# Image Mosaics Base on Homogeneous Coordinates

**Xi Shao, Changsheng Xu, Joo Hwee Lim**

Institute for Infocomm Research

21 Heng Mui Keng Terrace, Singapore 119613

{shaoxi, xucs, joohwee}@i2r.a-star.edu.sg

## Abstract

The need to combine pictures into panoramic mosaics has existed since the beginning of photography, as the camera's field of view is always smaller than the human field of view. Photo mosaicing, a technique to paste together several pictures to create a panoramic mosaic, gives us a more complete view of the scene. In this paper, an image mosaic approach is proposed. *Homogeneous coordinates* are used to represent points. The overlapped points for each RGB channel are interpolated to generate mosaics after projecting the points from different images to the reference image. Experiments illustrate that the proposed approach can obtain an ideal mosaic result.

*Keywords*: Image mosaics, homogeneous coordinates, panoramic, projection.

## 1 Introduction

Computer graphics is an emerging technique and can be applied to various fields. Among these fields, the special effects industry is one of the important applications. A technique like image mosaics would evoke uses such as better texture maps (i.e. higher resolution texture maps by stitching together several images) and image backgrounds. The image backgrounds can be used for environments maps or some blue/green screen techniques.

A number of methods have been proposed to build the image mosaicing system. Szeliski [1] proposed the Levenburg-Marquadt nonlinear minimization algorithm to refine the estimate to achieve the best transformation. Peleg [2] used *manifold projection* method to create the panoramic mosaics under very general conditions. Both of the methods achieve good mosaic results, but the computation is complex.

In this paper, the image mosaicing system utilizes a combination of manual user input for registration between two images and a homography method to attempt to get the relationship among several images. In this particular implementation, there is an underlying assumption that the pair images are related through some sort of planar transformation (i.e. one that is either projective or at least affine). The relationship between two images can be obtained through a homography, *H*. More specifically, if *x* is the homogeneous coordinates of a point in the source image, and *u* is the homogeneous coordinates of the corresponding point in the destination image, then $u = Hx$. The steps to build an image mosaicing system can be described as follows:

(1). Establish point correspondences.

(2). Estimate the forward homographies between pairs of images.

(3). Compute the bounding box of the output image.

(4). Compute the backward homographies.

(5). Produce the output image.

The rest of this paper is organized as follows. Section 2 introduces the mosaic algorithm. The steps to implement an image mosaicing system are described in section 3. Experimental results are illustrated in section 4. Finally, conclusions and future work are given in section 5.

## 2 Algorithm

### 2.1 Homogeneous Coordinates Represent Points

Before proceeding, we need to consider the geometric transformations that relate the images to the mosaic. To do this, *homogeneous coordinates* is used to represent points, that is, 2D points in the image plane can be represented as $(x, y, w)$. The corresponding Cartesian coordinates are $(x/w, y/w)$.

In *homogeneous coordinates*, all geometric transformations can be written as *Matrix Multiplication*:

$$\boldsymbol{u} = \boldsymbol{H} \bullet \boldsymbol{x} \tag{1}$$

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

where

$$u = u'/w', v = v'/w', w' \neq 0 \tag{3}$$

Composition can be done by applying transformation (1) and transformation (4) to *x*.

$$\boldsymbol{u} = \boldsymbol{H_2 H_1} \bullet \boldsymbol{x} \tag{4}$$

The simplest transformations in this general class are pure translations, followed by translations and rotations (rigid transformations), plus scaling (similarity transformations), affine transformations, and full projective transformations. Figure 1 shows a square and possible rigid, affine, and projective deformations. Forms for the rigid and affine transformation matrix ***H*** are with three and six degrees of freedom, respectively, while projective transformations have a general ***H*** matrix with eight degrees of freedom.

$$H_{\text{rigid-2D}}=\begin{bmatrix}\cos\theta & -\sin\theta & t_x\\ \sin\theta & \cos\theta & t_y\\ 0 & 0 & 1\end{bmatrix} \quad (5)$$

$$H_{\text{affine-2D}}=\begin{bmatrix}a & b & c\\ d & e & f\\ 0 & 0 & 1\end{bmatrix} \quad (6)$$

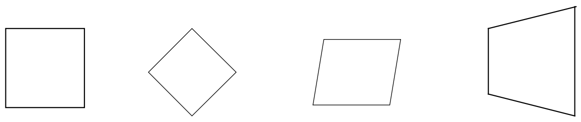


Figure 1. Square and rigid, affine, and projective transformations.

## 2.2 Image Blending Calculations:

Since the two images that are used will probably not have perfectly matching pixels at all regions where they overlap, the image blending calculations are designed to average and more properly meld the two images together. In addition, this calculation is aimed at eliminating the boundary line from one image to another. Without loss of generality, let the four pixels immediately surrounding ($u$,$v$) have coordinates (0,0); (0,1); (1,0); (1,1). And let ($u$,$v$) have values between 0 and 1. Thus ($u$,$v$) falls within the square whose corners are the coordinates given above. Let the colors at these corners be $c_{00}$, $c_{01}$, $c_{10}$, and $c_{11}$ respectively. We have

$$c_{uv} = (1-u)(1-v)\, c_{00} + u(1-v)\, c_{01} + (1-u)v\, c_{01} + uv\, c_{11} \quad (7)$$

When working with RGB colours, we need to simply interpolate for each channel (i.e. do it for R, G, B, separately). The above formula can be optimized in terms of following calculations, which uses three multiplies instead of eight:

$$c_{u0} = c_{00} + u(c_{10} - c_{00}) \quad (8)$$

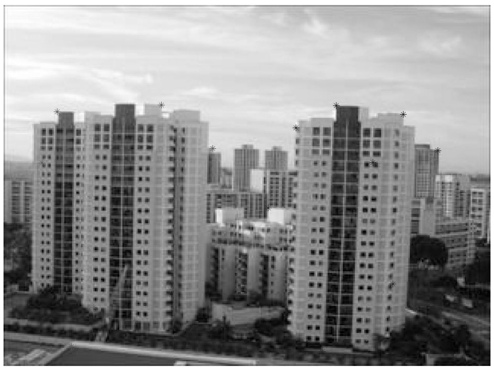
$$c_{u1} = c_{01} + u(c_{11} - c_{01}) \quad (9)$$

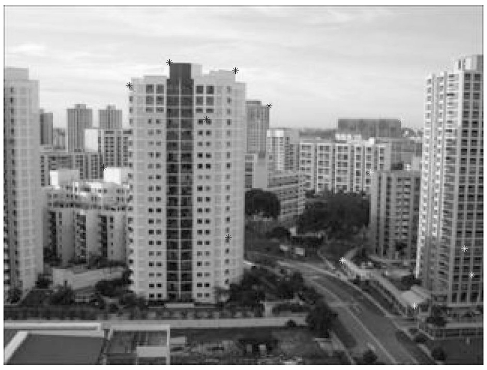
$$c_{uv} = c_{u0} + v(c_{u1} - c_{u0}) \quad (10)$$

## 3 Mosaic System Implementation

### 3.1 Establish Point Correspondences

mosaic1.jpg


mosaic2.jpg


mosaic3.jpg

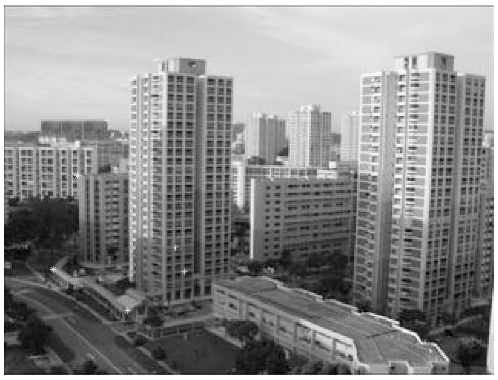

mosaic4.jpg

Figure 2. Four input images

There are four input images: mosaic1.jpg, mosaic2.jpg, mosaic3.jpg, and mosaic4.jpg, shown in Figure 2.

The first step is to establish point correspondences between the images. Because the input images only partially overlap, the correspondences are obtained only between the first and second, between the second and third, and between the third and fourth images. Using Matlab, manually mark and record at least four corresponding points between each pair of images. For the best results, the points should not all lie on a straight line, but should be spread out in the image. Use the zoom feature in Matlab to mark the points more accurately. It is usually easier to mark prominent features in the image, e.g. corners of buildings.

## 3.2 Estimate Forward Homographies

For each pair of images, the forward homographies are estimated. For instance, for the first pair of images, estimate $H_{12}$, the homography that will map the points in mosaic1.jpg to their corresponding points in mosaic2.jpg.

Do this as follows: let the points in the first image be $(x_1, y_1), \dots, (x_4, y_4)$, and their corresponding ones in the second image be $(u_1, v_1), \dots, (u_4, v_4)$. Then for each corresponding pair of points, we can obtain:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{11}$$

Although there are nine unknowns $a,\dots,I$ in the homography matrix, only eight of them need to be calculated because we are working in homogeneous coordinates. It is customary to let $i = 1$,and then seek to determine the other unknowns. Rewriting all the equations in terms of the unknowns $a,\dots,h$, we get an 8x8 system:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -u_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -u_4x_4 & -u_4y_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1x_1 & -v_1y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -v_2x_2 & -v_2y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -v_3x_3 & -v_3y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -v_4x_4 & -v_4y_4 \end{bmatrix} \bullet \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{12}$$

Note the structure of matrix $A$, we can extend this matrix to handle $n > 4$ corresponding points. The vector $p$, and hence the homography $H12$, are solved using the pseudo inverse:

$$p = A^{+} \bullet b \tag{13}$$

where

$$A^{+} = (A^{T} \bullet A)^{-1} \bullet A^{T} \tag{14}$$

## 3.3 Compute Bounding Box

To figure out the size of the output image we need to compute the maximum extent of each image after it is warped. But we need to specify a reference image at first. This is the image to whose viewpoint all other images will be warped. Here, we will designate the third image (mosaic3.jpg) as the reference image. The other three images will be warped and blended with the reference image to produce the final mosaic. Because homographies transform rectangles into quadrilaterals, all we need to do is to keep track of the four corners of each image to be warped. After warping, we find the minimum and maximum corner coordinates from all the 16 corners, and these will determine the bounding box (the smallest rectangle that contains the mosaic).

Since our reference image is the third one, we need the homographies: $H13$, $H23$, $H33$ and $H43$. We already have $H23$ from the previous step. Compute $H13$ as the product $H13 = H23H12$. This product warps a point from image 1 to image 2, and then from 2 to 3. $H33$ is simply the 3x3 identity matrix, and $H43$ is the inverse of $H34$. With these homographies, the four corners of each input image are warped into the reference image. The corners have coordinates: (1,1), $(w,1)$, $(1,h)$, and $(w,h)$, where $w$ is the width of each image, and $h$ is its height. Then, find the minimum and maximum coordinates of the warped corners. These will become the upper left corner and lower right corner of the bounding box, respectively. Let ($x$min , $y$min); ($x$max , $y$max) be these coordinates. Then the width and height of the bounding box are:

$$bw = x\text{max} - x\text{min} \tag{15}$$

$$bh = y\text{max} - y\text{min}. \tag{16}$$

## 3.4 Compute Backward Homographies

Now we need to get the backward homographies $H'31$, $H'32$, $H'33$ and $H'34$. However, these are not just the inverse of the forward homographies, because we are shifting the origin. With respect to the reference image, the upper left corner of the bounding box is at ($x$min, $y$min). We have to make this the new origin, for convenience when producing the output image. Thus we introduce a shift (translation) in the homographies. This is achieved as follows:

$$\mathrm{H}'31 = H_{13}^{-1}\,\mathrm{T} \tag{17}$$

where the translation matrix is given by:

$$\mathrm{T} = \begin{bmatrix} 1 & 0 & x_{\min} \\ 0 & 1 & y_{\min} \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

The other backward homographies $H'32$, $H'33$, and $H'34$ are similarly computed.

### 3.5 Produce the Mosaic

We can now produce the final mosaic using the destination scan method:

```
(1). for x = 1 to bw
(2).   for y = 1 to bh
(3).     Find (u,v), the point in image 1 where (x,y) warps to.
         Use the backward homography H'31.
(4).     Compute c, the color at (u,v) in image 1. Use bilinear
         interpolation to do this. If (u,v) falls outside image 1,
         no color is assigned to c.
(5).     Repeat steps (3) and (4) for images 2, 3 and 4. Blend
         all the colors, so the blended color is computed by
         averaging them. Let c blend be the blended color.
(6).     Set the color in the output image at (x,y) to be c blend.
```

## 4 Experimental Results

The experimental result of mosaic from input images in Figure 2 is illustrated in Figure 3.

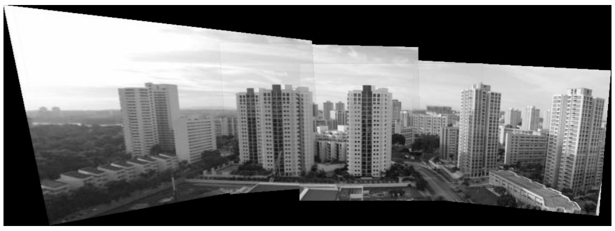


Figure 3. Mosaic image

It can be seen that the mosaic image is able to get a better panoramic view than individual images that are used to generate the mosaic image.

## 5 Conclusions and Future Work

In this paper, we use *homogeneous coordinates* to represent points. After projecting the points from different images to the reference image, we simply interpolate overlapped points for each RGB channel and get the Mosaics.

Still imagery can be used in a variety of ways, including the manipulation and compositing of photographs inside video paint systems, and the texture mapping of still photographs onto 3D graphical models to achieve photorealism. Although laborious, it is also possible to merge 3D computer graphics seamlessly with video imagery to produce dramatic special effects. As computer-based video becomes ubiquitous with the expansion of transmission, storage, and manipulation capabilities, it will offer a rich source of imagery for computer graphics applications. One novel use of image mosaics, or at least a related variant of it, would involve not only stitching images over space in a continuous fashion, but also over time.

## 6 References

[1] R. Szeliski. Video mosaics for virtual nvironments. *IEEE Computer Graphics and Applications*, pages 22-30, March 1996.

[2] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 338-343, San Juan, Puerto Rico, June 1997.