Digital Image Processing

Fast Panorama Stitching on Mobile Phones

(Github Link)

Fig. 12. Panoramic image produced by the fast panorama stitching with 7 source images in an indoor scene with moving objects.

Additya Popli - 20161215
Devansh Gautam - 20161171

24th November, 2018

Abstract

Image stitching is the process of creating high-resolution and high-quality panoramic images from long image sequences with very different colors and luminance in source images.

The algorithm proposed here uses a sequential stitching method to combine the set of photos. This approach converts individual photos into a single panoramic photo using very little memory and is thus suitable to be implemented for mobile phones. In this approach, color correction is done initially to reduce color differences of source images and balances colors and luminance in the whole image sequence. Dynamic programming finds optimal seams in overlapping areas between adjacent images and merges them together. Finally, blending is done to further smoothen the transition from one image to other and hence hide the seam and any artifacts which may arise due to stitching. The approach has been tested with different image sequences and it works well on both indoor and outdoor scenes.

Sequential stitching helps to keep the memory requirements low since now we do not need to keep all the given images in memory together, it is just sufficient to keep the output image and the next image in the sequence to be stitched. Dynamic Programming for seam finding makes the algorithm extremely fast.

Introduction

Problem Statement

The goal of the project is to understand and implement an algorithms that takes a set of high-quality images and stitches these images together to obtain a panoramic image. The main challenge here is the make ensure that the algorithms is efficient in terms of both memory and time complexity, while maintaining the quality of the output stitched image.

Motivation

With rapid advancement in technology, focus is now being shifted from desktops to portable devices like mobile phones. Mobile phones which were earlier limited in use as communication tools only, now are equipped with high-resolution digital cameras, high-quality color displays, and GPU hardware which make then very capable computation devices. Even with so much advancement, mobiles phones are still not computationally strong enough to be able to run existing image stitching algorithms efficiently. Thus, there is a need to an algorithms that helps users click high-quality and high-resolution images and then merge them using to obtain a single high-quality panoramic image.

Constraints

The algorithm assumes that the overlap between adjacent images, and the order of the images in which they are supposed to be stitched is known.

Overview

The basic steps followed are :

- **Pre - Processing Images :** Colour correction is done on the given input images to account for changes in colours and lighting which might create artificial edges in the resultant image.
- **Stitching Adjacent Images :** An error surface is constructed with squared differences between overlapping images. A low-cost path is found through the error surface by dynamic programming and used as an optimal seam to merge the images.
- **Blending :** Whenever a new image is merged with the current resultant image, the colours of both the images needs to be blended to hide the seam used to stitch the images.

Procedure

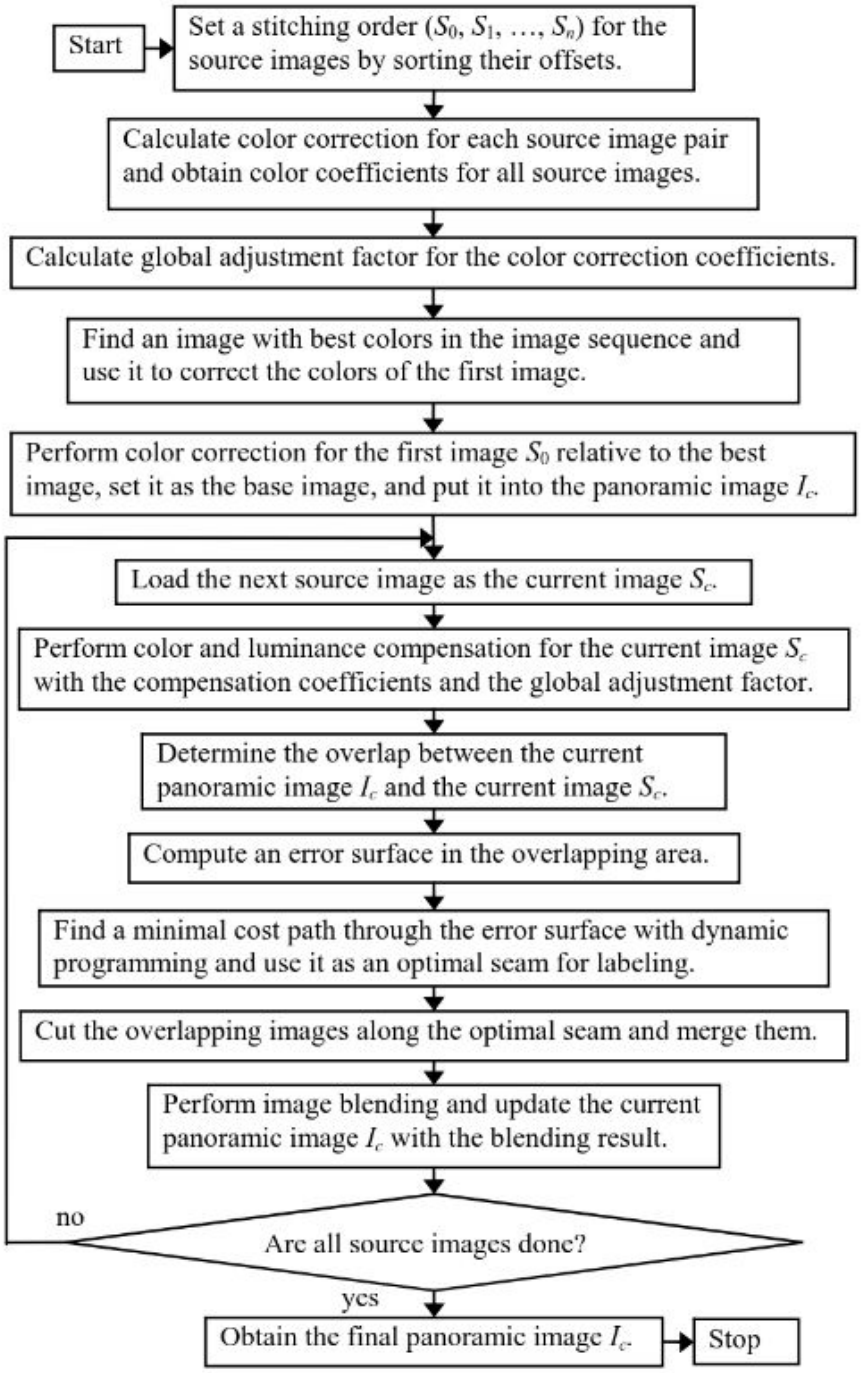


Fig. 1. Workflow of the fast panorama stitching approach.

1. Colour Correction

Different images, although of the same scene, may have different values for illumination, exposure, white balance etc, which if not corrected will lead to creation of visible artifacts.

How to do it?

In an image sequence, $S_0, S_1, .S_{i-1}, S_i, S_{i+1}$, where S_i $and S\_\{i+1\} are adjacent$ mages, we find their overlapping area by the value of the overlap provided to us.

Let us denote this by S_{i-1}^o and S_i^o. Now our color correction coefficient, calculated separately for each channel is -

$$\alpha_{c,i} = \frac{\sum_p (P_{c,i-1}(p))^\gamma}{\sum_p (P_{c,i}(p))^\gamma} \qquad c \in \{R, G, B\}\ (i = 1,2,3,\cdots,n),$$

Where *P(c, p)* means the p[th] pixel of S of the c[th] channel and *gamma* is just a coefficient which we take as 2.2.

To avoid saturating colour values, a global adjustment is done for the whole image sequence by calculating a global coefficient separately for each colour channel given by:

$$g_c = \frac{\sum_{i=0}^{n} \alpha_{c,i}}{\sum_{i=0}^{n} \alpha_{c,i}^2} \quad c \in \{R, G, B\}\ (i = 0,1,\ldots,n).$$

With these coefficients, we perform colour correction according to the formula

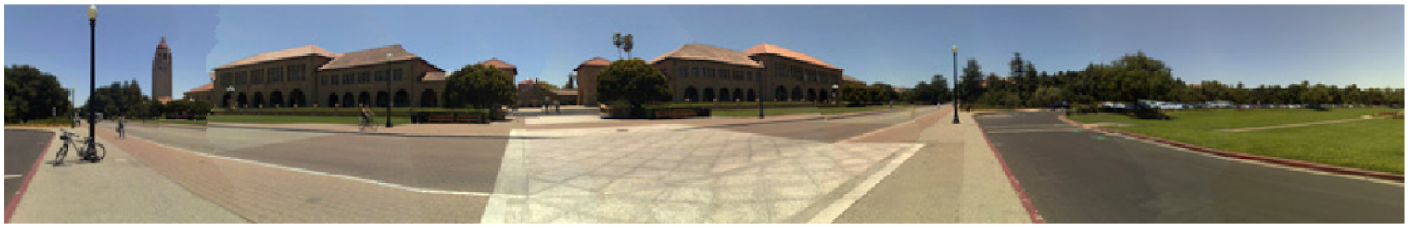
$$P_{c,i}(p) \leftarrow (g_c \alpha_{c,i})^{1/\gamma} P_{c,i}(p), c \in \{R, G, B\}\ (i = 0,1,\ldots,n)$$

Now, there might be a case the the first image might now always be the ideal image to be used to correct other images, so we first find such an “ideal” image and use it to colour-correct the first image.

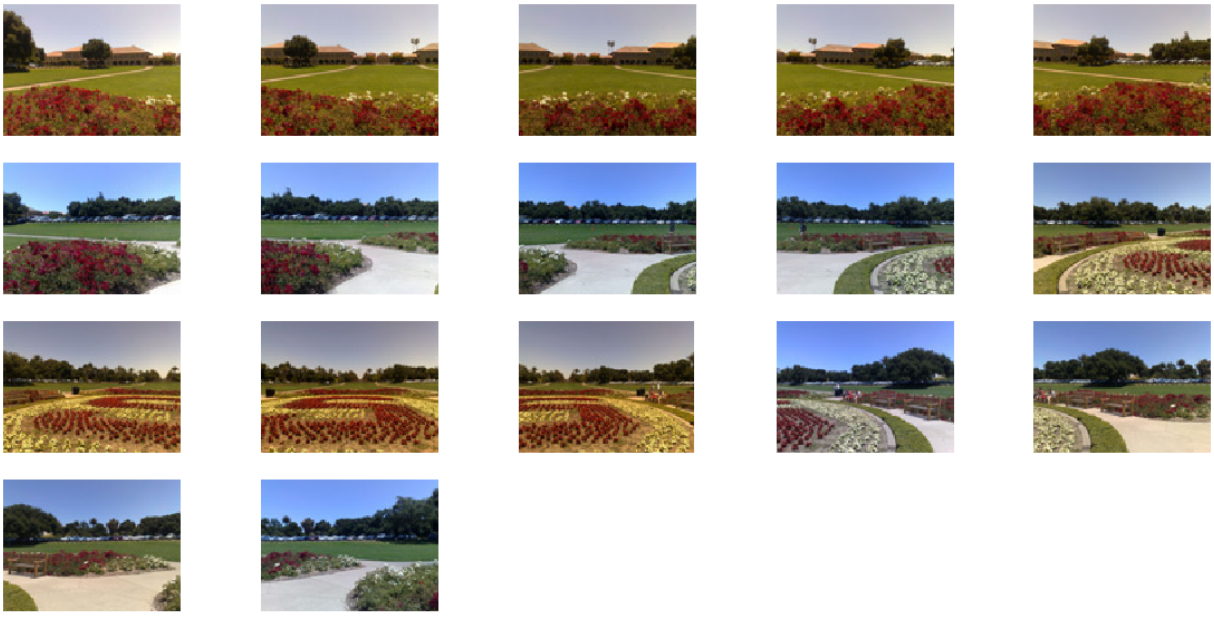
Source Images

Without Colour Correction

With Colour Correction

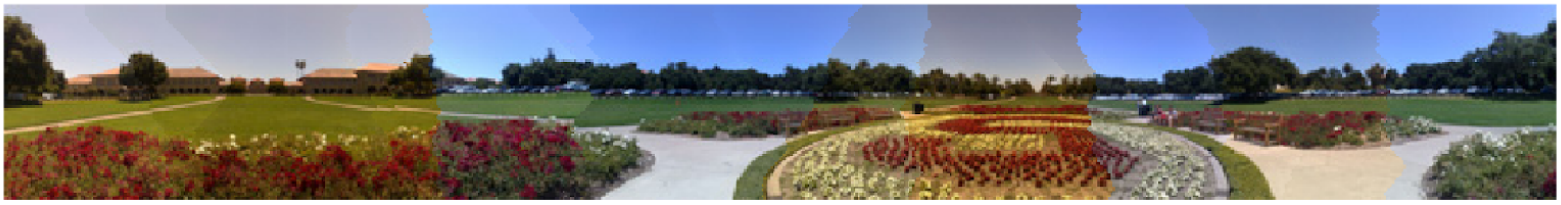


Source Images



With Colour Correction

Without Colour Correction



2. Optimal Seam Finding

The objective of seam finding is the find a path along with adjacent images are to be merged. We used Dynamic Programming to solve this task.

An error surface is constructed with squared differences between overlapping images. A low-cost path is found through the error surface by dynamic programming and used as an optimal seam to create labeling and the overlapping images are merged together along the optimal seam.

Suppose that $abcd$ is the overlapping area between the current composite image I_c and the current source image S_c. I_i^o and S_i^o are the overlapping images in the area $abcd$ of I_c and S_c respectively. We compute squared differences $error$ between I_i^o and S_{i^o} as an error surface,

$$error = (I_c^o - S_c^o)$$

We apply dynamic programming to find a minimal cost path through this surface. We loop through each pixel, and compute a minimum squared difference array defined as -

$$E_{h,w} = e_{h,w} + min(E_{h-1,w-1}, E_{h-1,w}, E_{h-1,w+1})$$

for all valid h, w.

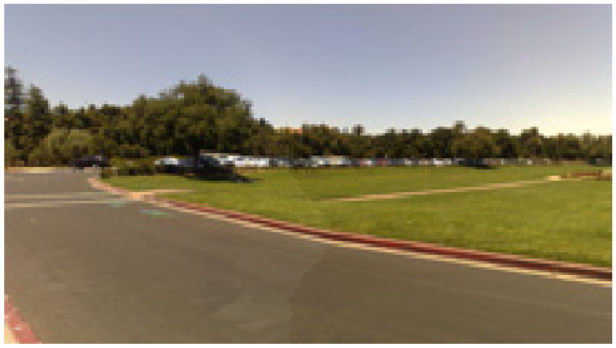
Now to trace back the actual path from this, we start at the bottom most row, pick h, w such that $E_{h,w}$ is minimum for all possible values of h, w. Now, from each h, w, we find h', w' such that $h' = h - 1$ and w' is such that $E_{h',w'}$ is minimum among $w' = \{w - 1, w, w + 1\}$ until we reach the first row.

Alternate Method : Other methods exist to find the seam (example using Graph Cut), which are not feasible here due to their high computational requirements.

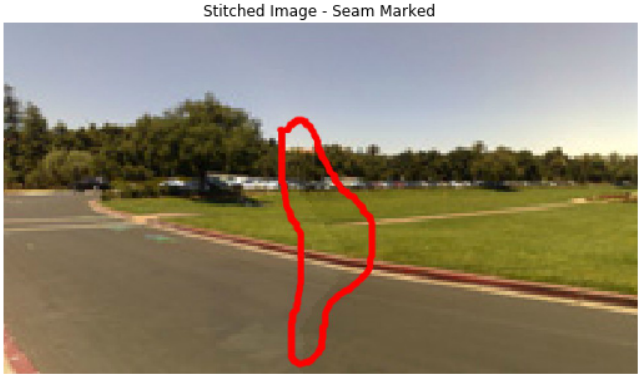
Source #1

Source #2

Stitched Image



Stitched Image - Seam Marked



3. Image Blending

We take a band δ pixels wide on both sides of the seam, and adjust the values of these pixels. The new value for a pixel is calculated by a weighted combination of pixels values of the two adjacent images, defined as -

$$P_{I_{c,new}}(p) = \frac{d_1^n P_{I_c}(p) + d_2^n P_{S_c}(p)}{d_1^n + d_2^n},$$

Where d_1 and d_2 are the distances from the pixel p to the boundaries, $P_{I_{c,new}}$ is the new colour of the pixel, $P_{Ic}(p)$ is the colour of pixel p in the image I_c and $P_{Sc}(p)$ is the colour of pixel p in the same S_c.

n here is like a hyperparameter, changing the values of n leads to different transformations.

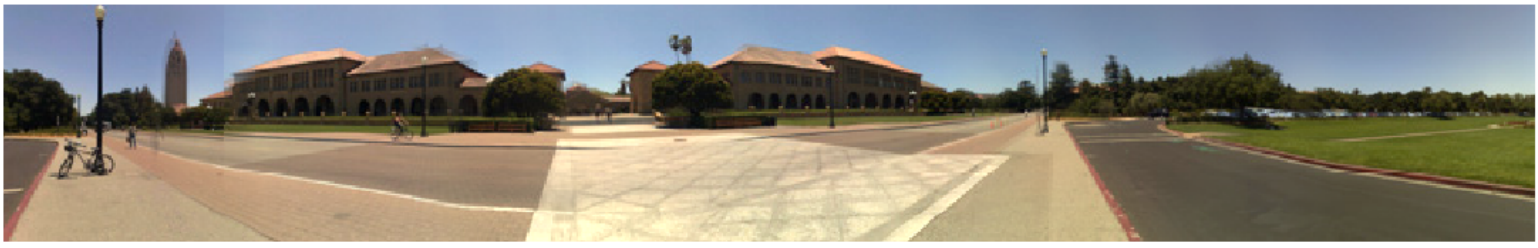
This is a very simple operation, and thus the memory and computational costs are very low, which is ideal for our purpose.

Stitched Image without blending

Stitched Image with blending

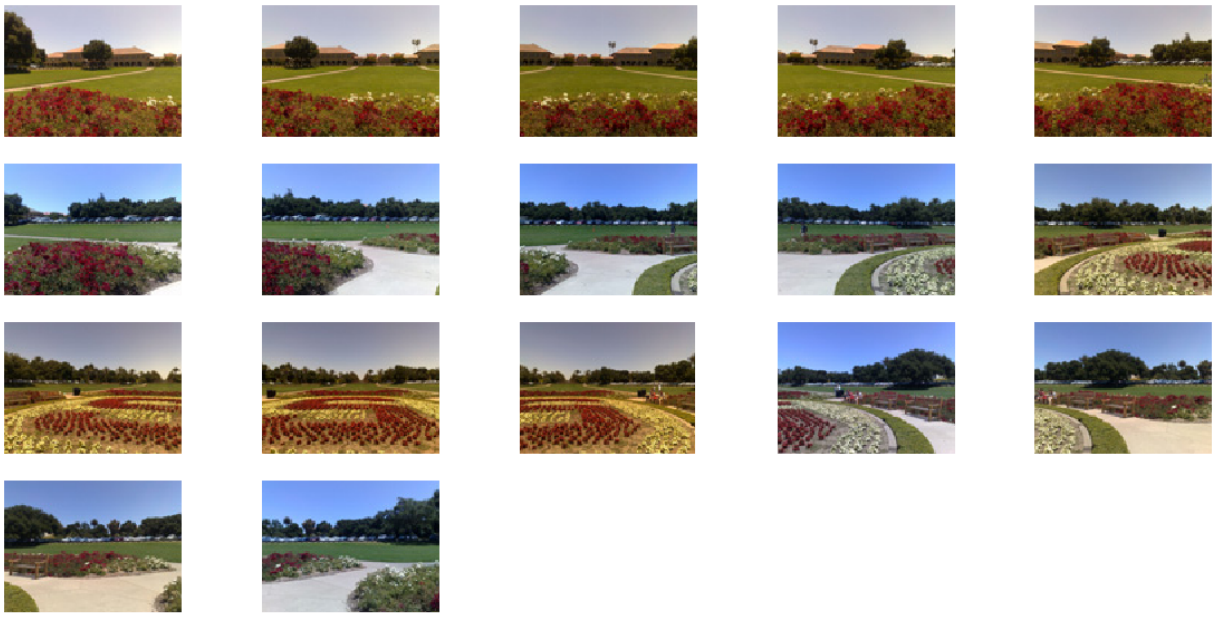
Stitched Image without blending

Stitched Image with blending



Examples

Source Images



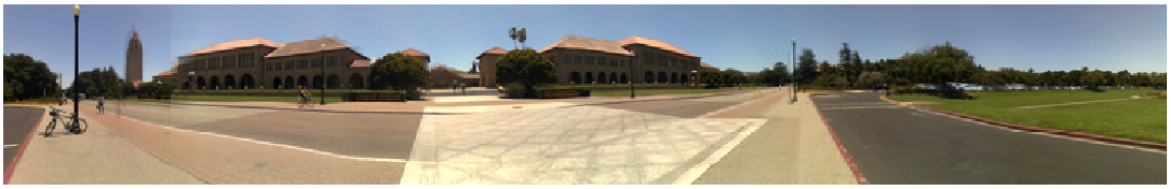
Stitched Image

Source Images

Stitched Image

Source Images

Stitched Image



Source Images






Additional Experiments

- ## Overlap

  The algorithm as described in the paper uses a fixed overlap between adjacent images. We have modified the code such that the overlap can now be different between different pairs of adjacent images, but is still known beforehand.

  How to remove this restriction?

  We consider a range of values $[x, y]$ as valid values for the overlap and loop through them. For each such $x \leq z \leq y$ we calculate the error function and the required overlap is defined by -

  $overlap = argmin(z)E'(z)$ where $E'(z)$ is the cumulative value of the error function e.

  Problem with this?

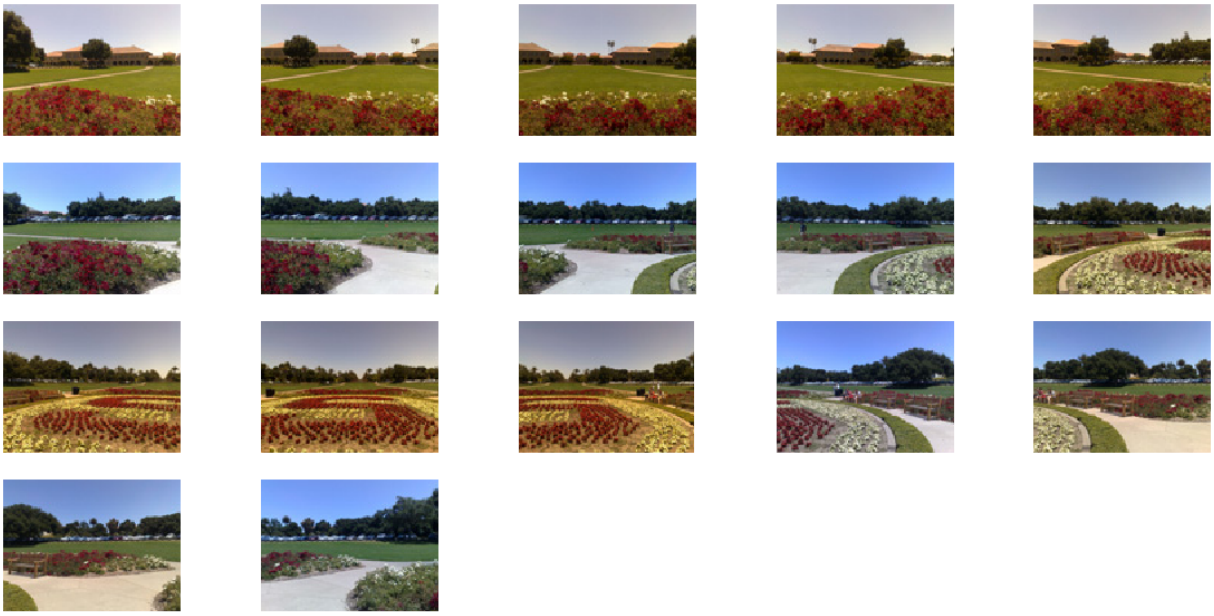
  But prior to seam finding, colour correction is done, which assumes the overlap is provided. So to eliminate that, we can assume 20 - 30% overlap between adjacent images and calculate correction parameters according to that.

- ## “Ideal” Image for correction of first image

  Colour Correction for the first image is done with respect to some ideal image. But this is ambiguous, since there can be multiple ways as to how an image can be called ideal. We tried multiple functions as “ideal functions”.

  For example, we tried to maximize the contrast of the image by taking the variance into account rather than taking the means of each channel as mentioned in the paper.

Source Images



This is the output we get when take the method mentioned in the paper, the best image in this case is the 15th image.

This is the output we get when we maximize the variance instead. The best image in this case is the 6th image.

- **Comparison with SIFT**

Source Images




We can see that the outputs we get are similar.

While the SIFT approach takes 0.8473 seconds to get the output image after merging these four images, our method takes just 0.64 seconds.

Although the time difference isn't much noticeable here, it increases as we increase the number of images to merge and these differences become significant when we have to run these algorithms on mobiles which have less computing power.

Stitched Image using SIFT

Stitched Image using our method

Division of Work

- Additya
    - Dataset collection
    - Color Correction
    - Calculation of error surface between overlapping images
    - Comparison with SIFT
- Devansh
    - Dataset Collection
    - Finding a low cost path through the error surface using dynamic programming
    - Selection of base image for colour correction
    - Image Blending