Guillaume Drui Computer Vision NYCU
Jakub Zíka
Tymofii Voitekh

Homework 3 Report

Introduction

The objective of this project is to develop an automatic panoramic image stitching system, capable of seamlessly combining two overlapping images into a single panoramic view. Image stitching is widely used in applications such as virtual tours, panoramic photography, and computer vision tasks where wide-angle views are essential. In the sections that follow, we provide a detailed breakdown of the implementation procedure, including key steps for feature detection and matching, homography computation, and image blending for final panorama construction.

Folder structure

- *HW3.py* contains the implementation
- *output* folder contains the result on our own data
- *my_data folder* contains our own data for image stitching

Implementation procedure

After loading the images, we applied OpenCV's Scale-Invariant Feature Transform (SIFT) to detect keypoints and compute descriptors, providing features that are invariant to scale and rotation. To match these features between the two images, we used OpenCV's brute force KNN matcher to find potential matches. To refine these matches, we applied Lowe's ratio test: if the closest match (m) is significantly better than the second-closest match (n), it is likely a valid match. Specifically, if m.distance < 0.75 * n.distance, the match is retained as a 'good match.' The commonly used 0.75 threshold ensures that only matches with a distinctly lower distance to the nearest neighbor (compared to the second-closest) are selected, helping to filter out ambiguous matches.

Ransac

1. Random Sampling of Matches:

   In each iteration, RANSAC selects a random subset of four matched points, from the good matches found earlier, between the two images. Using four points is the minimum required to compute a homography matrix, as each point provides two equations, and we need a total of eight equations to solve for the eight degrees of freedom in H.

2. Computing a Candidate Homography:

The goal is to compute the homography matrix H that maps points from the source image to the destination image. Here is the matrix formulation:

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

(xs,ys): source image points
(xd,yd): destination image points
H is a 3×3 matrix with 8 degrees of freedom, as one of the values can be set as a scaling factor.

We need at least 4 points to solve for H because each point provides 2 equations, and we need 8 equations in total.
Writing it all out and dividing by the z coordinate, this gives us:

$$x_d^{(i)} = \frac{h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}}$$
$$y_d^{(i)} = \frac{h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}}$$

Which can be rewritten to:

$$x_d^{(i)}(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}) = h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}$$
$$y_d^{(i)}(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}) = h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}$$

$$x_s^{(i)}h_{11} + y_s^{(i)}h_{12} + h_{13} - x_d^{(i)}x_s^{(i)}h_{31} - x_d^{(i)}y_s^{(i)}h_{32} - x_d^{(i)}h_{33} = 0,$$
$$x_s^{(i)}h_{21} + y_s^{(i)}h_{22} + h_{23} - y_d^{(i)}x_s^{(i)}h_{31} - y_d^{(i)}y_s^{(i)}h_{32} - y_d^{(i)}h_{33} = 0.$$

For n correspondences, we stack these equations into a matrix A:

$$A = \begin{bmatrix} x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)}x_s^{(1)} & -x_d^{(1)}y_s^{(1)} & -x_d^{(1)} \\ 0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -y_d^{(1)}x_s^{(1)} & -y_d^{(1)}y_s^{(1)} & -y_d^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_s^{(n)} & y_s^{(n)} & 1 & 0 & 0 & 0 & -x_d^{(n)}x_s^{(n)} & -x_d^{(n)}y_s^{(n)} & -x_d^{(n)} \\ 0 & 0 & 0 & x_s^{(n)} & y_s^{(n)} & 1 & -y_d^{(n)}x_s^{(n)} & -y_d^{(n)}y_s^{(n)} & -y_d^{(n)} \end{bmatrix}$$

Thus Ah = 0 where

$$h = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^T$$

The constraint $h^Th=1$ ensures that the solution for h does not collapse to the trivial solution h=0. To enforce this constraint, we introduce a Lagrange multiplier λ and define the following objective function:

$$\mathcal{L}(h, \lambda) = ||Ah||^2 + \lambda(h^Th - 1)$$

Finding the optimal h leads us to the eigenvalue problem:

$$A^T A h = \lambda h$$

The solution for h is the eigenvector corresponding to the smallest eigenvalue of A^TA. This gives the homography matrix H that minimizes the error in mapping points from the source to the destination image.
The code implementation of this follows exactly these steps.

3. Evaluating the Model with Inliers:

   After computing the candidate homography, we apply it to all matched points and measure the transformation error for each point. The error is typically defined as the Euclidean distance between the projected location of each point in the destination image and its actual location. Points with a transformation error below a predefined threshold are considered "inliers," meaning they fit well with the estimated transformation. We count the number of inliers for this candidate homography.

4. Selecting the Best Model:

   RANSAC iterates this process for a specified number of trials (e.g., 2000 iterations) to maximize the chance of finding a reliable model. Each iteration may produce a different candidate homography due to the random sampling of matches. At the end of all iterations, RANSAC selects the homography matrix with the highest number of inliers as the best model. This matrix represents the transformation that aligns the two images while minimizing the impact of outliers.

Image warping and alignment

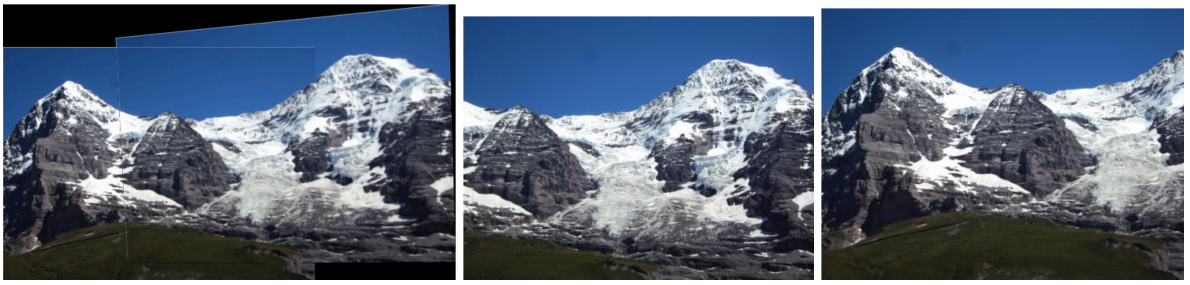
For this part our goal is to align two images using a homography matrix H and smoothly blend them. The following is the approach that was applied to implement a method that achieves just that:

1. First, we identify the merge region for the images. We need to find the transformed coordinates of the corners of img2 after we have applied the homography matrix H. This gives us the bounding box for the new panorama and allows more space for both images to fit in, and avoids the cutoff of any of the parts.

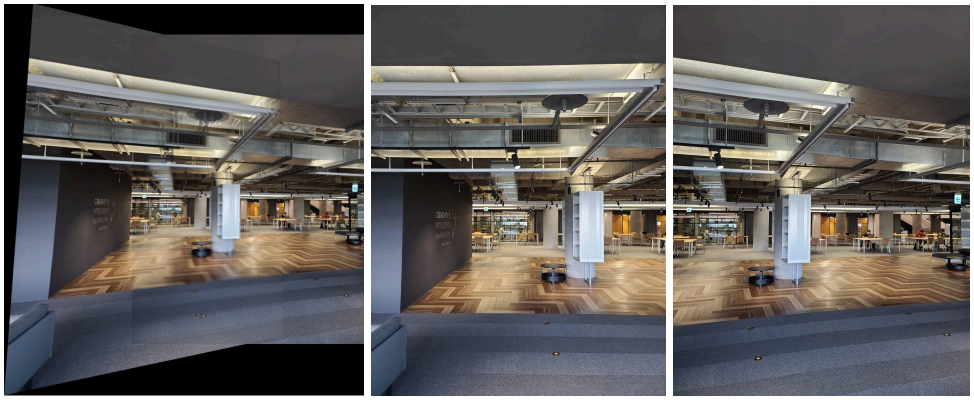
2. Next, we could map pixels from img1 to img2 (forward mapping), but this could lead to empty pixels (holes) in the resulting image. Hence, instead, we used backward mapping. Essentially in backward mapping for each pixel in the output image (result), we map it back to the source image (img1 or img2) via using the inverse of the homography matrix H^-1. This way we can make sure that every pixel in the result is filled without missing pixels.
3. Since backward mapping may map a pixel to non-integer value coordinates in the source images, which is not allowed, we use interpolation to assign a brightness value. What interpolation does is it smoothes the result by taking into consideration nearby pixel values, especially where coordinates fall between the pixels.
4. We want to avoid visible seams in the overlap regions. To achieve this, we implement a blending technique called Alpha Blending (or Linear Blending). In Alpha Blending we calculate an average or weighted combination of pixel values in overlapping areas. In our implementation, we apply a fixed alpha value of 0.5, which assigns equal weight to the overlapping pixels from both images. This means each pixel in the overlap is the average of the corresponding pixels from img1 and img2, creating a smooth transition between the images. The result is a seamless blend that reduces harsh edges and visible seams, allowing the images to merge more naturally.
   In our use case this simple linear blend in sufficient, though more advanced methods (multi-band blending or gradient-domain blending) could be used for finer control

Results

Here are the results for the different panoramas:



Here is the result for our custom images. Since the distance is not that great there is still some mixmatch but all in all it still gives satisfying results.



Discussion

This project focused on implementing a homography matrix and warping technique to align and blend two images into a seamless panorama. Using RANSAC to compute the homography matrix improved robustness by filtering out outliers, ensuring that only reliable correspondences influenced the transformation. This approach effectively handled minor mismatches and noise, producing a more accurate alignment.

Backward mapping combined with interpolation was key to avoiding gaps and ensuring smooth image alignment. By mapping each pixel from the final output back to its source in either image, we achieved a well-rendered warp without missing pixels. Alpha blending further reduced visible seams, creating a smooth transition in overlapping areas, although slight seams persisted in regions with lighting differences.

Overall, the implementation performed well, but challenges such as minor parallax and lighting differences suggest that future improvements like multi-band blending or advanced outlier handling could enhance the results, especially in complex scenarios.

Conclusion

In conclusion, this project successfully implemented an image stitching system that aligns and merges two images into a cohesive panorama. By computing a homography matrix using RANSAC, we ensured robust alignment despite potential outliers. The use of backward mapping and interpolation facilitated smooth warping without missing pixels, and alpha blending provided a simple yet effective method to reduce seams in overlapping areas.

This project reinforced core concepts in computer vision, such as homography estimation, warping, and blending, demonstrating the significance of each step in achieving a visually seamless panorama. With further refinement, this approach could be scaled to handle multiple images and more challenging scenes, broadening its applicability in fields like photography, virtual reality, and mapping.

Work assignment plan between team members

The workload has been divided as follows:

- Guillaume Drui:
    - Code writing

- Report writing

- Jakub Zíka:
  - Code writing
  - report writing

- Tymofii Voitekh:
  - code writing
  - report writing