

# Image Mosaicing using MATLAB

Rumel M S Pir  
Assistant Professor  
Leading University, Sylhet Bangladesh  
rumelpir@gmail.com

**Abstract** – In Image processing Mosaicing is one of the technique which is very useful for joining of large digital images. Mosaicing is mixing along of many indiscriminately formed pictures to make one giant radiometrically balanced image so the boundaries between the initial pictures don't seem to be seen. Any range of geocoded pictures will be mixed along on user-specified cut lines (polygons). Mosaicing could be a special case of geometric correction wherever registration takes place within the existing image. In the dark areas of pictures once Mosaicing, the obtained image could be a broad image with high resolution while not mask. However within the case of lighting areas the resultant image is generating mask. so as to gets wide angle panorama, within the existing system, extracting feature points, finding the most effective sewing line Cluster Analysis (CA) and Dynamic Programming (DP) ways area unit used. Image Mosaicing algorithms essentially divided in to five phases. Which includes Feature point extraction, Image registration, Homography computation, Warping and Blending if Image. Various corner detection algorithms is being used for Feature extraction

**Index Terms** – Image Mosaicing, resolution, Photogrammetric

## I. INTRODUCTION

Image mosaics are collections of overlapping images that are transformed in order to result in a complete image of a wide angle scene. The transformations can be viewed as simple relations between coordinate systems. Stitching together images taken from different viewpoints seamlessly Representation: cylindrical, spherical, and perspective panoramas. Mosaic is the art of creating images with an assemblage of small pieces of colored glass, stone, or other materials. It may be a technique of decorative art, an aspect of interior decoration, or of cultural and spiritual significance as in a cathedral. Small pieces, normally roughly quadratic, of stone or glass of different colors, known as tesserae, (diminutive tessellae), are used to create a pattern or picture.

An Image Mosaicing operation consists of three main steps. The first one is Image Registration. In this phase the correspondence between features common to both images is defined in order to compute an image-to-image transformation that maps one image into the other. In this work we selected the features to be the corners in the images and we used the Laganière method [1] for their detection. In the next step, Image Warping, the two images that will form the mosaic are warped, by using the geometric transformation computed in the previous phase. The final stage, Image Compositing, consists of blending images together on the reference coordinate system.

Why there is need of image Mosaicing?

Are you getting the whole picture?

Compact Camera FOV =  $50 \times 35^\circ$

Human FOV =  $200 \times 135^\circ$

Panoramic Mosaic =  $360 \times 180^\circ$

This is the camera view:

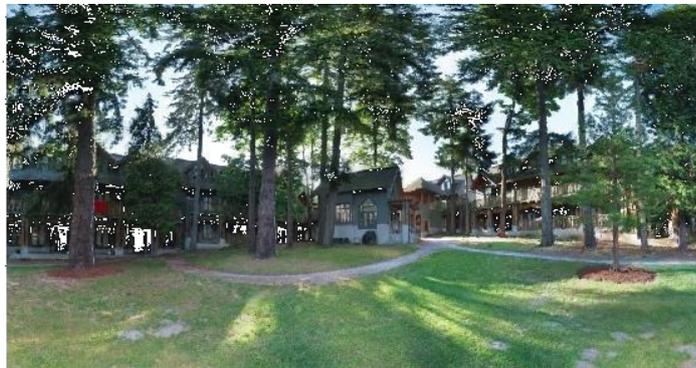


Fig-1 camera view of Image

This is the panorama view at 360:



Fig-2 the panorama view of Image at 360

We cannot get full 360 degree view by human eye or camera. So image Mosaicing help us to joint more than one images and we will get full view. Image Mosaicing help us to generate the full view without any type of noise.

## II. PROBLEMS IN IMAGE MOSAICING[5]

1. Global Alignment: involves calculation of the transform (homography), which aligns two images.
2. Local Adjustment: Even after good global alignment, some pixel might not align in the two images. This might cause ghosting or blur in the blended image.
3. Image Blending: deciding the color the overlapping region should take.
4. Automatic selection of images to blend.
5. Auto-exposure compensation: Variable brightness in the overlapping region might cause the mosaic to look unrealistic.
6. Camera error compensation.

## III. APPLICATION[4]

3 dimensional vision

- Photogrammetric
  - Satellite imagery
  - Video images
- Motion or change detection etc.

The basic reason for using Image Mosaicing is that camera vision is limited to 50 by 35 degrees, while Human Vision is limited to 200 degrees.

## IV. METHODS OF IMAGE MOSAICING[2]

4.1 Mosaicing methods can be classified broadly into:

1. Direct Methods
2. Feature Based Method

4.1.1 Direct Methods: Uses information from all pixels. Iteratively updates an estimate of homography so that a particular cost function is minimized. Sometimes Phase-Correlation is used to estimate the a few parameters of the homography.

4.1.2 Feature Based Method: To locate common points of interest in the images, we use a generalization of the gray-image Harris corner-detector to color data. Given a matrix M:

$$M = \begin{bmatrix} s_{\sigma}\{R_x^2 + G_x^2 + B_x^2\} & s_{\sigma}\{R_x R_y + G_x G_y + B_x B_y\} \\ s_{\sigma}\{R_x R_y + G_x G_y + B_x B_y\} & s_{\sigma}\{R_y^2 + G_y^2 + B_y^2\} \end{bmatrix}$$

$$R_x(x, y) = \frac{\partial R(x, y)}{\partial x}, R_y(x, y) = \frac{\partial R(x, y)}{\partial y}$$

where R, G and B are the red, green and blue pixel value functions of (x,y) and  $S_{\sigma}$  is an isotropic gaussian filter, the feature points are given by the positive local maxima of the function.

$$\det M - K(\text{trace } M)^2 \quad \text{with } K = 0.04$$

The value of the local color gradients used to compute M are estimated using the Deriche filter, which has low sensitivity to noise. The following figures show stages of the feature extraction pipeline.

4.2 Image Mosaicing process can be divided in to two main components:

1. Image Registration
2. Image Composing /Blending

4.2.1 Image Registration:

Image registration is the process of transforming the different sets of data into one coordinate system. Registration is necessary in order to be able to compare or integrate the data obtained from different measurements.

4.2.1.1 Various Techniques of Image Registration

1. Frequency domain phase-correlation method
2. Feature-based registration technique
3. Intensity Based Image Mosaicing

1. Frequency domain phase-correlation method[2]

Consider two images  $f_1(x,y)$  and  $f_2(x,y)$  that are related by a simple shift in the spatial domain:

$$f_1(x, y) = f_2(x - x_0, y - y_0)$$

Their corresponding Fourier transforms will then be related by

$$F_2(\omega_x, \omega_y) = e^{-j(\omega_x x_0 + \omega_y y_0)} F_1(\omega_x, \omega_y)$$

When taking the normalized cross-power spectrum of the two images as follows, a simple complex exponential results:

$$\frac{F_1^*(\omega_x, \omega_y) F_2(\omega_x, \omega_y)}{|F_1(\omega_x, \omega_y) F_2(\omega_x, \omega_y)|} = e^{-j(\omega_x x_0 + \omega_y y_0)} \xleftrightarrow{IFT} \delta(x - x_0, y - y_0)$$

The inverse Fourier transform of this is a Dirac pulse at the coordinates of the translational movement. One has to exhaustively search for this peak to find the displacement.

Problem: Unable to distinguish between negative shifts and shifts larger than  $N/2$  (where N is the image size in one dimension).

$$f'(n) = f(n + n_0) \Rightarrow S_{f,f'}(k) = e^{j2\pi \frac{k}{N} n_0} \xleftrightarrow{IDFT} \delta(n - N + n_0)$$

$$f'(n) = f(n - N + n_0) \Rightarrow S_{f,f'}(k) = e^{j2\pi \frac{k}{N} (-N + n_0)} = e^{j2\pi \frac{k}{N} n_0} \xleftrightarrow{IDFT} \delta(n - N + n_0)$$

n :spatial index, k: frequency index

Solution: Double the FFT sizes.

2. Feature-based registration technique[2]

To locate common points of interest in the images, we use a generalization of the gray-image Harris corner-detector to color data. Given a matrix M:

$$M = \begin{bmatrix} s_{\sigma}\{R_x^2 + G_x^2 + B_x^2\} & s_{\sigma}\{R_xR_y + G_xG_y + B_xB_y\} \\ s_{\sigma}\{R_xR_y + G_xG_y + B_xB_y\} & s_{\sigma}\{R_y^2 + G_y^2 + B_y^2\} \end{bmatrix}$$

$$R_x(x, y) = \frac{\partial R(x, y)}{\partial x}, R_y(x, y) = \frac{\partial R(x, y)}{\partial y}$$

Where R, G and B are the red, green and blue pixel value functions of (x,y) and  $s_{\sigma}$  is an isotropic gaussian filter, the feature points are given by the positive local maxima of the function.

$$|\det M - K(\text{trace } M)^2| \quad \text{with } K = 0.04$$

The value of the local color gradients used to compute M are estimated using the Deriche filter, which has low sensitivity to noise. The following figures show stages of the feature extraction pipeline.

3. Feature point extraction



Fig-3 Original Image

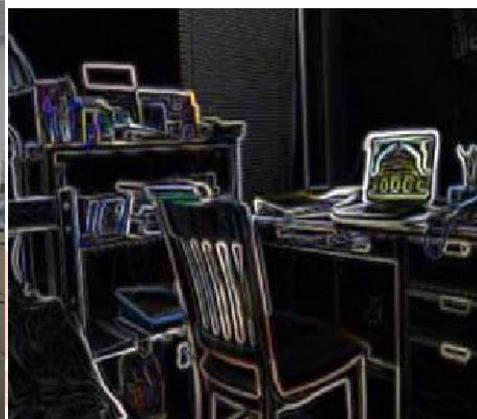


Fig-4 Deriche

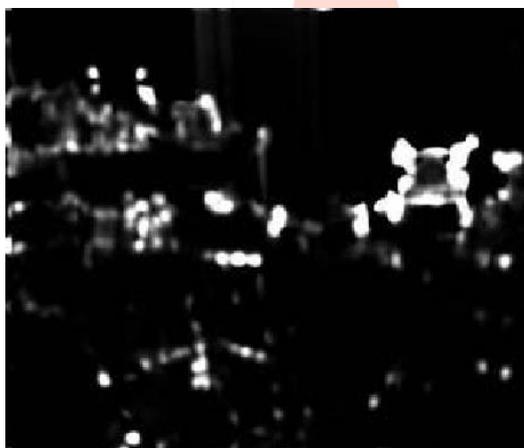


Fig-5 Hariss Function



Fig-6 Extracted Features

To get an approximate registration of the images, we proceed as follows:

- Select the N "best" features points from both images.
- Compute all NxN matching grades, local rotation angles and local scale factors.(polar log parameterization)
- Build a pair set S by keeping only the matches over two given threshold, one absolute, and the other relative to the best match of each feature.
- For all possible couple of pairs from S: compute a 3x3 homogeneous matrix from the coordinates of the 4 points, the 2 scaling factors and the 2 rotation angles.
- compute a global registration grade using a linear regression correlation
- measure applied to the overlapping area pixels.

**Futures Point Matching**



Figure 5: Features in the left image

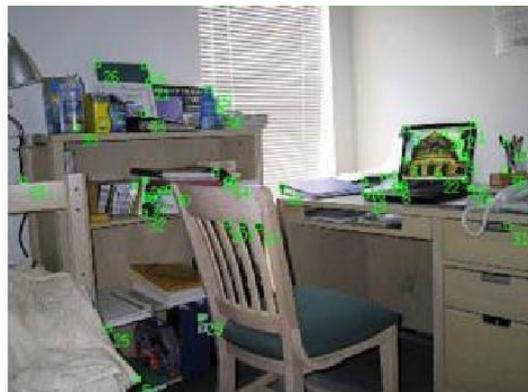


Figure 6: Features in the right image

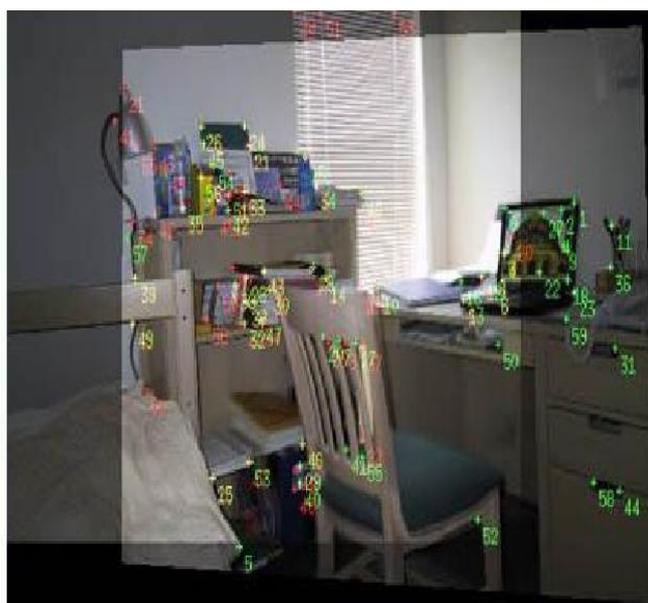


Figure 7: Resulting registration

Intensity Based Image Mosaicing[2]

1. for each pixel at (x, y) in the first image, we compute the corresponding pixel in the second image.

$$x' = \frac{m_0x + m_1y + m_2}{m_6x + m_7y + 1}$$

$$y' = \frac{m_3x + m_4y + m_5}{m_6x + m_7y + 1}$$

2. Compute the error in intensity between the corresponding pixels using[2]

$$E = \sum_i [I'(x'_i, y'_i) - I(x_i, y_i)]^2 = \sum_i e_i^2$$

3. The problem can be solved by an iterative algorithm.
  - Alternatively use the Levenberg Marquardt algorithm
  - Robust
  - result in about 20 iterations.

Global Registration[2]

We have possibly  $\binom{n}{2}$  different registered image pairs.

$$P_i = A_{ij}P_j \quad \forall i, j$$

- With P absolute rotation matrices, A relative rotation matrix
- Solve in a least squares sense to yield optimum solution:

$$\underbrace{\begin{bmatrix} \mu_1 \mathbf{I} \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}}_{\mathbf{L}} = \underbrace{\begin{bmatrix} \mu_1 \mathbf{I} & 0 & 0 & \dots & 0 \\ \mu_2 \mathbf{A}_{21} & -\mu_2 \mathbf{I} & 0 & \dots & 0 \\ 0 & \mu_3 \mathbf{A}_{32} & -\mu_3 \mathbf{I} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \vdots \\ \mathbf{P}_n \end{bmatrix}}_{\mathbf{P}}$$

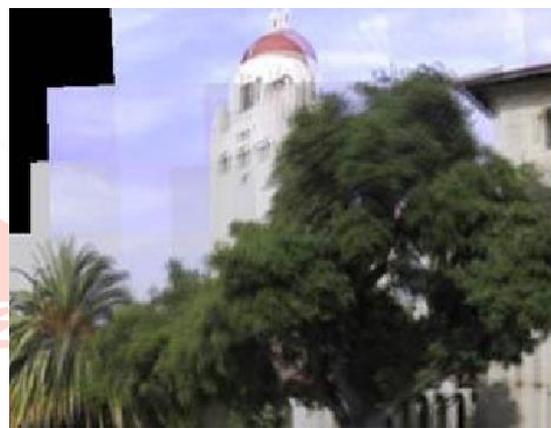
$$\Rightarrow \mathbf{P}_{LS} = \mathbf{A}^\dagger \mathbf{L} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{L}$$

Weighing each pair by its matching factor  $\mu$ . Benefit : pair wise registrations that have a small matching factor (which means that they are either a bogus registration or correspond to an error- prone small overlap) are given less influence in the solution.

Examples:



Positioning by linear pair wise traverse



Positioning by global registration

#### 4.2.2 Image Composing /Blending[1]

Once we know each absolute image's rotation matrix, we can project on arbitrary geometries:

- Planar



- Spherical



- Angular



- Problem: Smoothly blend over between images to hide seams
- Rather complicated math
- Instead: Use simple heuristic
- Every pixel is weighted with the distance to the closest image boundary to the nth power.

$$f_{res}(P) = \frac{\sum_{i=1}^M f_i(P) d_i^n}{\sum_{i=1}^M d_i^n}$$

- M - Number of overlapping images and d - the distance to a source image's closest edge.
- In practice, a value for n between 3 and 4 has proven to yield the best results.

**Intensity Based Approach[2]**

In order for the edges not to be visible, we want the intensities of both images to be weighted less as the pixel locations get closer to its edges. That way, when an averaged sum is computed to find out the final intensity of the image as follows.

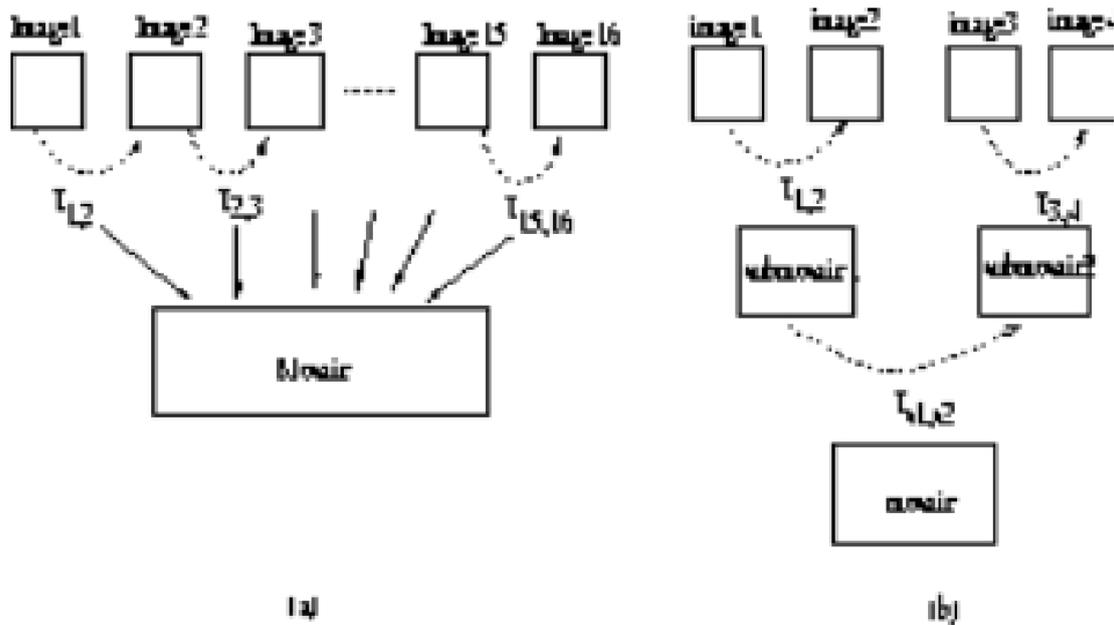
$$Intensity = \frac{(W1 * I1 + W2 * I2)}{(W1 + W2)}$$



Composing without blending

Composing with blending

**Tree Based Mosaicing[2]**



- Homography relating the two images was estimated using Random Sample Consensus (RANSAC). RANSAC implementation is gets a remarkable improvement over Least Squares Estimation of Homography.
- Phase-correlation proved to be fast and reliable. However it appears to be limited to cases where the rotation and perspective distortions are small enough.
- Feature-based method is capable of handling any kind of projective transforms, so it can register wide-angle shots. It can find a rough registration even if the overlap is small. But it is likely to fail in the absence of distinctive features (as in sky images), and is sensitive to noise and moving objects.
- It also appeared to be significantly slower than phase-correlation.

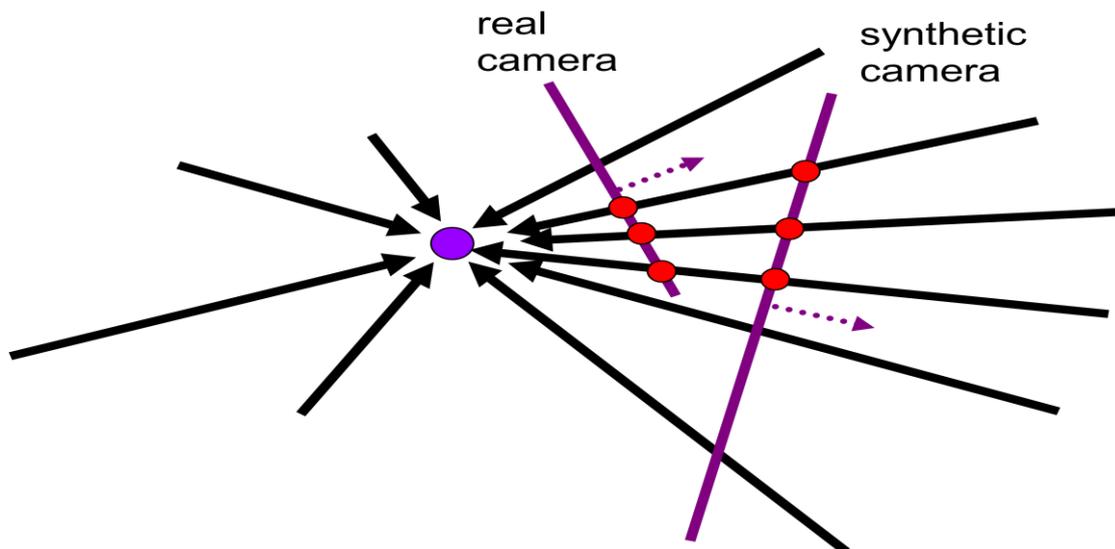
**V. MOSAIC PROCEDURE[2]**

**4.1 Basic Procedure**

- Take a sequence of images from the same position
- Rotate the camera about its optical center
- Compute transformation between second image and first
- Transform the second image to overlap with the first
- Blend the two together to create a mosaic
- If there are more images, repeat

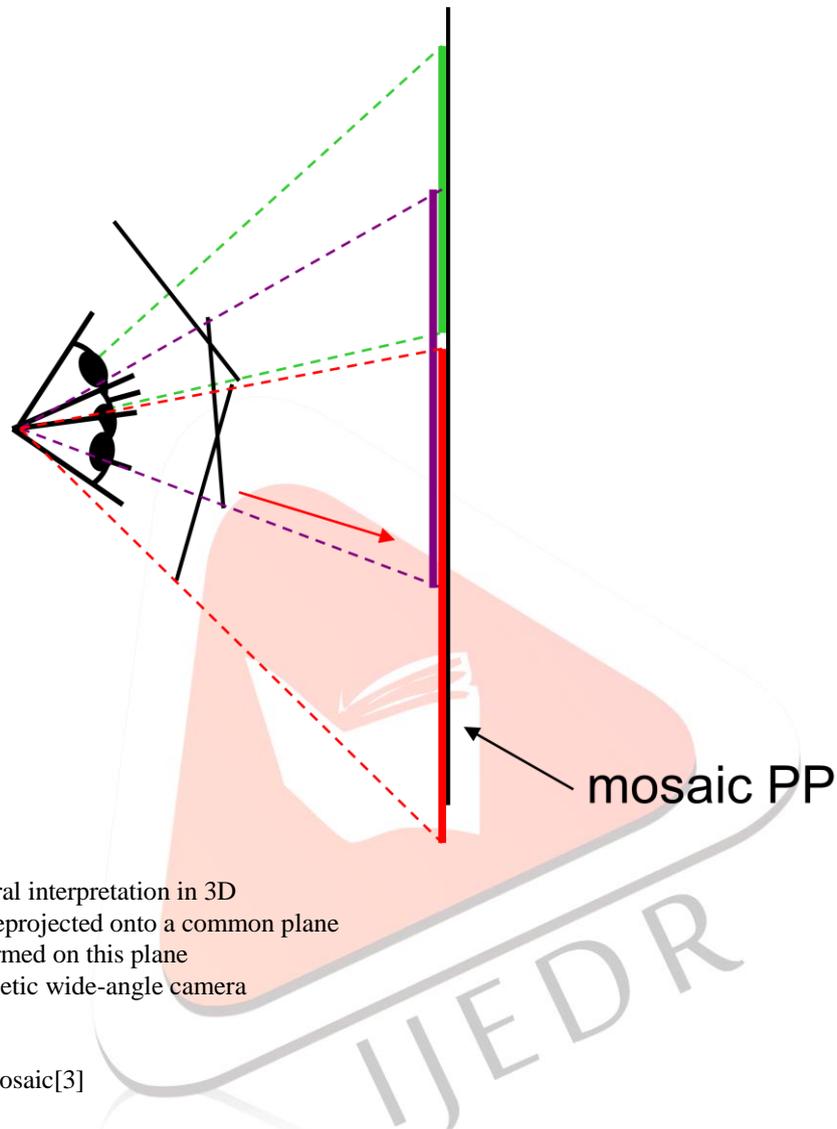
**4.2 Image Mosaic**

Is a pencil of rays contains all views



Can generate any synthetic camera view as long as it has the same center of projection!

#### 4.3 Image Re-projection[2]

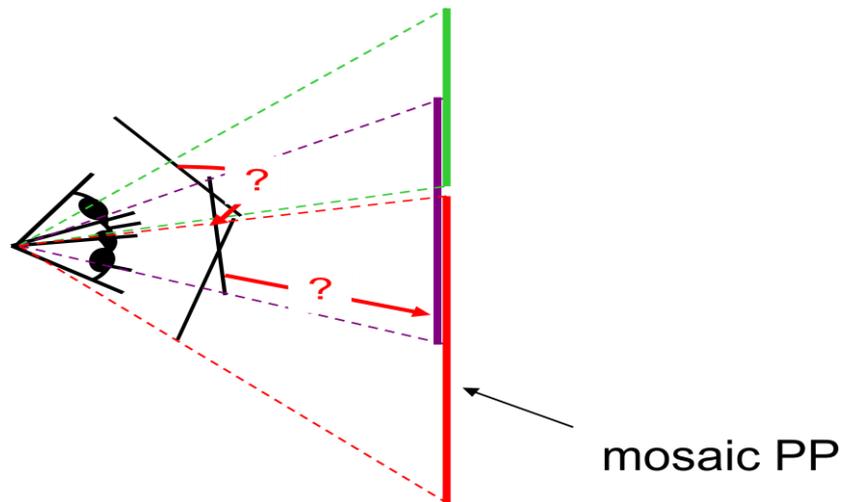


The mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a synthetic wide-angle camera

#### 4.4 Issues in Image Mosaic[3]

- How to relate two images from the same camera center?
  - Image registration
- How to re-project images to a common plane?
  - Image warping



- Geometric relationship between images
  - Use 8-parameter projective transformation matrix

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Use a 3D rotation model (one R per image)

$$\begin{pmatrix} r_{00} & r_{01} & f_0 r_{02} \\ r_{10} & r_{11} & f_0 r_{12} \\ r_{20}/f_1 & r_{21}/f_1 & f_0 r_{21}/f_1 \end{pmatrix}$$

- Register *all* pair wise overlapping images
  - Feature-based registration
  - Pixel-based registration
- *Chain* together inter-frame rotations
- Image Stitching
  - Stitch pairs together, blend, then crop

## VI. IMPLEMENTATION

### About Matlab

MATLAB (matrix laboratory) is a numerical computing environment and fourth-generation programming language. Developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

### Code(Matlab)

#### Mosaic.m

```
% load input images
I1 = double(imread('left.jpg'));
[h1 w1 d1] = size(I1);
I2 = double(imread('right.jpg'));
[h2 w2 d2] = size(I2);

% show input images and prompt for correspondences
figure; subplot(1,2,1); image(I1/255); axis image; hold on;
title('first input image');
[X1 Y1] = ginput2(2); % get two points from the user
subplot(1,2,2); image(I2/255); axis image; hold on;
title('second input image');
```

```

[X2 Y2] = ginput2(2); % get two points from the user

% estimate parameter vector (t)
Z = [ X2' Y2' ; Y2' -X2' ; 1 1 0 0 ; 0 0 1 1 ]';
xp = [ X1 ; Y1 ];
t = Z \ xp; % solve the linear system
a = t(1); % = s cos(alpha)
b = t(2); % = s sin(alpha)
tx = t(3);
ty = t(4);

% construct transformation matrix (T)
T = [ a b tx ; -b a ty ; 0 0 1];

% warp incoming corners to determine the size of the output image (in to out)
cp = T*[ 1 1 w2 w2 ; 1 h2 1 h2 ; 1 1 1 1 ];
Xpr = min( [ cp(1,:) 0 ] ) : max( [ cp(1,:) w1 ] ); % min x : max x
Ypr = min( [ cp(2,:) 0 ] ) : max( [ cp(2,:) h1 ] ); % min y : max y
[Xp,Yp] = ndgrid(Xpr,Ypr);
[wp hp] = size(Xp); % = size(Yp)

% do backwards transform (from out to in)
X = T \ [ Xp(:) Yp(:) ones(wp*hp,1) ]'; % warp

% re-sample pixel values with bilinear interpolation
clear Ip;
xI = reshape( X(1,:),wp,hp)';
yI = reshape( X(2,:),wp,hp)';
Ip(:,1) = interp2(I2(:,1), xI, yI, '*bilinear'); % red
Ip(:,2) = interp2(I2(:,2), xI, yI, '*bilinear'); % green
Ip(:,3) = interp2(I2(:,3), xI, yI, '*bilinear'); % blue

% offset and copy original image into the warped image
offset = -round( [ min( [ cp(1,:) 0 ] ) min( [ cp(2,:) 0 ] ) ] );
Ip(1+offset(2):h1+offset(2),1+offset(1):w1+offset(1),:) = double(I1(1:h1,1:w1,:));

% show the result
figure; image(Ip/255); axis image;
title('mosaic image');

```

### Code Description

The mosaic.m script operates as follows:

First, it converts the values of the images from unsigned int to double. This is done because most of the MATLAB operations use double as default type. Then, it asks for the points to be matched. Once it has this information it performs a transformation based on an estimate. The value of each pixel is then chosen using a bilinear interpolation. Finally, the mosaicked image is displayed.

### Ginput.m

```

function [X, Y] = ginput(n)

X = [];
Y = [];
for i=1:n,
    [px py] = ginput(1);
    X = [X; px];
    Y = [Y; py];
    plot(px, py, 'r+');
    text_handle = text(px+5, py, num2str(i));
    set(text_handle,'Color',[1 1 1])
end

```

### Steps to run this program:

1. Open Matlab program on the personal computer you are working on.
2. Create the mosaic.m by editor.
3. Create the ginput.m by editor.
4. Save two different images called as left and right in jpeg format.
5. Execute Mosaic.m file.
6. You will get the output window.
7. A new window will appear entitled 'Figure No. 1'. Select two points, in the first image, that would match two other points on the second image. Red crosses would mark the desired points.
8. Use the mouse again to click on the second image. Be sure to select the points on the same order you did the first ones.
9. Review the mosaicked image in Figure No. 2. If the image is distorted, at odd angles, or it is not satisfactory in general, run the script one more time.
10. Go to File > Save As, and save the image as \*.jpg

## VII. CONCLUSION

Image Mosaicing is the technique which used to stitch the more than two images which taken from the different angles. We can take the efficient panoramas. Moving objects are removed by this Mosaicing. There are different techniques which mosaic the images. We can project on arbitrary geometries like angular, spherical and planar. Images from the different angles can also implement. Images with different project point cannot be wrapped together.

## VIII. REFERENCES

- [1] Satya Prakash Mallick *Feature Based Image Mosaicing*, Department of Electrical and Computer Engineering, University of California, San Diego
- [2] Juan Zhang, Lili Sang and Yunjun Liu *Image Mosaics*[2004]
- [3] A. Rav-Acha, Y. Pritch, D. Lischinski, & S. Peleg, "Dynamosaicing: Mosaicing of Dynamic Scenes", IEEE TRAN ON PAMI, VOL. 29, NO. 10, 2007
- [4] Kantilal P. Rane, S. G. Bhirud, "Image Mosaicing with Strip Search Algorithm based on Novel Similarity Measure," International Journal of computer application, vol. 27, no. 2, Aug. 2011.
- [5] Shum, H., Szeliski, R., "Construction and refinement of panoramic mosaics with global and local alignment," IEEE Int'l Conf. Computer Vision, 1998, pp: 953-958.
- [6] Steven L. Kilthau, Mark S. Drew, Torsten Moller, "Full Search Content Independent Block Matching Based on the Fast Fourier Transform," IEEE International Conference on Image Processing, pp: 669-672, 2002.