Homework 3: Image Mosaicking

CS 566, Fall 2025

Total points: 13

Due on Thursday, Oct 23, 2025

Please follow the homework guidelines at `https://pages.cs.wisc.edu/~mohitg/courses/CS566/hw-guidelines.html` in your submission.

`runHw3.py` will be your main interface for running your code. Parameters for the different programs or unit tests can also be set in that file. Before submission, make sure that you can run all your programs with the command `python runHw3.py all` with no errors.

This assignment makes use of the following Python packages: `NumPy`, `Matplotlib`, `scikit-image`, `opencv`.

Vectorization

The Python library NumPy is optimized for operations involving matrices and vectors. Avoid using loops (e.g., for, while) with NumPy whenever possible – looping can result in long-running code. Instead, you should vectorize loops to optimize your code for performance. In many cases, vectorization also results in more compact code (fewer lines to write!).

If you are new to NumPy, refer to these articles:

- `https://www.geeksforgeeks.org/numpy/vectorized-operations-in-numpy/`
- `https://blog.paperspace.com/numpy-optimization-vectorization-and-broadcasting/`

Image Mosaicking App (13 points)

Your task is to develop an Image Mosaicking App that stitches a collection of photos into a "mosaic." Creating image mosaics and panoramas has become one of the most popular features on smartphones. Outside the realm of smartphones, similar applications have also been developed to create stunning panoramas seen on GigaPan. With the concepts and algorithms presented in the Image Alignment lecture, you too can create an image mosaic.

We will take one photo from the collection as the reference image. Given this reference, the steps are:

1. **Registration**: estimate the geometric transformation relating each image to the reference image.
2. **Warping**: undo the transform so all images are in the same frame of reference.
3. **Blending**: combine the warped images into a single panoramic view.

The registration method is prone to errors, so we apply RANSAC (RANdom Sampling And Consensus). Each step will be built as its own program. This homework has five parts: Challenge 1a through 1e.

Registration (already implemented)

The registration program calculates the homography between a pair of images.

- `computeHomography.py` estimates the homography:

$$H = \texttt{computeHomography(src\_pts, dest\_pts)}$$

- `applyHomography.py` applies a homography to a set of points:

```
dest_pts = applyHomography(H, test_pts)
```

- `showCorrespondence.py` visualizes mappings between corresponding points in two images.

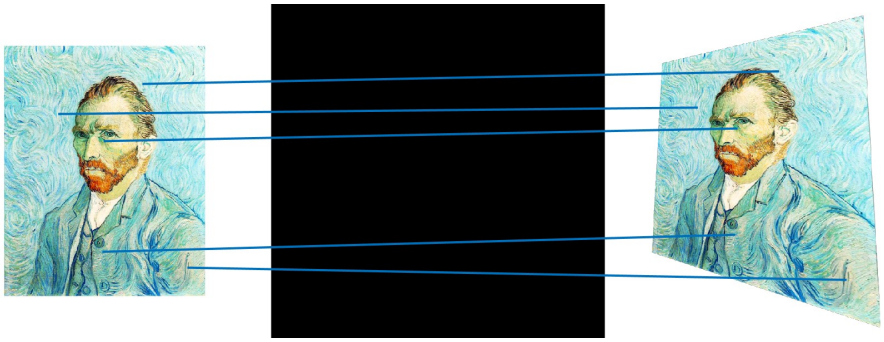

Figure 1: Example output of `showCorrespondence`.

Challenge 1a: Warping (4 points)

Warp an image and paste a portrait of Vincent (`portrait_small.png`) to the empty billboard in `osaka.png`.

1. Estimate the homography using `computeHomography`. (1 point)
2. Implement backward warping:

   $$[mask, dest\_img] = \texttt{backwardWarpImg(src\_img, dest\_to\_src\_H, dest\_canvas\_size)}$$

   where `dest_img` is the warped portrait, and `mask` indicates the region. (3 points)

Figure 2: Warping example: Before (left), After (right).

Challenge 1b: RANSAC (3 points)

Write `runRANSAC` to robustly compute homography:

$$[inliers\_id, src\_to\_dest\_H] = \texttt{runRANSAC(src\_pt, dest\_pt, ransac\_n, ransac\_eps)}$$

Use Euclidean distance for error. Recommended: `ransac_n = 100`, `ransac_eps = 3`.

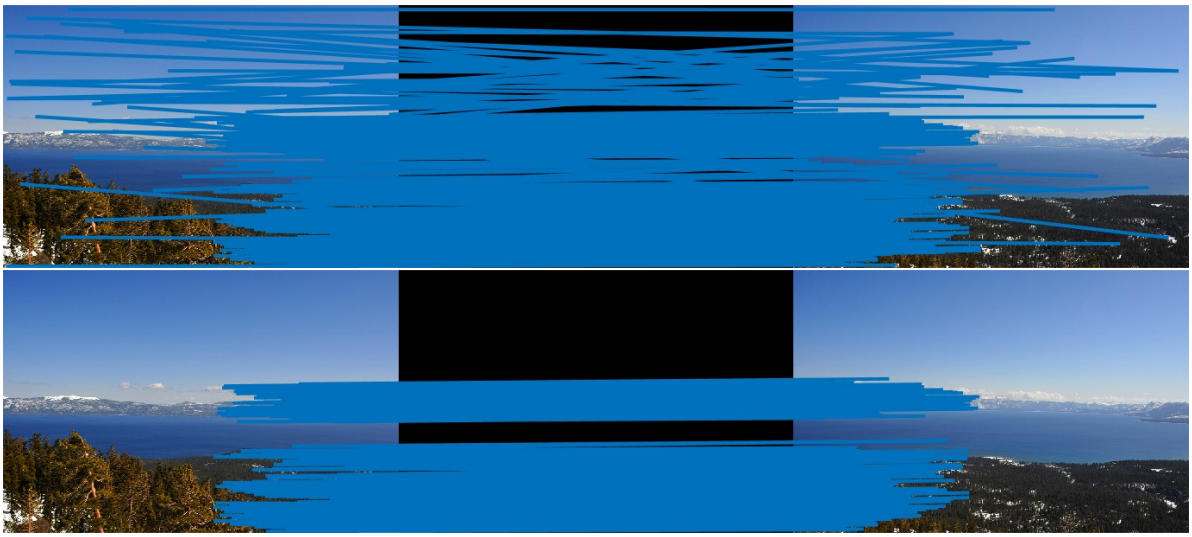

Figure 3: Keypoint matches before (above) and after (below) running RANSAC.

Challenge 1c: Blending (3 points)

Write `blendImagePair` to blend two images:

$$result = \texttt{blendImagePair(img1, mask1, img2, mask2, blending\_mode)}$$

Blending modes:

- **overlay**: copy `img2` over `img1`. (already implemented)
- **blend**: weighted blending using `bwdist` and normalized masks.

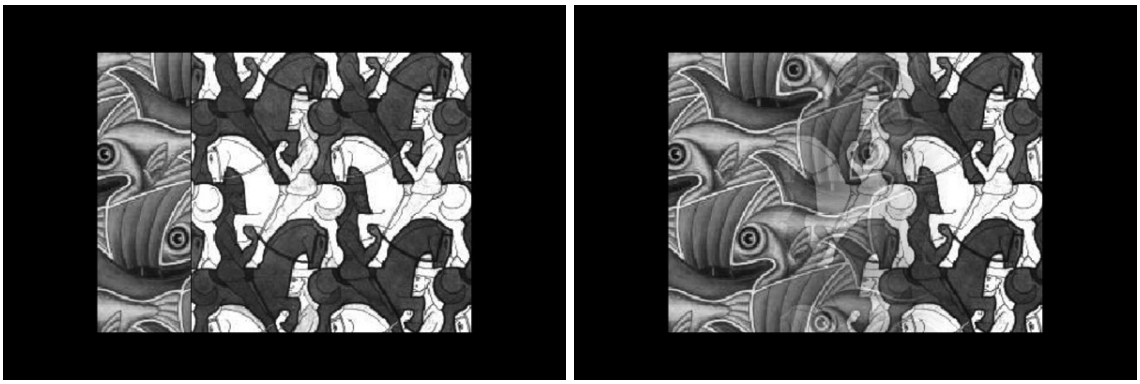

Figure 4: Blending examples: overlay (left), blend (right).

Challenge 1d: Stitching (2 points)

Write `stitchImg` to stitch an arbitrary number of images:

$$stitched\_img = \texttt{stitchImg(img1, img2, ..., imgN)}$$

Use "blend" mode from `blendImagePair`.

Challenge 1e: Your own photos (1 point)

Capture images using your own camera and stitch them to create a mosaic. Submit both the captured and stitched images.