Automatic Image Stitching

Ananay Garg[1], Mrigank Ashesh[2] and Hardik Jain[3]

***Abstract*— In this project report, we tried to implement the paper by Lowe, Brown['03]. We couldn't find a complete and exact open source implementation of the algorithm, and one of the team members required it for their pet-projects. This project wants to do fully automated panoramic image stitching. There are three parts: (1) Feature recognition independent of illumination, ordering, scale of the input image. This is also insensitive to noise. (2) Image matching using RANSAC, followed by a probabilistic verification of the model. (3) Performing multi-band blending to remove stitch marks.**

I. INTRODUCTION

In this report, we describe an invariant feature based approach to fully automatic panoramic image stitching. Advantages of such an approach are: (1) Use of invariant features enables reliable matching of panoramic image sequences despite rotation, zoom and illumination change in the input images. (2) We can automatically discover the matching relationships between the images, and recognize panoramas in unordered datasets. (3) We generate high-quality results using multi-band blending to render seamless output panoramas.

The structure of the report is as follows. Section 2 will discuss all the three steps in further details as subsections, namely, feature detection through SIFT, Image Matching and multiband blending. Section 3 and 4, will discuss results and future scope respectively. Section 5, briefly touches about learning points followed by a link to the code.

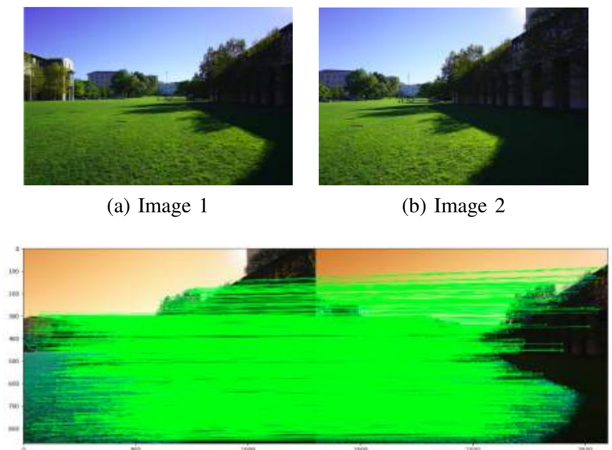
II. ALGORITHM

A. Feature Matching

The first step in the panoramic recognition algorithm is to extract and match SIFT features between all of the images. SIFT features are located at scale-space maxima/minima of a difference of Gaussian function. At each feature location, a characteristic scale and orientation is established. This gives a similarity-invariant frame in which to make measurements.

The SIFT algorithm returns key points are descriptors. Descriptors summarize, in vector format (of constant length) some characteristics about the key points. For example, it could be their intensity in the direction of their most pronounced orientation. It's assigning a numerical description to the area of the image the key point refers to. Since SIFT features are invariant under rotation and scale changes, our system can handle images with varying orientation and zoom.

Once features have been extracted from all n images (linear time), they must be matched. Since multiple images may overlap a single ray, each feature is matched to its k nearest neighbours in feature space (we use k = 2). This can be done in $O(n\ logn)$ time by using a k-d tree to find approximate nearest neighbours (Beis and Lowe, 1997). A k-d tree is an axis aligned binary space partition, which recursively partitions the feature space at the mean in the dimension with highest variance.



(a) Image 1

(b) Image 2

(c) SIFT matches

Fig. 1: Feature matching: Every matched feature is shown as a green line from one image to the other

B. Image Matching

At this stage the objective is to find all matching (i.e. overlapping) images. Connected sets of image matches will later become panoramas.

From the feature matching step, we have identified images that have a large number of matches between them. We consider a constant number m images, that have the greatest number of feature matches to the current image, as potential image matches (we use $m = 6$). First, we use RANSAC to select a set of inliers that are compatible with a homography between the images. Next we apply a probabilistic model to verify the match.

1) Robust Homography Estimation using RANSAC: RANSAC (random sample consensus) is a robust estimation procedure that uses a minimal set of randomly sampled correspondences to estimate image transformation parameters, and finds a solution that has the best consensus with the data.

$$p(H\ is\ correct) = 1 - (1 - (p_i^r))^n$$

This equation represents the the probability of finding the correct transformation after n trials, where p_i is the

[1]Ananay Garg - 160260007
[2]Mrigank Ashesh - 160260021
[3]Hardik Jain - 16D260008

Fig. 2: Image obtained after stitching. The stitch boundaries are seen as black lines.

inlier probability and r is the number of features, for us it is 2.

2) Probabilistic Model for Image Match Verification: For each pair of potentially matching images we have a set of feature matches that are geometrically consistent (RANSAC inliers) and a set of features that are inside the area of overlap but not consistent (RANSAC outliers). The idea of our verification model is to compare the probabilities that this set of inliers/outliers was generated by a correct image match or by a false image match.

For a given image we denote the total number of features in the area of overlap n_f and the number of inliers n_i. The event that this image matches correctly/incorrectly is represented by the binary variable $m \in \{0, 1\}$. The event that the ith feature match $f^{(i)} \in \{0, 1\}$ is an inlier/outlier is assumed to be independent Bernoulli, so that the total number of inliers is Binomial.

$$p(f^{(1:n_f)}|m = 1) = B(n_i; n_f, p_1)$$

$$p(f^{(1:n_f)}|m = 0) = B(n_i; n_f, p_0)$$

From the paper, it follows that choosing values $p(m = 1) = 10^6$ and $p_{min} = 0.999$ gives the condition:

$$n_i > \alpha + \beta n_f$$

for a correct image match, where $\alpha = 8.0$ and $\beta = 0.3$.

Once pairwise matches have been established between images, we can find panoramic sequences as connected sets of matching images. This allows us to recognize multiple panoramas in a set of images, and reject noise images which match to no other images

C. Multi-band Blending

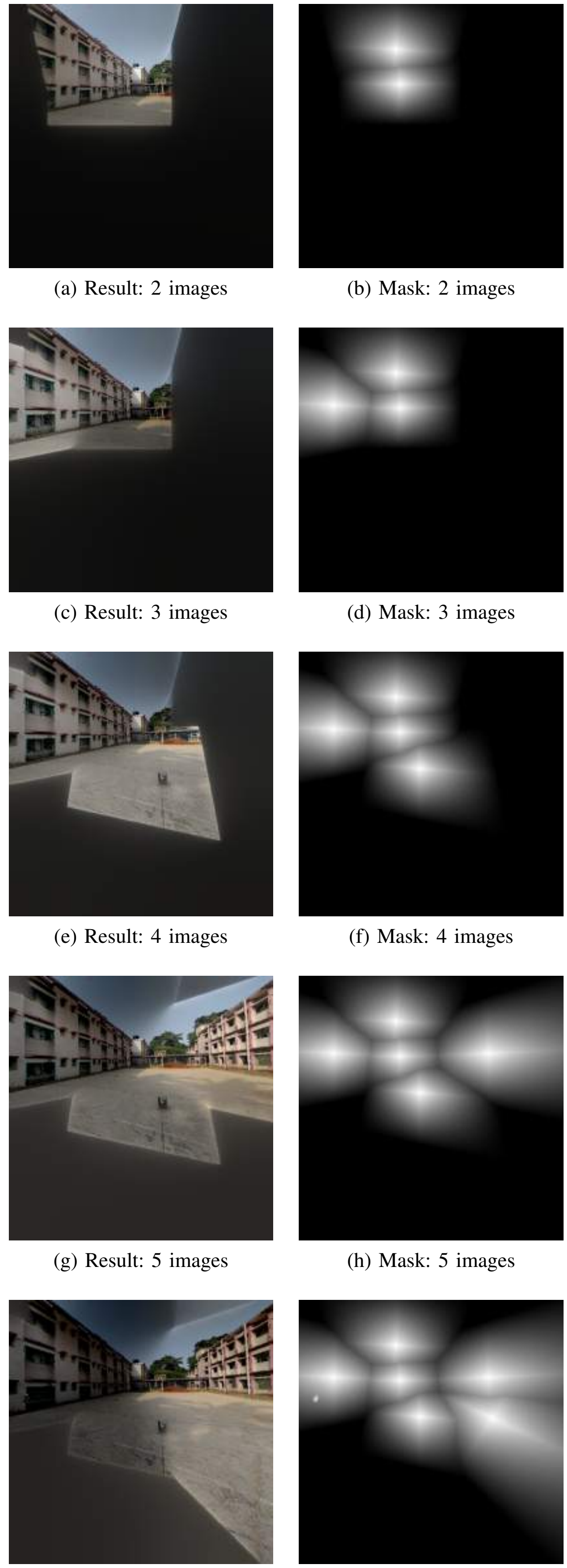
The above methods used to stitch images still has some stitching marks. We implemented Multi Band Blending algorithm of Burt and Adelson to get rid of the stitching marks. To combine information from multiple images we assign a weight function to each image W(x, y) = w(x)w(y) where w(x) varies linearly from 1 at the centre of the image to 0 at the edge. We initialize blending weights for each image by finding the set of points for which image i is most responsible

$$W^i_{max} = \begin{cases} 1, & if\ W_i = argmax_j W_j. \quad (1) \\ 0, & \text{otherwise.} \quad (2) \end{cases}$$

We form Laplacian Pyramid of the images upto k levels.

$$B^i_{(k+1)} = I^i_k - I^i_{(k+1)}$$

$$I^i_{(k+1)} = I^i_k * g$$



(a) Result: 2 images (b) Mask: 2 images

(c) Result: 3 images (d) Mask: 3 images

(e) Result: 4 images (f) Mask: 4 images

(g) Result: 5 images (h) Mask: 5 images

(i) Result: 6 images (j) Mask: 6 images

Fig. 3: Algorithm applied on images taken from Hostel 6, basketball court

$$W^i_{(k+1)} = W^k_i * g$$

Also, we form the Gaussian Pyramid of the mask W_i upto k levels. Now weighted average of each of the j^{th} level of the images are computed with the corresponding W^i_j as weights. Thus we have a blended pyramid of k levels. These k levels are used to reconstruct the final blended image from the blended pyramid by upsampling the j^{th} level image and adding it to the resized j^{th}.

III. Results

The obtained results are shown in Figure 3 and in Figure 4. The results weren't compared to state of the art papers, but there are some errors which can be easily worked on. This includes, the blackening of images when high number of images are added, blackening towards the edges of the image and panaromic straightening.

IV. Conclusions and Future Scope

Though the results are acceptable, there is a lot of scope of improvement in the results. The following points were also reworked in the papers published later.

- **Gain Compensation:** As is clearly seen in the results, the code messes up with the illumination and contrast balancing. In the second version of the paper in '07, an extra layer of gain compensation was added.
- **Scene Motion:** As is clearly seen in the results, the code messes up with the illumination and contrast balancing. In the second version of the paper in '07, an extra layer of gain compensation was added.
- **Automatic Panoramic Straightening:** It was seen, when multiple horizontal images were stitched, there was this 'wavy' structure of the images. This is probably due to accumulation of alignment errors across images. This can be corrected.
- **Photo metric modelling:** It is generally observed that the corners of a image are less bright as compared to the middle. This can be modelled and eliminated.

V. Learning Points

This project was beneficial to all the team members in the following ways:

- We got an opportunity to implement theoretical concepts of image processing, like laplacian pyramids, masks, etc.
- Reading, skimming and subsequently understanding research papers in the field of image processing
- Working in a team and managing the synergy towards the common goal

VI. Code

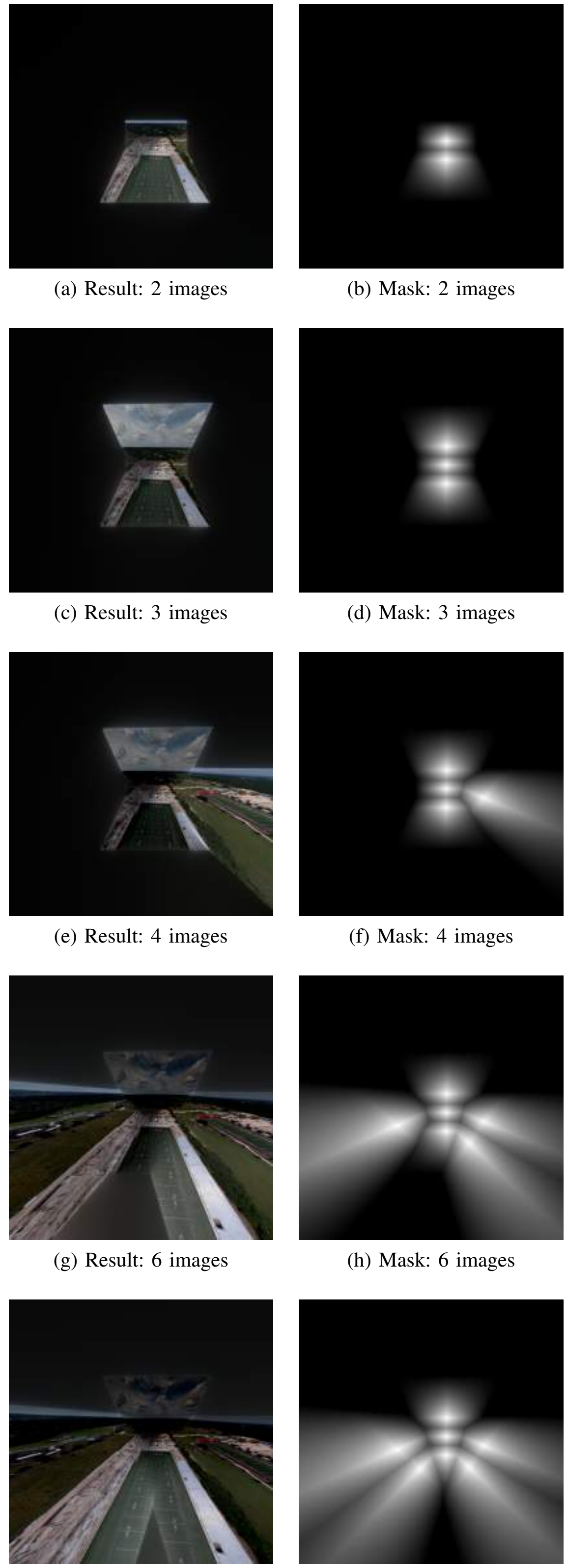
The link of the code is Github Code. As a part of the initial aim, we wanted to release the code for futhur development. We couldn't find a complete implementation of the paper.

References

[1] Matthew Brown and David G. Lowe, Automatic Panoramic Image Stitching using Invariant Features, International Journal of Computer Vision 74(1), 5973, 2007

[2] https://github.com/kushalvyas/Python-Multiple-Image-Stitching

[3] http://matthewalunbrown.com/papers/iccv2003.pdf



(a) Result: 2 images (b) Mask: 2 images

(c) Result: 3 images (d) Mask: 3 images

(e) Result: 4 images (f) Mask: 4 images

(g) Result: 6 images (h) Mask: 6 images

(i) Result: 7 images (j) Mask: 7 images

Fig. 4: Algorithm applied on sample images