A Fast Method for Image Mosaicing using Geometric Hashing

Udhav Bhosle Subhasis Chaudhuri Sumantra Dutta Roy
Department of Electrical Engineering,
Indian Institute of Technology Bombay, Powai, Mumbai-400706.
{udhav,sc,sumantra}@ee.iitb.ac.in

Abstract

The general problem of mosaicing is to create a single seamless image by aligning a series of spatially overlapped images. The result is an image with a field of view greater than that of a single image. Traditionally this research has been aimed at stitching together images taken by aerial or satellite reconnaissance equipment. With the advancement of personal computing equipment, the creation of image mosaic has entered the consumer market. Thus, automation of the process is an important issue. This paper proposes a new method for automatic generation of mosaics using Geometric Hashing. This speeds up the matching process. We show the application of our method on two important cases namely, rigid planar motion and panoramic mosaics. We provide experimental results in support of our proposed method.

1 Introduction

The automatic construction of large high resolution image mosaics is an active area of research in the field of computer vision, image processing and computer graphics. Image mosaicing is commonly used to increase the visual field of view by pasting together many video frames. The camera's field of view is always smaller than the human field of view. Further large objects often cannot be captured in a single picture as is the case in aerial photography. Using a lens having a wider field of view (fish eye lens) can be a partial solution, but the images obtained with such a lens have substantial distortion, further, capturing the entire scene with a limited resolution of video camera compromises image quality. Panoramic mosaics can be created by special devices such as quick time VR, surround video, which move around the camera optical centre. However, it has strong limitations on the imaging conditions. A common solution is photo saucing: aligning and pasting frames in video sequences, which enables a more complete view [12, 8].

Three major issue are important in image mosaicing:

1. *Image alignment:* Determines the transformations that align images to be combined into a mosaic. This may be Euclidean (Rigid body) transformation, a similarity transformation, affine or, in the most general case, projective transformation.(see the Appendix for a brief description).

   Image registration or image alignment is a fundamental task in image processing to overlay two or more images used. Registration methods can be loosely divided into following classes.

   (a) algorithms that use image pixel values directly *i.e.*, correlation method [4].

   (b) algorithms that use frequency domain method *i.e.*, Fast Fourier transform based methods (FFT) [10]

   (c) algorithms that use low level feature such as edges and corners *i.e.*, feature based method [15].

   (d) algorithms that use high level features such as identified (parts of) object or a relation between features *i.e.*, graph theoretic methods [10].

2. *Image cut and paste:* Image mosaicing involves a combination of images which have overlapping regions. The cut and paste process involves selecting this region in mosaics. There are two ways to determine this region.

   (a) Using colour/gray scale information from all constituent images for the region of overlap (median, average, etc.)

   (b) Selecting a region from one of images.

   Method (a) requires accurate alignment over the entire image area, otherwise resulting mosaic will be blurred. The method (b) requires alignment only along the seams. This is more useful in cases where camera motion, scene geometry and imaging condition are challenging [7, 14].

3. *Image blending :* It is used to overcome the intensity difference between the images, differences that are present even when images are perfectly aligned. These are created by dynamically changing camera gain [8, 7].

In this article, we deal with 2-D mosaics. We consider two cases. The first is mosaics for planar rigid camera motion and second is panoramic mosaics. A automation of the process is an important issue. We propose a new method for automatic generation of mosaics using geometric hashing. We use a feature based method for image registration. Matching features across images has exponential time complexity. We reduce this to the polynomial-time. This speed up the matching process in addition to automating it.

The simplest mosaics are created from a set of images whose mutual displacement are pure image plane translation. This is approximately the case with satellite images. Such translation can either be computed by manually pointing to corresponding points or by an image correlation method. Other simple mosaics are created by rotating the camera about its optical center, using a special device and creating a panoramic image, which represent the projection of the scene onto a cylinder [12, 8]. Since it is not simple to ensure a pure rotation around the optical center, such mosaics are used only in limited cases. In more general camera motion (that includes both camera translation and camera rotation), more general transformation for image alignment are used [8, 7].

Some efficient methods have been developed to build mosaic, when homography is mainly translation. For example, if the overlap between the images is very large (*i.e.* the motion is very small), it has been shown that the Levenberg Marquardt method yields good result [12], but it is very sensitive to local minima and computationally expensive. In another case, when the overlapping is smaller, we can use a hierarchical matching to avoid local minima. For large camera motion the phase correlation method has been used [1].

A less hardware intensive method for constructing full view panoramas is to take many regular photographs or video images in order to cover the whole viewing space. These images must then be aligned and composited into complete panoramic images using an image mosaic or stitching algorithms [3, 12]. Most stitching systems requires a carefully controlled camera motion, and only produce cylindrical images. In this paper, we don't make any restrictive assumption on the specific camera movement, given a particular imaging setup.

In all cases images are aligned pairwise, using a parametric transformation like an affine transformation or planar projective transformation. A reference frame is selected, all images are aligned with this reference frame, and are combined to create mosaics. Aligning all frames to a single reference frame is reasonable when camera is far away and its motion is mainly translation and rotation around the optical axis. Significant distortions are created when camera motion include other rotation [8].

The rest of the paper is organized as follow. Section 2 describes Geometric Hashing. We discuss two important classes of motion namely, for planar rigid camera motion and panoramic mosaics, in Section 3 and Section 4, respectively. We conclude the paper in Section 5.

2 Geometric Hashing

Image alignment requires matching M points in one image with N points in another. As such, this process has an exponential time complexity, $\mathcal{O}(M^N)$. Lamdan *et al.* [6] propose geometric hashing as a fast method for 2-D object recognition using an affine assumption where M object points are to be matched to N image points, We generalize this idea for image alignment (the first step in image mosaicing), according to the specific transformation between two images – Euclidean, Affine, or the most general Projective case. The main feature of our technique is the parameters chosen to represent the images in the hash table, so that number of computations required in the matching part is very small.

A 2-D transformation requires K basis points (K = 3 for Euclidean and Affine, 4 for Projective). We can select ordered pairs of K basis points from the first image in $\binom{M}{K} \times K!$ ways (this is $\mathcal{O}(M^K)$). For each such basis, we compute the coordinates of the remaining $M - K$ ($\mathcal{O}(M)$) points. A *hash table* stores these coordinates, indexed by the basis points. We repeat the process for the second image. Matching rows of coordinates between hash tables of the two images has *quadratic* time complexity. We can reduce this to *linear* is we sort each row in the hash tables. Hence, the problem of matching image features reduces to $\mathcal{O}(M^{K+1}N^{K+1}) \times$ the row matching time. This is has polynomial time complexity, an improvement over the exponential time complexity required for a naive feature match. We show the application of Geometric Hashing to two important cases of mosaicing. In each case, we use the above idea to further reduce the time complexity of image alignment.

3 Mosaics for Planar Rigid Camera Motion

Two camera positions are related by a 3-D Euclidean (rigid-body) transformation:(see the Appendix for a brief description of imaging geometry)

$$\mathbf{P}' = \mathcal{R}\mathbf{P} + \mathcal{T} \tag{1}$$

Here, $\mathbf{P} = [X\ Y\ Z]^T$ and $\mathbf{P}' = [X'\ Y'\ Z']^T$ represent the (non-homogeneous) 3-D coordinates of a point viewed by the two camera stations, and let $\mathbf{p} = [x\ y\ 1]^T$ and $\mathbf{p}' = [x'\ y'\ 1]^T$ be the corresponding image points. $\mathcal{R}$

and $\mathcal{T}$ represent the 3-D rotation and translation between the world coordinate system and camera coordinate system. For a planar rigid transformation (say in the XY-plane),

$\mathcal{T} = [T_x\ T_y\ 0]^T$ *i.e.* no translation along Z-axis. and

$$\mathcal{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The 2-D image points and 3-D points in the camera coordinate system are related by

$$\lambda \mathbf{p} = \mathbf{A}\mathbf{P} \tag{2}$$

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3}$$

where $\mathbf{A}$ represent the matrix of internal camera parameters. Here λ represents a projective constant, f_x and f_y represent the focal lengths in the x- and y- directions, s a skew factor and (u_0, v_0) represent the position of the principal point [5].

This gives two independent equations

$$\frac{y}{z} = \frac{y - v_0}{f_y} \tag{4}$$

$$\frac{x}{z} = \frac{x - u_0}{f_x} - s\frac{(y - v_0)}{f_x f_y} \tag{5}$$

Putting these equation in the equation (1) , we have

$$\frac{x' - u'_0}{f'_x} - \frac{s'(y' - v'_0)}{f'_x f'_y} = \frac{(x - u_0)cos\theta}{f_x} - \frac{scos\theta(y - v_0)}{f_x f_y} - \frac{sin\theta(y - v_0)}{f_y} + \frac{T_x}{z} \tag{6}$$

$$\frac{y' - v'_0}{f'_y} = \frac{(x - u_0)sin\theta}{f_x} - \frac{ssin\theta(y - v_0)}{f_x f_y} + \frac{cos\theta(y - v_0)}{f_y} + \frac{T_y}{z} \tag{7}$$

where the primed quantities are the internal camera parameters in the corresponding second image. This can be written as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix} \tag{8}$$

where

$$r_{11} = \frac{(s'sin\theta + f'_x cos\theta)}{f_x};$$

$$r_{12} = \frac{f_x(s'cos\theta - f'_x sin\theta) - s'(f'_x cos\theta + s'\sin\theta)}{f_x f_y};$$

$$T_x = \left(u'_0 - \frac{u_0(f'_x cos\theta + s'sin\theta)}{f_x}\right) + \left(\frac{s'v'_0 + v_0(f'_x sin\theta - s'sin\theta - s'cos\theta)}{f_y} + \frac{s'v'_0}{f'_y}\right) + \left(\frac{v_0 s(f'_x cos\theta + s'sin\theta)}{f_x f_y} + \frac{T_x f'_x + T_y s'}{Z}\right);$$

$$r_{21} = \frac{f'_y sin\theta}{f_x}; r_{22} = \frac{f'_y(f_x cos\theta - ssin\theta)}{f_x f_y};$$

$$T_y = v'_0 - \frac{f'_y u_0 sin\theta}{f_x} + \frac{v_0 s f'_y sin\theta}{f_x f_y} - \frac{v_0 f'_y cos\theta}{f_y} + \frac{f'_y T_y}{z} \tag{9}$$

This is a 2-D affine transformation with 6 parameters. The transformation can be computed from three point correspondences. *It is important to note that the relative change of successive camera positions is often kept small to maximize the numbers of corresponding points between images*. We use this observation to make a simplifying assumption.

Hence, instead of storing $(m-3)$ coordinate for each basis triplet, we have taken the angle θ formed by two linearly independent vectors based on these basis triplet and length l between the two end points as a parameters in the hash table. So, there will be $2m$ number of values in the hash table for comparison with $2n$ values derived from second image. So the order of computations is thus lower. Actually, only those triplet pairs from reference image and second image with minimum difference in angle will be considered for comparison with respect to length.

Algorithm 1:

1. Represent the reference frame by the sets of corner points.

2. For every non-collinear triplet of points, form two vectors and find the angle (θ) formed by two linearly independent vectors and length l between two end points. We use these as parameters in the hash table. In this way, we have $\binom{M}{3}$ values of θ and l.

3. For the second frame of the scene, find the angle θ and l for every triplet as shown in the Figure 1. So we have $\binom{N}{3}$ values of θ and l for hash table comparison.

4. For every basis triplet in the second image, find the difference in angles θ_{s_j} and angle θ_{r_i} of all basis triplet in the reference image.

$$\delta_{\theta_{(i,j)}} = \mid \theta_{s_j} - \theta_{r_i} \mid$$

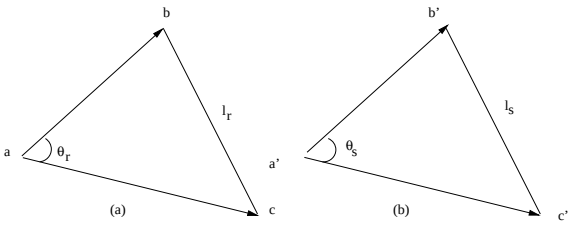


Figure 1: (a, b, c) triplet in the ref.image (left) and (a', b', c') triplet in second image(right)

where $i = 1, 2, 3 \ldots \binom{M}{3}$; $j = 1, 2, 3 \ldots \binom{N}{3}$. Similarly, we calculate the difference in length as

$$\delta_{l_{(i,j)}} = \mid l_{s_j} - l_{r_i} \mid$$

Out of $\binom{M}{3} \times \binom{N}{3}$ combinations, we discard the basis triplets which give an angle difference more than a threshold. In this pass many pairs are expected to be disqualified. The pair of triplets with minimum difference in θ is considered for comparison based on length. In one of set experiment, number of feature points in both reference image and second images are 90. For these feature, we have 117480 values of θ and l in the hash table. First these pair of triplets are compared based on angle θ. By fixing some threshold, we can discard triplet pair, which gives angle difference more then threshold. It is found the at 105700 triplet (*i.e.,89.97 present*) get disqualified. The remaining 11794 (*i.e., 10.13 Present*) triplet are compared based on length. By sorting based on δ_l, choose the triplet pair with minimum value of δ_l. Thus the triplet pair with least values of δ_θ and δ_l can be considered as the right candidate for matching.

The idea of doing this is to reduce the length of the hash table, so that one has to compute only a few candidate matching triplets between the two image pairs. Since we are looking for correspondences between interest points detected in for separate images, only those triplets which preserve the shape and size in the two images are considered for possible matching. It should be noted that θ and l are *not affine invariants* [13, 9]. However, we may often make this assumption as motion of the camera is often kept very small to generate good quality mosaics.

The required transformation can be obtained from a pair of matched triplets or estimated from more matched vertices by using least square error(LSE) estimation method. If there are more than three correspondent vertex pair, say $((x_1, y_1), (x_2, y_2) \ldots (x_h, y_h))$ and $((x'_1, y'_1), (x'_2, y'_2) \ldots (x'_h, y'_h))$ where $h \geq 3$, the required transformation can be obtained as the solution of LSE estimation which minimizes the LSE measure

$$\sum_{i=1}^{h} \parallel p'_i - (\mathcal{R}p_i + \mathcal{T}) \parallel^2 \qquad (10)$$

with respect to the motion parameters. By estimating the transformation, then second frame is transformed with respect to reference image and both are combined to form mosaic. We take the region of overlap from one constituent image. The Figure 2(a) shows the experimental set up for planar rigid camera mosaic. The first row of Figure 3 shows two images taken by such an imaging setup. The image at

Figure 3: Two sample images and their resultant mosaic (bottom row)

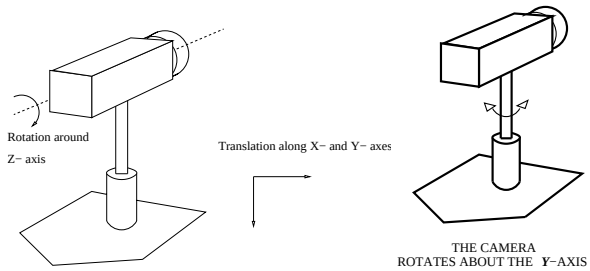
the bottom shows the resultant mosaic. We show another example of this case in Figure 4. There is very little registration error in this example, too.

4 Panoramic Image Mosaicing

In the case of a collection of images of a planar scene taken from different points of view or a collection of images of 3-D scene taken from the same point of view (*i.e.* the only difference between the images is a rotation around the optical center of the camera, as in Figure 2(b)), the transformation between the images is a linear transformation of 2-D projective space $\mathcal{P}^2$, called a collineation or a homography [15]. A commonly used camera model is [5]: (See Appendix for a short summary)

$$\lambda \mathbf{p} = \mathbf{A} \left[\begin{array}{c|c} \mathcal{R} & \mathcal{T} \end{array} \right] \mathbf{P} \qquad (11)$$

relating the coordinates of a 3-D point in the world coordinate system $\mathbf{P} = [X\ Y\ Z\ 1]^T$ to its image point $[x\ y\ 1]^T$. λ is a projective constant. Here $\mathbf{A}$ denote matrix of internal camera parameters, $\mathcal{R}$ denote a rotation matrix and $\mathcal{T}$,



(a) (b)

Figure 2: Experimental set up for (a) planar rigid camera motion, and (b) panoramic imaging

Figure 4: Two sample images and their resultant mosaic(bottom row): Details in text

a translation vector. We can relate the image coordinates to the (non-homogeneous) coordinates of the 3-D points in the camera coordinate systems using $\lambda \mathbf{p} = \mathbf{AP}$ and $\lambda' \mathbf{p}' = \mathbf{A}'\mathbf{P}'$. For two cameras looking at the same point 3-D point $\mathbf{P}$

$$\mathbf{P}' = \mathcal{R}\mathbf{P} + \mathcal{T} \tag{12}$$

For panoramic image mosaicing, $\mathcal{T} = 0$. So $\lambda' \mathbf{A}'^{-1}\mathbf{p}' = \lambda \mathcal{R} \mathbf{A}^{-1}\mathbf{p}$. Hence, we have

$$\mu \mathbf{p}' = \mathbf{H}\mathbf{p} \tag{13}$$

$$\mu \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{14}$$

H is a 3×3 invertible, non-singular homography matrix. The above homography matrix represents a 2-D to 2-D projective transformation. Homographies and points are defined up to a nonzero scalar. For the principle point of image 1 we have $(x_1, y_1) = (0, 0)$. Its corresponding location in the coordinates of image 2 is $(\frac{h_3}{h_9}, \frac{h_6}{h_9})$. As long as camera is well above the ground, the principal point of image 1 must be a well defined point(finite) in the coordinates of image 2. Hence $h_9 \neq 0$. so we take $h_9 = 1$. So eight parameters are to be found out [2]. The above equation can be written as

$$x' = \frac{h_1 x + h_2 y + h_3}{h_7 x + h_8 y + 1} \quad y' = \frac{h_4 x + h_5 y + h_6}{h_7 x + h_8 y + 1} \tag{15}$$

Every point correspondence gives two equations, thus to compute H, we need four point correspondence. For a pair of corresponding points, it can be written as

$$h_1 x + h_2 y + h_3 - h_7 x x' - h_8 y y' = x'$$
$$h_4 x + h_5 y + h_6 - h_7 x y' - h_8 y y' = y'$$

Therefore, we use a projective basis for our geometric hashing-based scheme. We consider projective bases defined by pairs of four non-collinear projective points, using the canonical frame construction of [11]. This method considers mappings from the four non-collinear points to the corners of a unit square. Thus, we have $\binom{m}{4} \times m!$ possible choices for the basis vectors. We repeat the procedure of Section 2 for $k = 4$ here. However, as in Section 3, we can make a similar assumption here, to simplify the image alignment computation.

Algorithm 2:

1. Represent the reference image by the sets of corners.

2. For every quadruplet (of which three must be non-collinear), find the angles (θ_1, θ_2) formed by two linearly independent vectors and lengths(l_1, l_2) between two end points as shown in Figure 5.

3. For the second frame of the scene, for every quadruplet find the corresponding (θ, l) values.

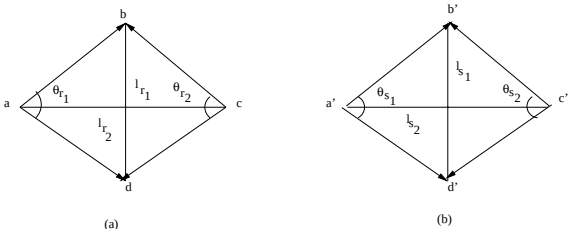


Figure 5: (a, b, c, d) basis quadruplet in reference image (left) and (a', b', c', d') basis quadruplet in second image(right).

4. for every quadruplet in the second image, find the difference between angle $\theta_{s_{1_j}}$ and angle $\theta_{r_{1_i}}$ and difference between $\theta_{s_{2_j}}$ and angle $\theta_{r_{2_i}}$ of all quadruplet in the reference image:

$$\delta_{\theta_{1_{(i,j)}}} = \mid \theta_{s_{1_j}} - \theta_{r_{1_i}} \mid,\ \delta_{\theta_{2_{(i,j)}}} = \mid \theta_{s_{2_j}} - \theta_{r_{2_i}} \mid$$

Similarly, Calculate the difference in lengths

$$\delta_{l_{1_{(i,j)}}} = \mid l_{s_{1_j}} - l_{r_{1_i}} \mid,\ \delta_{l_{2_{(i,j)}}} = \mid l_{s_{2_j}} - l_{r_{2_i}} \mid$$

where $i = 1, 2, 3 \ldots \binom{M}{4}$; $j = 1, 2, 3 \ldots \binom{N}{4}$. out of $\binom{M}{4} \times \binom{N}{4}$ combinations, few most likely correct pairs can be identified through two passes. We can discard the quadruplets which give angle difference more than threshold. The pairs of quadruplets with small difference in θ_1 and θ_2 will be considered for comparison based on lengths. By sorting based on δ_{l_1} and δ_{l_2}, choose pairs with minimum value of δ_{l_1} and δ_{l_2}. So, the pair with least values of $\delta_{\theta_1}, \delta_{\theta_2}, \delta_{l_1}, \delta_{1_2}$, considered as right candidate. So a quadruplet in the reference image matches with quadruplet in the second image. Even in the absence of any invariance in parameters θ and l, the above constraints can be safely used as the relative change in these parameters is very small due to dense time sampling of images. So, the pair with least values of $\delta_{\theta_1}, \delta_{\theta_2}, \delta_{l_1}, \delta_{1_2}$ is considered as a right candidate. So this means this four points in the first image have correspondence with the four points in the second image. In the above algorithm for reference image $\theta_1, \theta_2, l_1, 1_2$ values are stored. In the matching part, while choosing any four points from the second image, depending on the $\theta_1, \theta_2, l_1, l_2$, their will be no match. So in order to have robustness in matching , every non collinear four points are stored in the hash table. Also in the second image we are looking for only one quadruplet arbitrarily, and based on $\theta_1, \theta_2, l_1, l_2$ we are looking for match in the table. Though it match, it might be be the wrong candidate and there may be some other quadruplet in the second image which can match with the the first image. So, in order to avoid this ambiguity, comparison is done for all the possible non collinear quadruplet in the second image. By knowing these correspondence, we can find the $T_{initial}$ between the images. Then T_{final}, is obtained by using least square estimation. By estimating the transformation, the second image is transformed, then these images are combined to form mosaic. Here the reference image is selected and all other image are registered with respect to the reference image, and they are combined and complete mosaic is constructed. In this case, the region in the overlapping region is taken form one one image, so there is no effect of blurring in the mosaic image. Figure 2 shows the experimental setup for capturing the images to generate panoramic mosaic. Figure 6 shows the result obtained with our approach. We took an arbitrary set of images of the Hiranandani Complex, Powai, Mumbai using a panoramic imaging setup (as in Figure 2(b)). To capture the images, the camera was mounted on a level tripod and thirty two images were taken over an angle of approximately 270^0. Since the images were taken in a single planar rotation, the topology of the mosaic is known(*i.e.*, temporial neighbour are spatial neighbours).The rotation between the images is unknown and is not assumed to be constant. The registration error is very small up to one pixel in the initial part and it goes up to maximum two pixels in the latter part. Here alignment along the seam is accurate. The presence of seams in the resultant mosaic is due to the automatic gain adjustment of the camera.

5 Conclusion

This paper presents a new method for automatic generation of mosaics. Our method is based on geometric hashing. Matching features across images has exponential time complexity, we reduce it to polynomial time complexity. Additionally, the entire process does not require human intervention. Thus, entire process is automatic and fast. We show results in support of the proposed strategies.

Appendix:Basic Imaging Geometry and Geometric Transformation

Basic Imaging Geometry

A commonly used camera model is [5]:

$$\lambda \mathbf{p} = \mathbf{A} \left[\ \mathcal{R} \mid \mathcal{T} \ \right] \mathbf{P_w} \tag{16}$$

This relates the coordinates of a 3-D point in the world coordinate system $\mathbf{P_w} = [X\ Y\ Z\ 1]^T$ to its corresponding image point $[x\ y\ 1]^T$. λ is a projective constant. Here $\mathcal{R}$ denotes a rotation matrix and $\mathcal{T}$, a translation vector. $\mathbf{A}$

Figure 6: A panoramic mosaic created from a set of 32 frames of the Hiranandani complex, Powai

represents the matrix of internal camera parameters. The internal parameter matrix is of the form [5]:

$$\begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here f_x and f_y are the camera focal lengths in the $x-$ and $y-$ directions respectively; u_0 and v_0 represent the position of the principal point, and s is the skew factor between the two image coordinate system axes.

Geometric Transformations

We classify geometirc transformations, in increasing order of generality, as follows: (for simplicity, we consider 2-D to 2-D transformations alone)

1. Euclidean or Rigid Body transformation:

$$\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{t} \tag{17}$$

where

$$\mathbf{R} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

is a rotation matrix and $\mathbf{t} = [t_x \ t_y]^T$ translation vector. $\mathbf{p}'$ and $\mathbf{p}$ are the transofrmed and original 2-D points, respectively, represented in non-homogeneous coordinates $[x'y']^T$ and $[xy]^T$, respectively. Euclidean invariants are lengths (distances between two points), and angles.

2. Similarity transformation:

$$\mathbf{p}' = c\mathbf{R}\mathbf{p} + \mathbf{t} \tag{18}$$

where c is scaling factor. Similarity invariants are angles, ratios of lengths, and ratios of areas.

3. Affine transformation:

$$\mathbf{p}' = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \mathbf{p} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{19}$$

Affine invariants are parallel lines, ratio of lengths of parallel lines, ratio of areas.

4. Projective transformation: This is the most general gemoetric transformation. Here, the two 2-D points $\mathbf{p}'$ and $\mathbf{p}$ (represented in homogenous coordinates), are related by a 3×3 non-singular transformation matrix (a homography)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{31} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{20}$$

Projective invariants include the cross ratio of four collinear points, or four concurrent lines.

References

[1] L. G. Brown. A Survey of Image Registration Techniques. *ACM Computing Surveys*, 4:325 – 376, March 1992.

[2] R. Chellappa and Q. Zheng. Automatic Registration of Oblique Arieal Images. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 218 – 222, 1994.

[3] S. Chen. Quicktime VR, an Image Based Approch to Virtual E nvironment Navigation. In *SIGGRAPH*, pages 29 – 38, August 1995.

[4] P. Dani and S. Chaudhuri. Automated Assembly of Images:Image Montage Preparation. *Pattern Recognition*, 28(3):431 – 445, March 1995.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[6] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Object Recogination by Affine Invariant Matching. *Pattern Recognition*, pages 335 – 344, June 1998.

[7] S. Peleg and J. Herman. Panormaic Mosaic by Manifold Projection. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 338 – 343, April 1997.

[8] S. Peleg and B. Rousso. Universal Mosaicing Using Pipe Projection. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 123 – 132, March 1998.

[9] O. P.Gros and E.Boyer. Using Local Planar Geometric Invariant to Match Images of Line Segments. *Computer Vision and Image Understanding*, pages 135 – 155, Feburary 1998.

[10] B. S. Reddy and B. N. Chatterji. An FFT Based Technique for Translation, Rotation and Scale Invariant Image Registration. *IEEE Trans. on Image Processing*, (8):1266 – 1271, August 1996.

[11] C. A. Rothwell. *Recognition using Projective Invariance*. PhD thesis, University of Oxford, 1993.

[12] R.Szeliski and H.Yeung. Creating Full View Panoramic Image Mosaic and Environment Maps. In *Computer Graphic Procedding, Annual Conference Series*, pages 251 – 258, 1991.

[13] H. Sawhney and R. Kumar. True Multi Image Alignment and its Application to Mosaicing and Lens Distortion Correction. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 21(3):235 – 243, March 1999.

[14] A. Zisserman and D. Capel. Automated Mosaicing with Super-resolution Zoom. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 120 – 128, 1998.

[15] I. Zoghlami and R. Deriche. Using Geometric Corners to Build a 2D Mosaic from a Set of Images. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 420 – 425, March 1997.