Fast Panorama Stitching on Mobile Devices

Digital Image Processing - Project

Additya Popli - 20161215
Devansh Gautam - 20161171

Project Overview

The goal of the project is to understand and implement an algorithm 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.

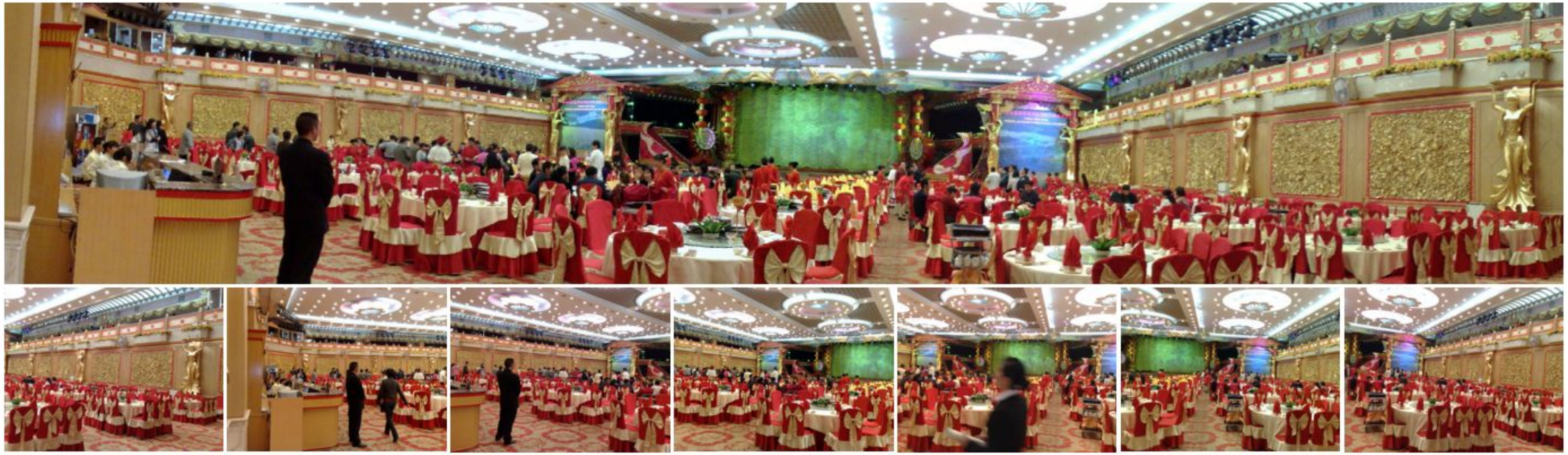
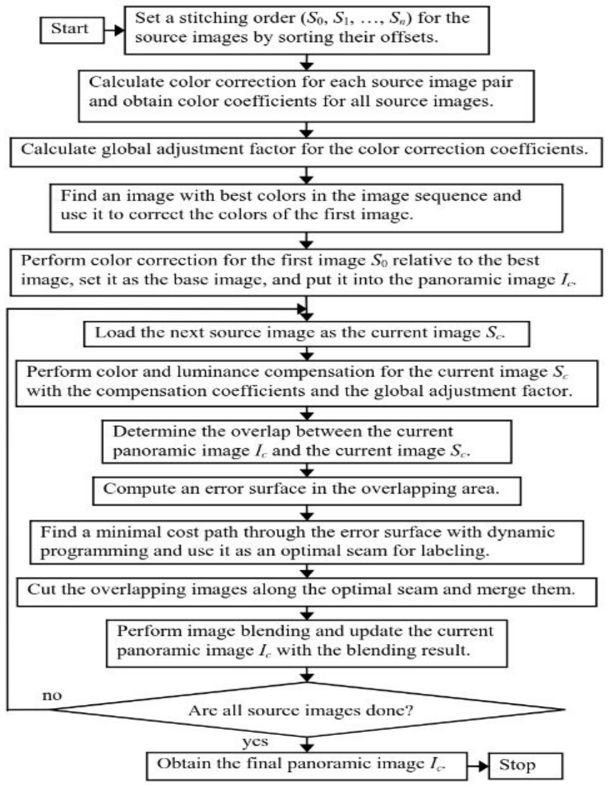


Fig. 12. Panoramic image produced by the fast panorama stitching with 7 source images in an indoor scene with moving objects.

Basic Pipeline

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.



Start
Set a stitching order (S_0, S_1, …, S_n) for the source images by sorting their offsets.
Calculate color correction for each source image pair and obtain color coefficients for all source images.
Calculate global adjustment factor for the color correction coefficients.
Find an image with best colors in the image sequence and use it to correct the colors of the first image.
Perform color correction for the first image S_0 relative to the best image, set it as the base image, and put it into the panoramic image I_c.
Load the next source image as the current image S_c.
Perform color and luminance compensation for the current image S_c with the compensation coefficients and the global adjustment factor.
Determine the overlap between the current panoramic image I_c and the current image S_c.
Compute an error surface in the overlapping area.
Find a minimal cost path through the error surface with dynamic programming and use it as an optimal seam for labeling.
Cut the overlapping images along the optimal seam and merge them.
Perform image blending and update the current panoramic image I_c with the blending result.
no
Are all source images done?
yes
Obtain the final panoramic image I_c.
Stop

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

Colour Correction

- Different images, although of the same scene, have different values for illumination, exposure and white balance. This if not corrected will lead to creation of visible artifacts.
- Consider an image sequence, S_0, S_1, ... S_{i-1}, S_i, S_{i+1} .... S_N, where S_i and S_{i+1} are adjacent images, 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}.
- Color correction coefficient, calculated separately for each channel is -

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

Where *P(c, p)* means the pth pixel of S of the cth 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:

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

Colour Correction

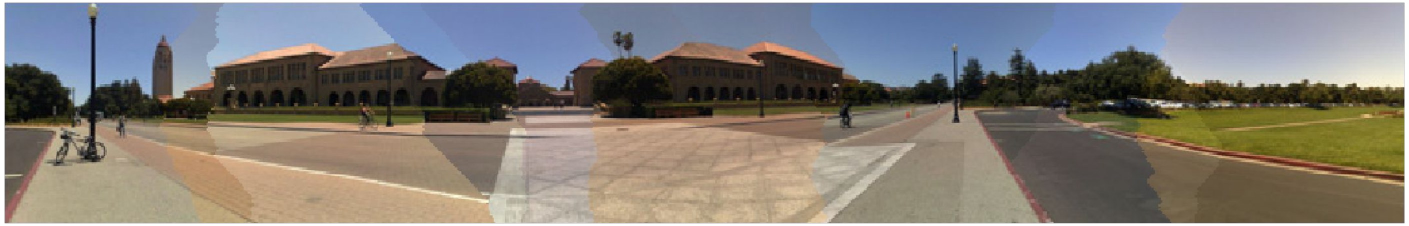
- With these coefficients, we perform colour correction according to the formula -

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

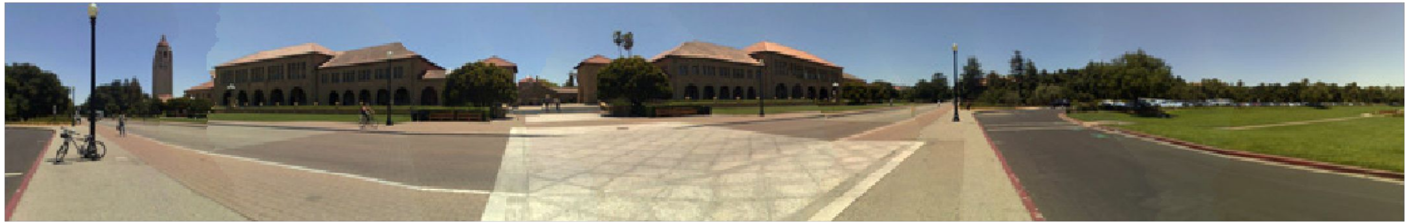
Case to handle - The first image might now always be the ideal image to be used to correct other images, so we first find such an “ideal” image and use it to colour-correct the first image.

Colour Correction - Outputs

Without Colour Correction

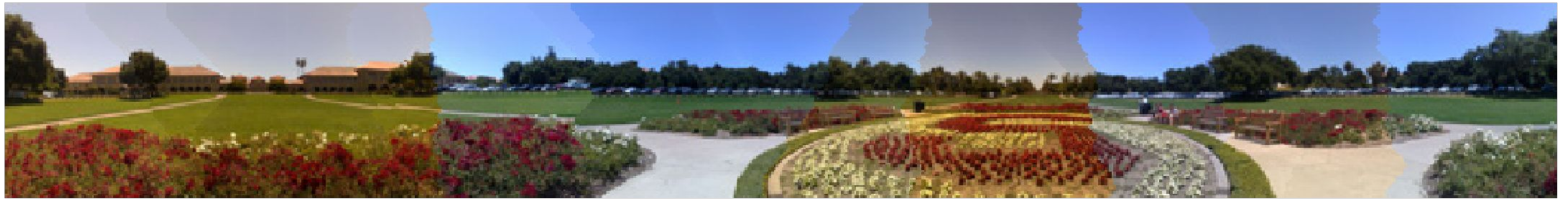


With Colour Correction

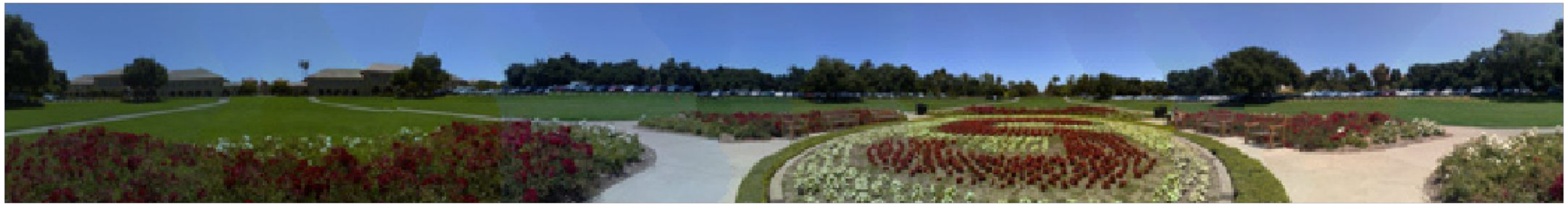


Colour Correction - Outputs

Without Colour Correction



With Colour Correction



Seam Finding

- 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, where ***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 -

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

- Now to find the actual path, we can just start from the bottommost row and trace back until we reach the first row.

Seam Finding - Outputs

Source #1

Source #2

Stitched Image

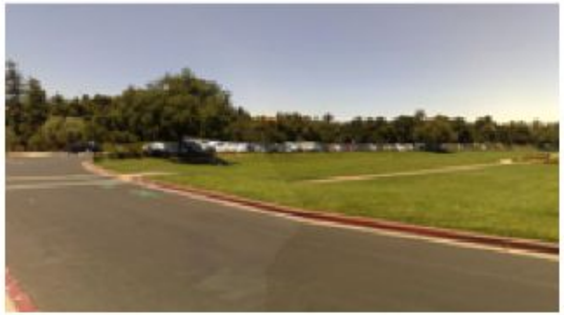


Image Blending

- 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.
- 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 -

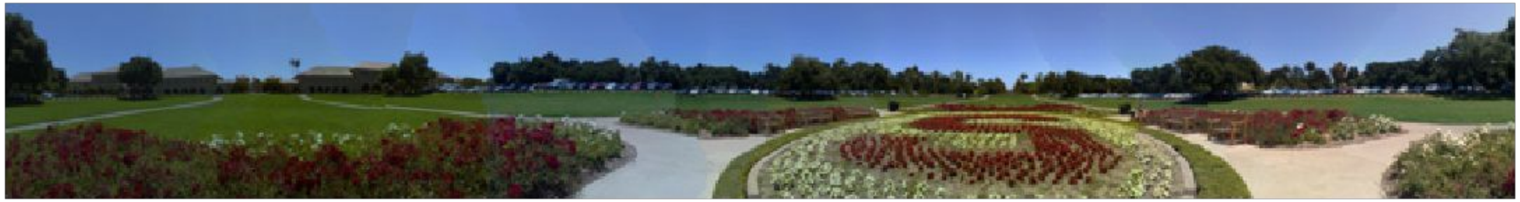
$$P_{I_{c,new}}(p) = \frac{d_1^n P_{I_c}(p) + d_2^n P_{S_c}(p)}{d_1^n + d_2^n},$$

where d_1 and d_2 are the distances from the pixel to the boundaries

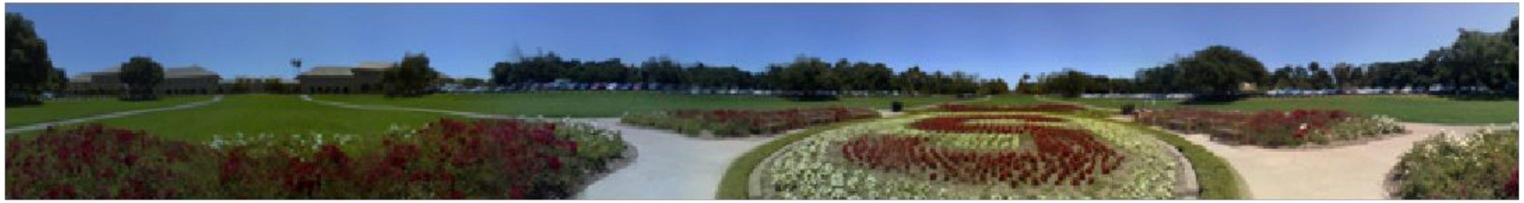
- This is a very simple operation, and thus the memory and computational costs are very low, which is ideal for our purpose.

Image Blending - Outputs

Stitched Image without blending

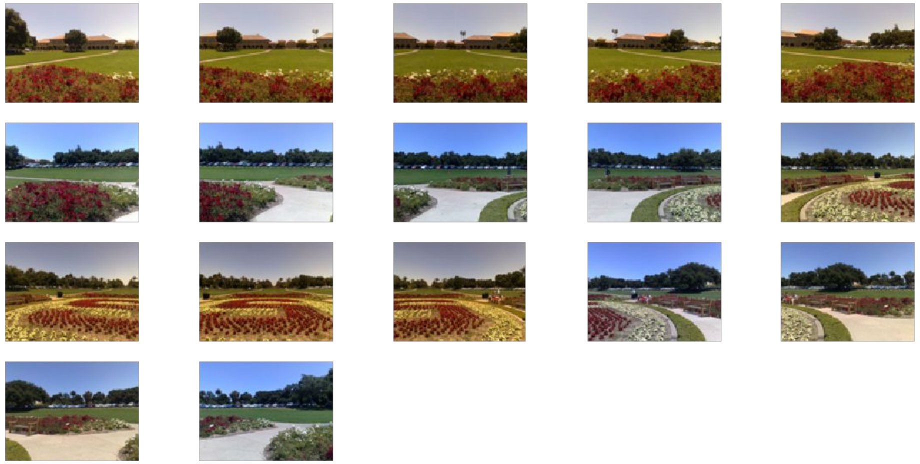


Stitched Image with blending

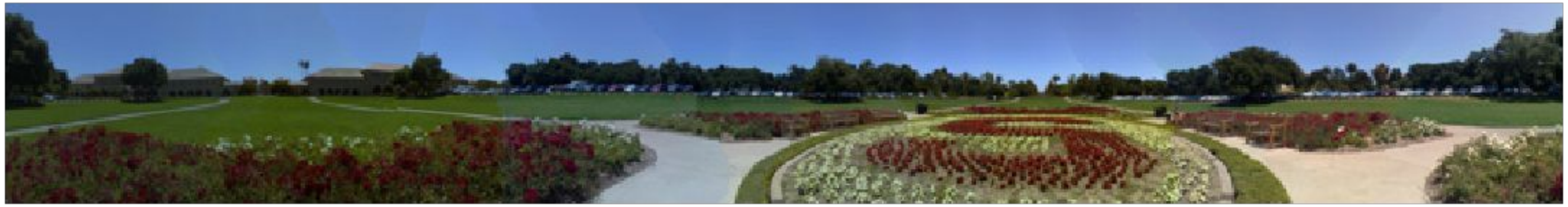


Outputs

Source Images

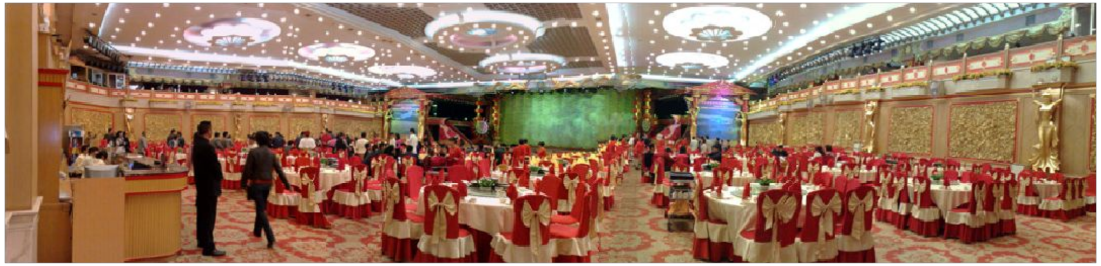


Stitched Image



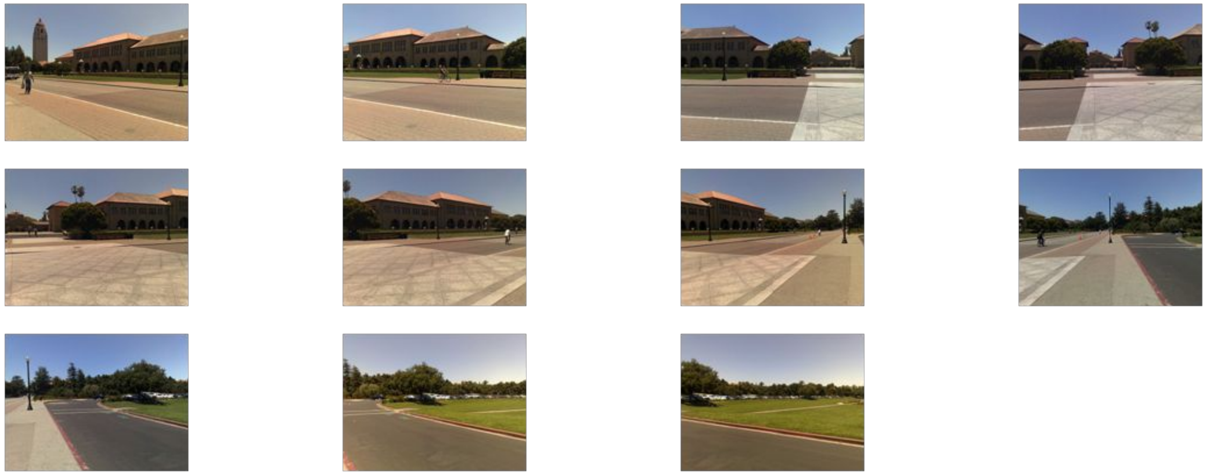
Outputs

Stitched Image

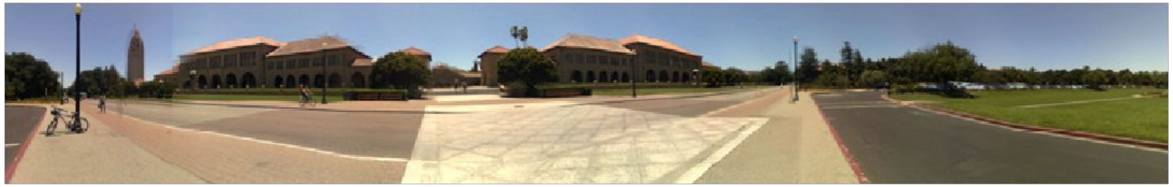


Outputs

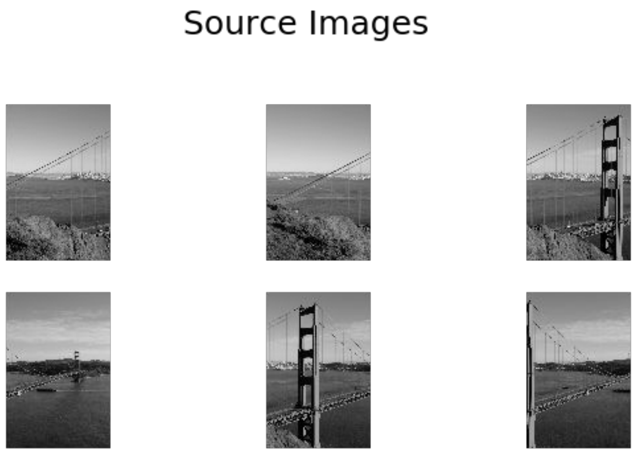
Source Images

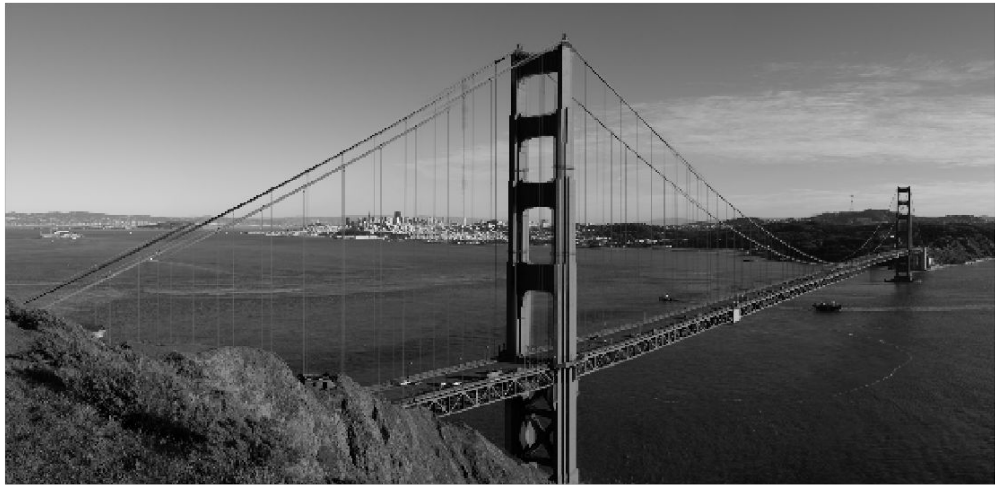


Stitched Image



Outputs





Experiments

- **Calculating 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 as valid values for the overlap and loop through them. For each such we calculate the error function and the required overlap is just the value which gives the lowest error.

  Problem with this?

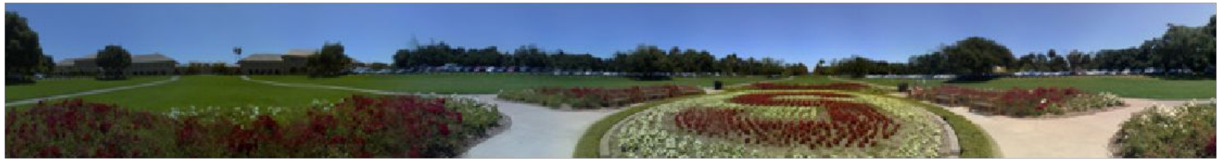
  But prior to seam finding, colour correction is done, which assumes the overlap is provided. So to eliminate that, we can assume 20 - 30% overlap between adjacent images and calculate correction parameters according to that.

Experiments

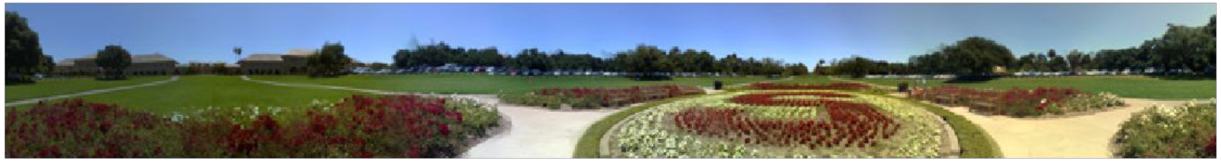
- **“Ideal” Image for correction of first image**

  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.



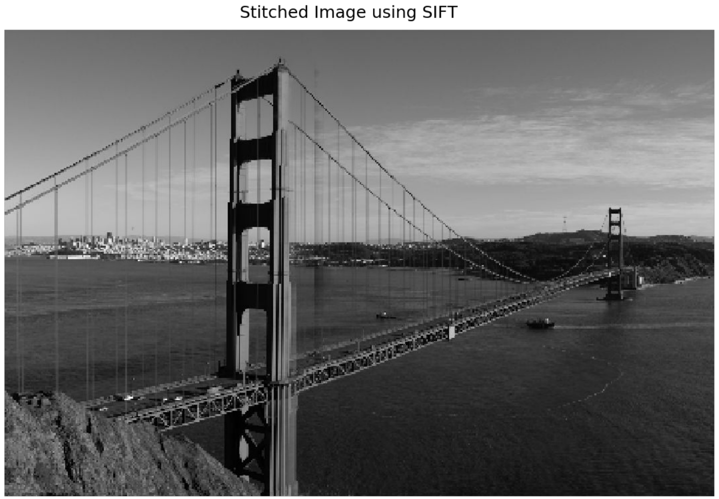
This is the output we get when take the method mentioned in the paper, the best image in this case is the 15th image.

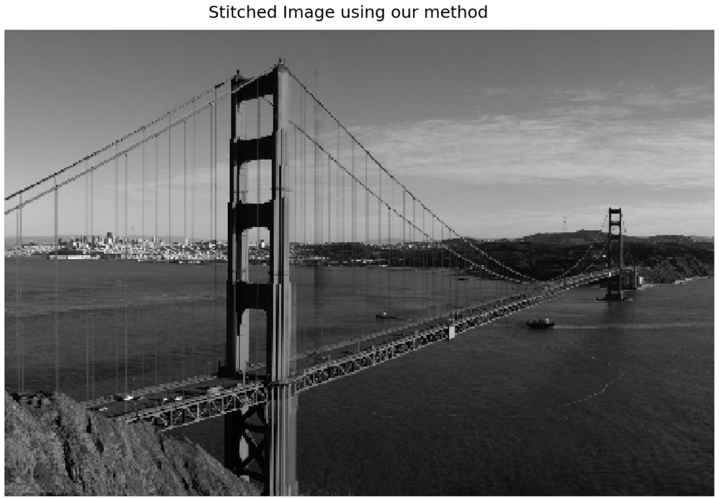


This is the output we get when we maximize the variance instead. The best image in this case is the 6th image.

Experiments

- **Comparison with SIFT**





While the outputs are identical**,** 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 since the number of images to be merged is small, it become significant when we have to run these algorithms on mobiles which have less computing power.

Constraints and Issues Faced

- **Limitations**
  - The algorithm used here assumes the value of the overlap is known.
  - The algorithm doesn’t perform that well when the input images are not aligned properly or are skewed.

- **Issues Faced**
  - The official dataset used in the paper is not available online.