Image mosaicing

Tsneem Basem Omara   
201501278 *s-tsneem.omara@zewailcity.edu.eg*

Mohamed SSayed Abbas   
2015011239 *s-mohamedsayed@zewailcity.edu.eg*

Shady Atef Mohamed   
201500841 *s-shadyatef@zewailcity.edu.eg*

*Abstract*— for this project, we take multiple pictures with overlapping parts to build a panorama with cylindrical mapping.

Keywords—SIFT, RANSAC, mosaic.

# Introduction

In this project, we are taking pictures of the same scene that have overlapping parts, and creating the panorama. The workflow for the image mosaicing includes detecting SIFT features, computing the possible matches of the SIFT features, detecting the best feature matches and the best homography matrix using RANSAC and stitching the images so that the matched points overlap.

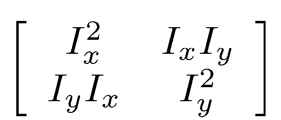
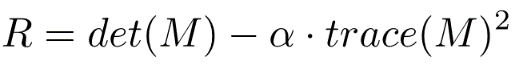
Project steps:

* Feature descriptor
* Matching features
* Computing Homography matrix
* Image stitching
* Image stitching with cylindrical mapping
* Image blinding

Note: We used adobe panoramas pictures dataset.

# Feature descriptor and matching

**Algorithm of Harris corner detector:**

* Convolving the image of sobel filters to get Ix (the image filtered with sobel-x) and Iy(the image filtered with sobel-y)
* Getting ,, and Ixy then convolving them with Gaussian filter
* Find the Harris matrix M
* Getting where alpha = 0.04
* Applying non maximum suppression

**Implementation of Harris corner detector**

The implementation of the first four steps is straight forward. The non-maximum suppression is implemented using ordfilt2 function in MATLAB with radius equals 1.

**SIFT features detector algorithm**

1. Assigning orientations to the key points

* The magnitude and orientation is calculated for all pixels around the keypoint. Then, A histogram is created. In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees). The histogram will have a peak at the dominant orientation
* The window is rotated with the dominant orientation

1. Generating SIFT features

* A window of 16x16 is created around the keypoint. This 16×16 window is broken into sixteen 4×4 windows. A histogram is created for every window after discretizing the angles to 8 bins
* The 16 windows are defined by 128 numbers which are normalized

**Implementation of sift like features**

A for loop that goes through every interest point that does the following

1. Removes the interest points if it is at the edge of the image
2. Convolves the window around the interest point with sobel filters and gets the gradient using imfilter function
3. For loop that goes through the 36 bins, discretize them and finds the dominant direction
4. Rotates the gradient with the dominant direction
5. A For loop that goes through the 16 sub windows and do the follows

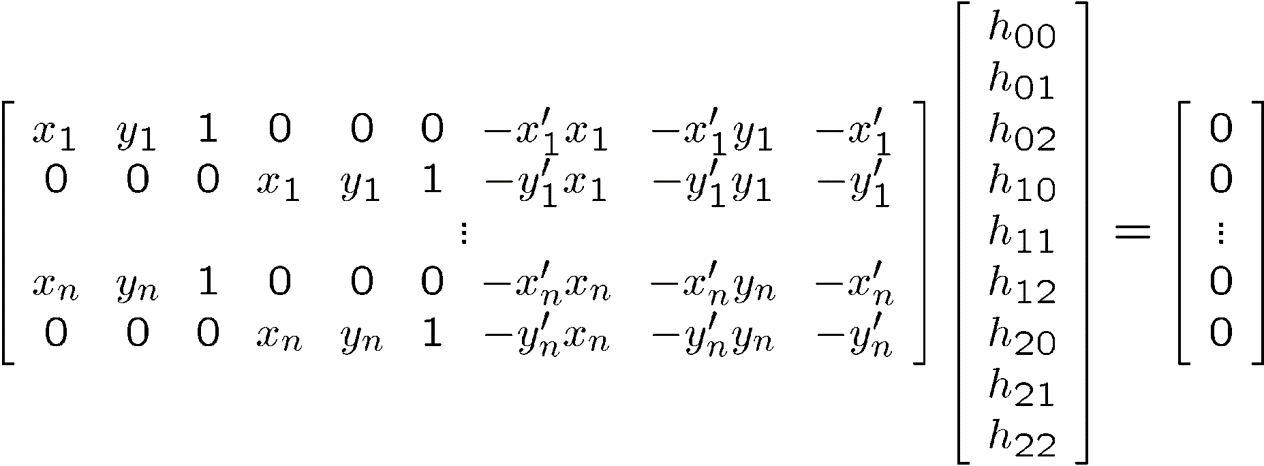
* Find the directions of the sub window and add 360 degrees to the negative ones
* For loop that goes through the 8 bins and creates a histogram for the gradient window

1. Storing the 128 numbers for every interest point

# Computing Homography Matrix

Having the SIFT features and descriptors, we use the Random Sample Consensus (RANSAC) algorithm to determine the transformation matrix (Homography matrix) used to project the first image into the coordinate system of the second one.

First of all, we choose some matched SIFT features at random and use the coordinates of each feature to solve for a homography matrix using the equations below:



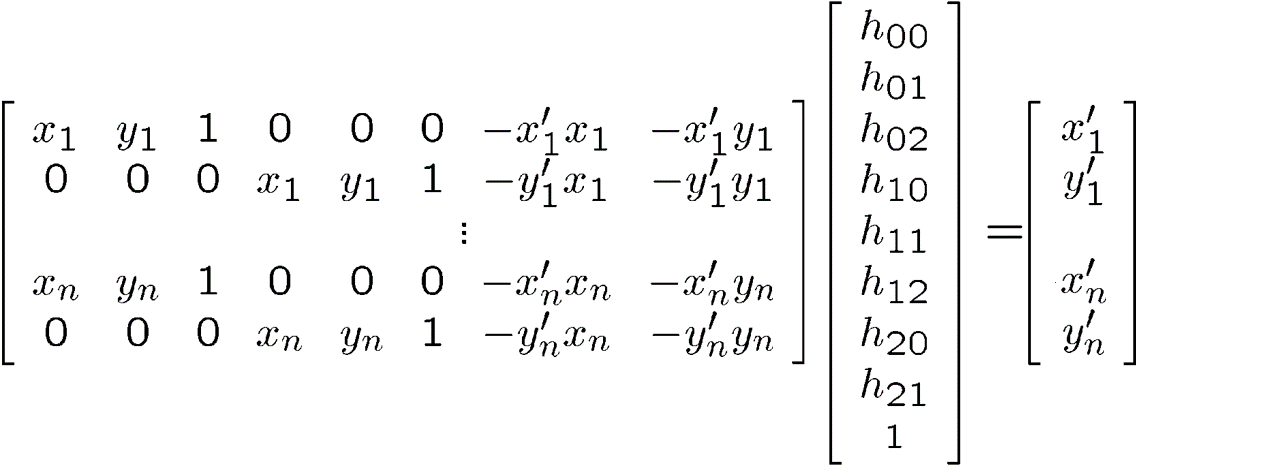
**9**



**2n × 9**



Where x1, y1 and x’1, y’2 are the coordinates of the first and second image …. And h00 to h22 are the elements of the homography matrix. The traditional way is to solve the system by getting h vector equals to the Eigen vector of A’ \* A with smallest eigenvalue. However, we simplified the problem by choosing h22 equals 1 so that the system will be as shown below:



**B**

**9**



**2n × 9**



Then we solve for h=A\B and reshape h to be 3\*3 matrix. For this step, 4 SIFT feature points were enough (this is the minimum) to get good results.

To test the accuracy of the calculated homography, we then iterate over all of the SIFT feature matches projecting the feature coordinates in the first image frame to coordinates in the second image frame using the homography. We then measure the distance between the projected feature coordinates and the actual feature coordinates determined by SIFT, tallying those falling within some threshold, currently **6** pixels.

We then repeat the process, selecting four more random SIFT features, calculating a homography, and tallying inliers. After repeating for a fixed number of iterations, currently **100**, we then keep the homography that resulted in the most number of inliers.

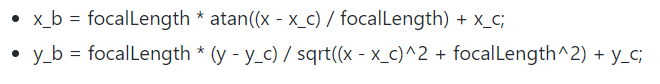
# Image stitching

Once a homography is determined, we then project the second image into the same coordinate system as the first image using the homography and then combine the two images.

But, we have to take into consideration the size inflation the will happen to the transformed image before we stitch it to the second image so that we don’t lose any part of the image. The known way is to find the max (x,y) coordinates of the transformed image and the second image and make a predefined mosaicked image of zero values with the size of maximum coordinates between two images. However, to reduce time and for the sake of simplicity, we pad the first image with zero rows above and below the image with the same number of rows as image two. And add zero columns after the image with the same number of columns as image 2. Then we transform the padded image by multiplying every point with the calculated homography matrix. After that, we overlay the second image over the transformed first image. Finally, we crop the extra padding in the final mosaicked image. We, then repeat this process for the number of images given, using the old mosaicked image as image one for the next iteration.

# image stitching with cylindrical mapping

We used the following equation (1) to transform from the spatial coordinates to cylindrical coordinates to make the cylindrical stitching.



Equation 1: Transforming to cylindrical coordinates.

Assuming a hypothetical cylinder whose radius represents the focal length of the camera, points can be transformed to have a location with respect to this cylinder as illustrated in figure (1) below.

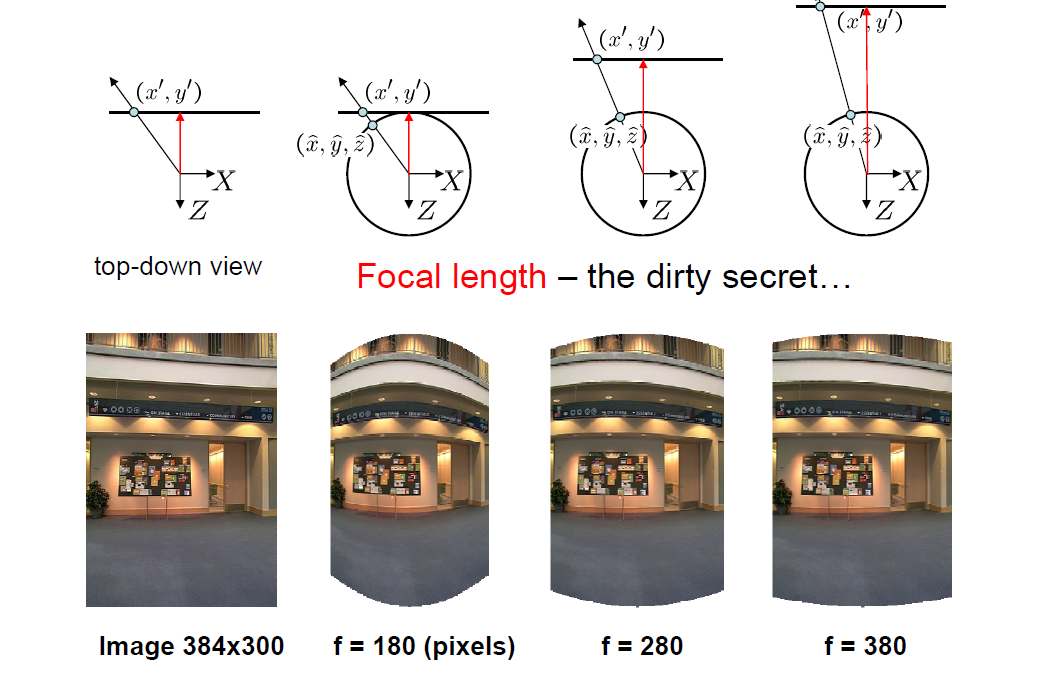
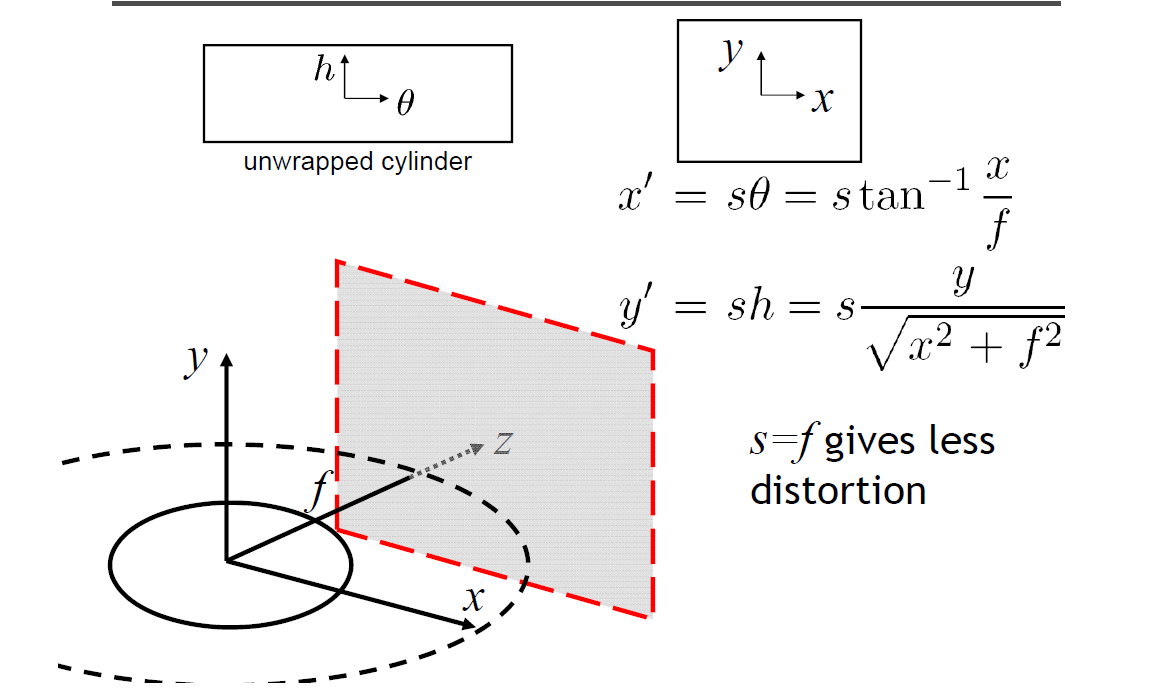


Figure 1: Cylindrical Coordinates Transformation

# Image Blending

After stitching the images together to get the panorama scene, we perform blending on the mutual edges of the images to provide a more comfortable view. This is done through using masks for the separate images and get the intersection of those masks to get the mutual areas of photos and assign weights to the contribution of each mask to get a nice looking. This can be shown in figure (2) below.

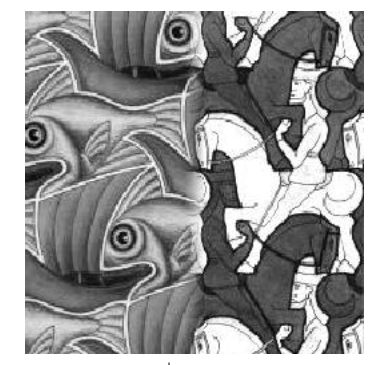
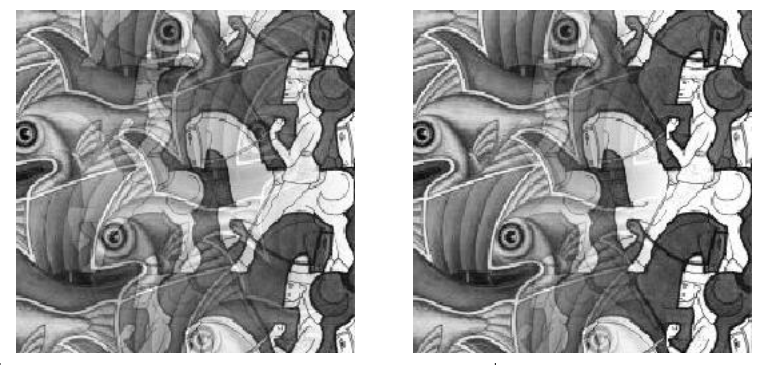


Figure 2: Image Blending