentially causing the artifacts in the results of my implementation.

(b) Shortcomings and Artifacts

The most noticeable artifacts in my implementation include:

Holes and Gaps: In the "F" images generated by my code, several areas show clear gaps and distortions, and loop-like artifacts. There also seems to be a pattern of symmetry to the holes. These artifacts and gaps are likely due to inaccuracies in calculating the mappings between source and destination lines.

Warped Regions in Building Image: The building image displays a significant distortion, pulling the image towards the corners and edges. Since this is from experiment 1, where bilinear was in-enabled and supersampling was also off, these distortions could be from errors in line pair selection or incorrect handling of edge cases during warping; rather than from my bilinear or supersampling implementation.

(c) Quality Improvement with Bilinear Interpolation and Supersampling

Enabling bilinear interpolation and supersampling improves image quality by:

- **Bilinear Interpolation:** Adding bilinear interpolation results in smoother transitions and reduces pixelation. Without bilinear interpolation, the results are likely to have pixelation and jagged edges, as seen in the outputs of Experiment 1 and outputs of the test suite.
- **Supersampling:** By sampling multiple points within each pixel's footprint, supersampling helps reduce aliasing artifacts and improves the sharpness of the final image. Without it, the outputs tend to have more aliasing and lose finer details, especially in areas with lots of details.

In summary, enabling bilinear interpolation and supersampling creates clearer and artifact-free morphs by increasing the sharpness and accuracy of image transformations.

Extra Notes:

When I uploaded my tar file to MarkUs, my file was far too large so I had to remove/compress some files to meet the 100MB upload size:

- In my tar file I have removed the files in the “data” folder (CSC320/morphing/app/a2/data) unrelated to image f.
- I have also removed files in the “test_results” folder (CSC320/morphing/app/a2/test_results) as they have been added to my “results” folder.
- When doing experiment 4, more than 20 images were outputted. I manually went through the photos and removed what appeared to be duplicates until there were only 20 images (frames) left.
    - Those 20 images each had a large file size, so I adjusted them to have a much lower resolution to decrease the size.
- I removed some . files (.DS_Store, .gitignore, .readme_media)