Image stitching

Introduction:

The panoramic image stitching challenge involves piecing together multiple images of the same scene to form a cohesive panorama, expanding the field of view. While aligning just two images requires finding their homography, achieving seamless panoramas necessitates working with more than two images and applying compensation techniques to minimize differences between them.

In this report study, I present the Python implementation of the Stitch algorithm from the paper [1]. This implementation utilizes SIFT and RANSAC to create panoramas from sets of images, incorporating gain compensation and blending to enhance naturalness. Through demonstrations on the Base and Challenge cases. However, I also acknowledge potential issues, such as excessive distortion or blur in high-frequency details, can arise in the challenging scenario.

SIFT (10%)

The SIFT, or Scale-Invariant Feature Transform, the algorithm is used for detecting and describing local features in the images. Its implementation involves identifying stable keypoints in an image using the Difference of Gaussian (DoG) pyramid. These keypoints are accurately localized by fitting a 3D quadratic function to intensity values near each keypoint like in Fig 1. SIFT then assigns orientations to keypoints based on local gradient orientations, achieving rotational invariance. Descriptors are generated for keypoints by constructing histograms of gradient orientations in their neighborhoods. Finally, SIFT matches keypoints between images by comparing their descriptors using nearest neighbor matching. This method is robust to transformations like scale, rotation, and illumination changes. And since I'm doing image stitching it becomes handy.

KNN (10%)

K Nearest Neighbors (KNN) is utilized in image stitching I are doing to match keypoints between images. Given a keypoint in one image, KNN identifies the k nearest keypoints in another image based on their feature descriptors. These descriptors encode local information about the keypoints' neighborhoods. The distance metric, often Euclidean distance, measures the similarity between descriptors. The keypoint in the second image is then matched to the keypoint in the first image with the most similar descriptor. KNN is simple to implement and can be effective for matching keypoints in image stitching. However, it may suffer from computational inefficiency when dealing with large datasets or high-dimensional feature spaces. Luckily the image complexity is of great concern in our case I got reasonable results

RANSAC (15%)

RANSAC (Random Sample Consensus) is a robust algorithm used in image stitching to estimate transformation models between images despite the presence of outliers and mismatches. In

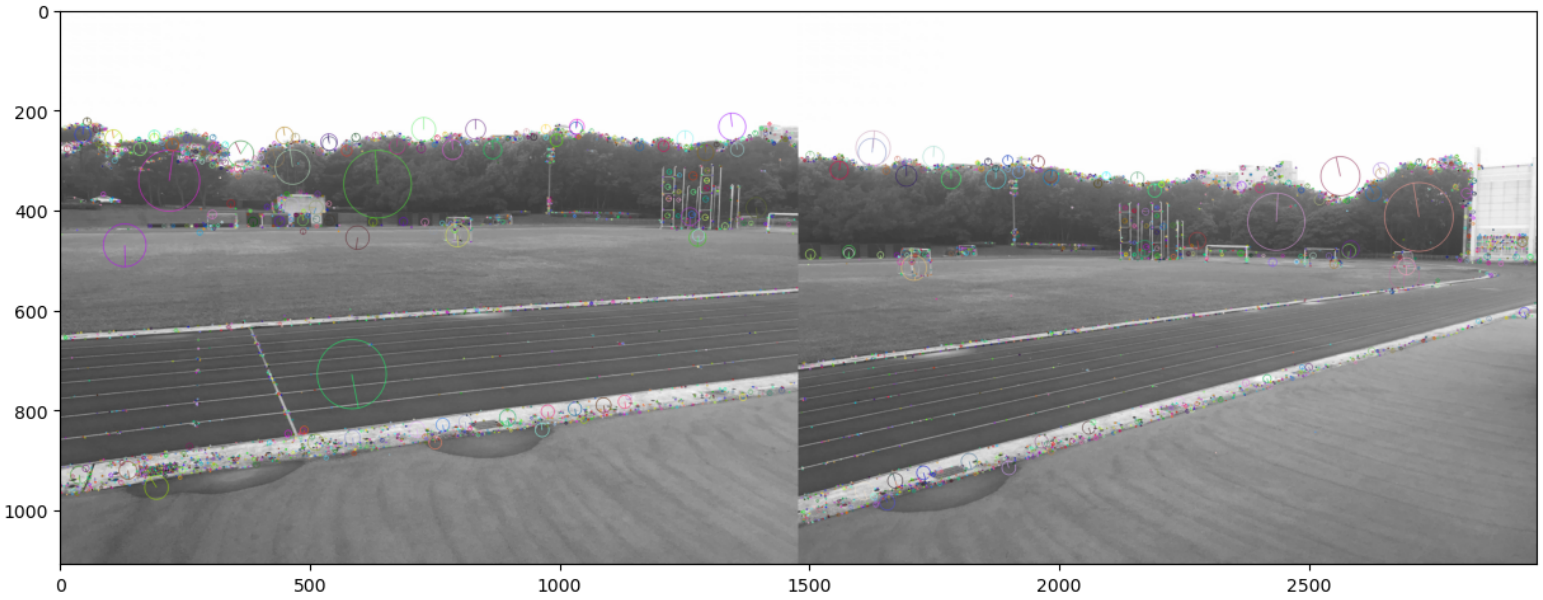
image stitching, RANSAC is typically employed to find the transformation (e.g., homography) that aligns two images properly. It works by randomly selecting a minimal subset of keypoints from both images and estimating a candidate transformation model. The algorithm then evaluates the quality of this model by counting the number of inliers, which are keypoints that are consistent with the model within a certain tolerance. This process is repeated for a specified number of iterations or until a sufficiently good model is found. The final transformation is determined by selecting the model with the maximum number of inliers. RANSAC is effective for handling outliers and robustly estimating transformation parameters, making it suitable for image stitching tasks. However, its performance may depend on the choice of parameters such as the number of iterations and the inlier threshold.

Homography (15%)

Homography is a transformation matrix that maps points from one image to another image in the same scene, typically used in image stitching. In image stitching, homography is used to align different images by estimating the geometric transformation between them. It represents a planar projective transformation, allowing for translation, rotation, scaling, and skewing of the image. Homography is computed using a set of corresponding points (typically keypoints) between images, often obtained through feature matching algorithms like SIFT or SURF. These corresponding points establish the relationship between the images, enabling the estimation of the homography matrix. Once the homography matrix is calculated, it can be applied to warp one image so that it aligns with another image. This process enables the creation of seamless panoramas by blending multiple images together. Homography was essential in our image stitching as it enables the alignment of images with different viewpoints or perspectives, resulting in panoramas.

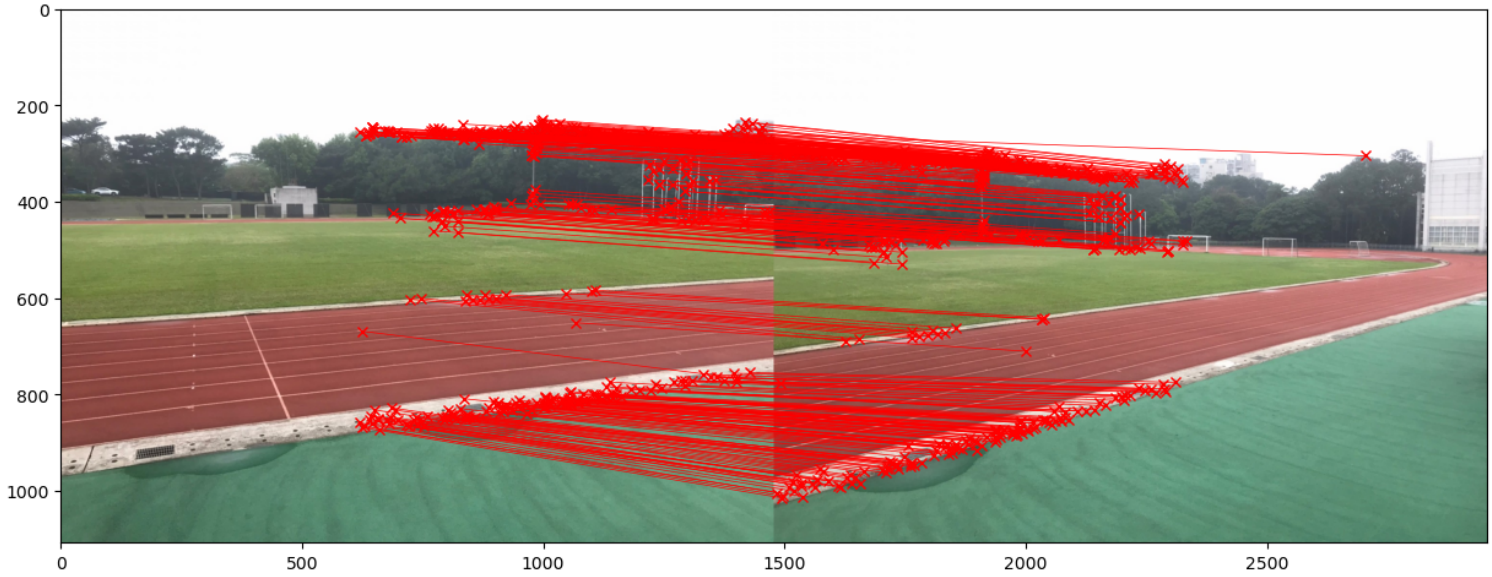
RESULTS:

SIFT performs better when using gray (Fig 1)

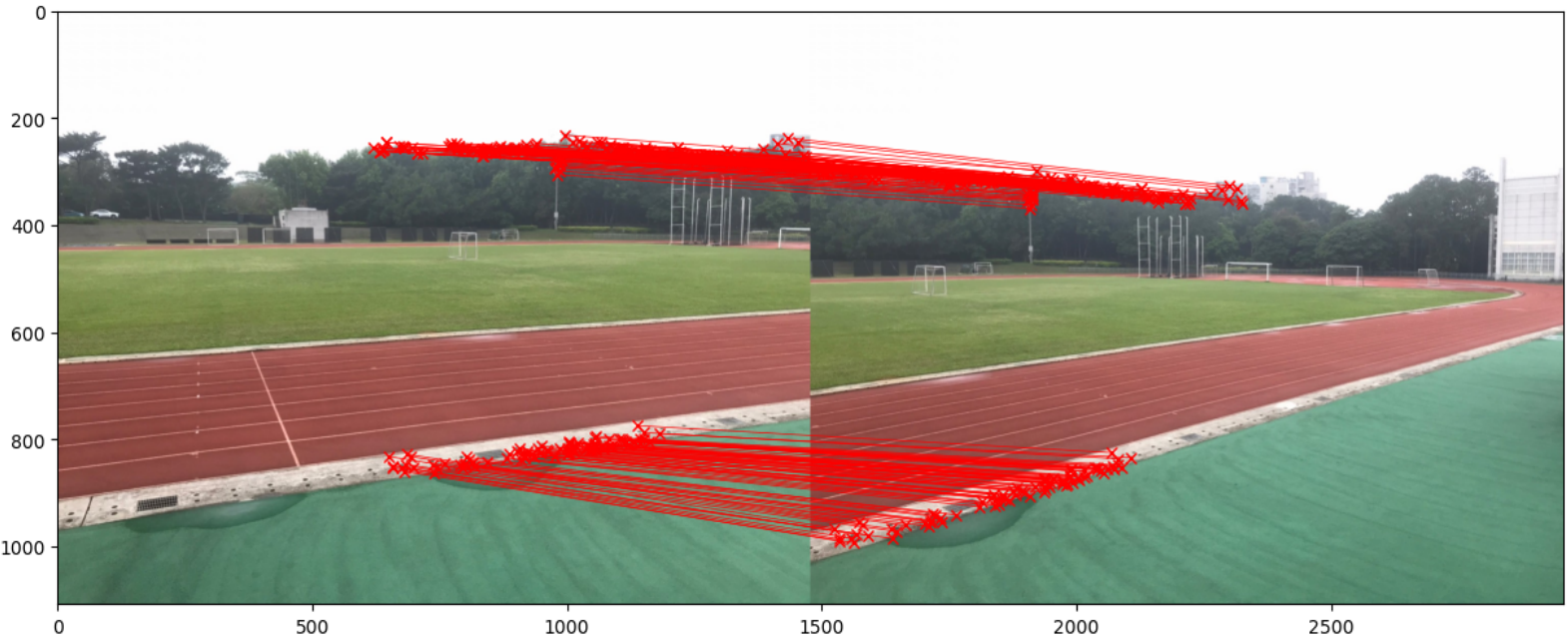


Betsaleel Henry 312540001

Matches between images (Fig 2)



Inliers match (Fig 3)

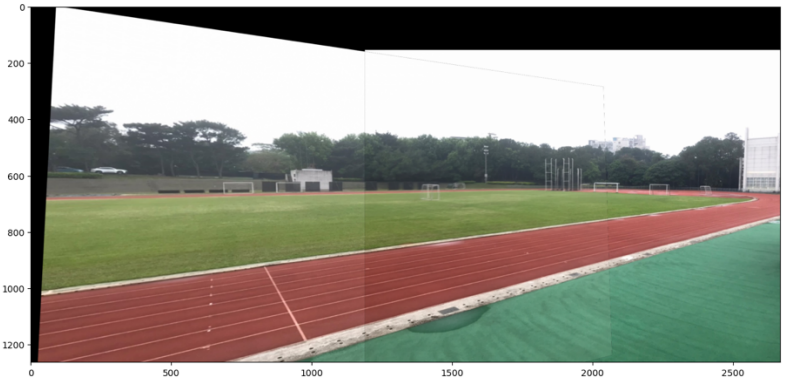


Betsaleel Henry 312540001

Base

Without Blending with cylindrical (Fig 4)

Without Cylindrical modification(Fig 5)



Result After Blending (Fig 6)

Challenge

Stitching Before performing Multi-Blending (Fig 7)

Result After Multi-Blending (Fig 8)

Methods Used

Gain Compensation:

After completing the initial alignment steps, I obtain a panorama where each image is correctly positioned relative to the others. However, the resulting panorama lacks aesthetic appeal, with noticeable discontinuities at the transitions between images. To mitigate these differences, I calculate the overall gain between images and apply compensation to minimize them.

To achieve this, for each panorama, I formulate a global error function that spans all the images, leveraging their overlapping areas. This function is defined as follows:

$$e = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij}\left(\frac{(g_i\bar{I}_{ij} - g_j\bar{I}_{ji})^2}{\sigma_N^2} + \frac{(1-g_i)^2}{\sigma_g^2}\right)$$

where Nij = |R(i, j)| is the number of pixels in image i that overlap in image j, $\bar{I}$ is the mean intensity of the
image i in the overlap region with image j, defined as

$$\bar{I}_{ij} = \frac{\sum_{u_i \in \mathcal{R}(i,j)} I_i(u_i}{N_{ij}}$$

and gi represents the gain for image i (this is what I want to compute). The parameters σN and σg are the standard deviations of the normalized intensity error and gain respectively. I use the values given in the paper, σN = 10.0, (I ∈ {0..255}) and σg = 0.1.
To solve this quadratic function, I set the derivative to 0, which gives:

$$g_j\left(\frac{2}{\sigma_N^2}\sum_{i=1}^{n} N_{ji}\bar{I}_{ji}^2 + \frac{\sum_{i=1}^{n} N_{ki}}{\sigma_g^2}\right) - \frac{2}{\sigma_N^2}\sum_{i=1}^{n} N_{ki}\bar{I}_{ki}\bar{I}_{ik}g_i - \frac{\sum_{i=1}^{n} N_{ki}}{\sigma_g^2} = 0$$

that can eventually be solved with a simple linear solver.

Linear Blending:

While the gain compensation allows to have already good results, the transitions between the different original images is still visible in most panoramas, which is annoying. This can be due to many other effects (misalignment, vignetting, etc.). To further reduce these artefacts, I triedto use a simple blending strategy that uses weights matrices to give more importance to images as we are close to their center, and less importance on the edges. This allows to do a better stitching than simply putting them on top of each other.

For each image, we consider a weight matrix Wij = wiwj where w is a vector that is equal to 0 on both ends and to 1 at the middle. Then, for each pixel of the panorama (x, y), we compute its value as a weighted sum of all the other images (considering that the weight matrices of each image outside of their actual location is equal to zero):

$$I(x,y) = \frac{\sum_{i=1}^{n} I_i(x,y) W_i(x,y)}{\sum_{i=1}^{n} W_i(x,y)}$$

The results from this one shows some part of the luminosity bright (Fig 9)

Multi-band Blending:

The concept underlying multi-band blending involves the blending of low frequencies across a broad spatial range and high frequencies across a narrow range. This approach ensures that fine details are primarily handled by individual images, while blending low frequencies smoothens transitions between images.

To initialize blending weights for each image, I identify the set of points where image.
i contributes the most.

$$W^i_{max}(x,y) = \begin{cases} 1 & \text{if } W^i(x,y) = \max_j W^j(x,y) \\ 0 & \text{otherwise} \end{cases}$$

These max-weight maps are successively blurred to form the blending weights for each band, along with a similar blurring performed on the image itself to build each band. For the first band, a high pass version of the rendered image is formed:

$$I^i_\sigma(x,y) = I^i(x,y) * g_\sigma(x,y)$$
$$B^i_\sigma(x,y) = I^i(x,y) - I^i_\sigma(x,y)$$

where gσ is a Gaussian of standard deviation σ, and ∗ denotes convolution. In doing so, Bσ represents spacial fre- quencies in the range of wavelengths λ ∈ [0, σ]. The weight matrix associated with this band is computed using the same convolution:

$$W^i_\sigma(x,y) = W^i_{\max}(x,y) * g_\sigma(x,y)$$

Then, other bands are computed iteratively, using a stan- dard deviation σ′ = sqrt(2k + 1)σ that is getting larger, with k ≥ 1:

$$I^i_{(k+1)\sigma} = I^i_{k\sigma} * g_{\sigma'}$$
$$B^i_{(k+1)\sigma} = I^i_{k\sigma} - I^i_{(k+1)\sigma}$$
$$W^i_{(k+1)\sigma} = W^i_{k\sigma} * g_{\sigma'}$$

This way, each band k represents spacial frequencies in the range of wavelengths λ ∈ [sqrt(2k – 1)σ, sqrt(2k + 1)σ]. The bands for the overall panorama are obtained by linear combination of the different images:

$$I_{k\sigma}^{multi}(x,y) = \frac{\sum_{i=1} nB_{k\sigma}^{i}(x,y)W_{k\sigma}^{i}(x,y)}{\sum_{i=1}^{n} >_{k\sigma}^{i}(x,y)}$$

For the final band, the remaining portions of the image are captured to represent all the low frequencies within the image.

$$B_{K\sigma}^{i} = I_{(K-1)\sigma}$$
$$W_{K\sigma}^{i} = W_{(K-1)\sigma}$$

Finally, different bands are added together to obtain the final panorama (see Fig 8).
More details and illustrations about this method can be found in the AutoStitch paper [1].

Conclusion:

This project provided an opportunity to develop an image stitching algorithm from scratch, leveraging existing methods for specific operations where implementation from scratch would have been impractical. The final Python implementation of the algorithm is highly efficient and incorporates advanced features that enhance the user experience:

- Image matching using SIFT and RANSAC enables the recognition of multiple distinct panoramas within a set of images, while also filtering out any noisy images not contributing to any panorama.
- Gain compensation contributes to a more natural appearance of the final panorama by reducing discrepancies between images.
- Linear and multi-band blending techniques further refine the panorama's appearance by smoothing transitions between images. However, the results achieved with multi-band blending are somewhat underwhelming.

Reference:

[1] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. International Journal of Computer Vision, 74(1):59–73, 2007. 1, 2, 3, 6

[2] Konstantinos G Derpanis. The harris corner detector. York University, 2, 2004. 2

[3] Richard Hartley and Andrew Zisserman. Multiple view geom- etry in computer vision. Cambridge university press, 2003. 2

[4] Chung-Ching Lin, Sharathchandra U Pankanti, Karthikeyan Natesan Ramamurthy, and Aleksandr Y Aravkin. Adaptive as-natural-as-possible image stitching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recogni- tion, pages 1155–1163, 2015. 2

[5] David G Lowe. Distinctive image features from scale- invariant keypoints. International journal of computer vision, 60(2):91–110, 2004. 2

[6] Julio Zaragoza, Tat-Jun Chin, Michael S Brown, and David Suter. As-projective-as-possible image stitching with moving dlt. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2339–2346, 2013. 2