

# View Synthesis by Image Mapping and Interpolation

Farris J. Halim<sup>1</sup> Jesse S. Jin<sup>1,2</sup>

<sup>1</sup> School of Computer Science & Engineering, University of New South Wales  
Sydney, NSW 2052, Australia

<sup>2</sup> Basser Department of Computer Science, University of Sydney  
Sydney, NSW 2006, Australia

{farrisjh, jesse}@cse.unsw.edu.au

## Abstract

This paper is about implementing and analysing a strategy of generating the intermediate views of a scene from a pair of images taken from different position and orientation. In particular, the algorithm uses image mapping for pixel registration purpose and a form of interpolation to produce the in-between view. The main purpose is to note the validity and limitation of such method and to explore potential improvement and development. This work may also be a solid base for future works including the application of 3D digital video.

*Keywords:* View Synthesis, Immersive Video, 3D Video

## 1 Introduction

The concept of virtual reality is growing and people are now experiencing the flexibility of viewing control in various programs such as 3D computer games, modelling program, and many more. Program written in VRML (Web3D Consortium 2001) has an ability to navigate the objects in the scene using mouse and control buttons. This means that observers can choose where and how to see the objects anyway they like.

Having the same sort of experience with real video scene requires more approach. The basic idea of this problem is to generate a view of a scene from the angle requested by the observer. Clearly, the observer may request a viewing location where a physical camera does not exist, so it needs to have some way to generate this view using the available cameras.

Logically, only a limited number of cameras can be placed. From these cameras it needs to produce the requested view. Fortunately, if the information from the multiple cameras is combined, it is possible to predict and therefore generate new viewpoints. In particular, the problem of generating in-between views from a set of images is usually called view synthesis. View synthesis can be applied to both still images and video, where video is simply a sequence of still images or frames.

### 1.1 Strategy

According to Pollard *et al.* (Pollard, Pilu, Hayes & Lorusso 1998), there are various approaches to this problem and they are divided into three main categories

based on the underlying technique employed in the system. They are reconstruction-projection, projective transfer, and forms of image interpolation/morphing.

The first category approaches the problem by reconstructing the objects in the scene as a 3D model. The colour of the objects themselves is obtained by texture mapping. In order to perform the reconstruction itself, the program requires some knowledge about the camera calibration. This includes the coordinate position, the angle and orientation, and the optical characteristic of the camera. Such method is employed in the Immersive Video system, as described by Moezzi *et al.* (Moezzi, Katkere, Kuramura & Jain 1996).

Projective transfer, as in the first category, uses dense correspondences to predict where pixels end up in the virtual projectively distorted images. This means that all pixels in the source images are transferred to the view image using projection mechanism.

The last category uses simple image interpolation and intensity blending to generate in-between view from the original sets. This approach will be the base of this project.

### 1.2 Related Work

#### 1.2.1 View Synthesis by Edge Transfer

Pollard *et al.* (Pollard, Pilu, Hayes & Lorusso 1998), worked on a novel automatic method for view synthesis from a triplet of uncalibrated images based on trinocular edge matching followed by edge transfers using linear interpolation, occlusion detection and correction and finally rendering. This method has the advantage of performing a much simpler computation than having to projectively calculate the pixel location or reconstructing the 3D objects. According to Seitz and Dyer (Seitz & Dyer 1995), the in-between views produced by interpolation are physically valid if the images are first re-projected to conform to parallel camera geometry. However Pollard *et al.* took this approach and produced an approximate view using linear interpolation.

The method begins by performing registration. It uses three source images, arranged in a triangular form. The edges are extracted in each image and then processed to produce full edge correspondences between the image triplets. Then, the matched edges are transferred to the desired viewpoint using simple linear interpolation. After that, it finally performs the rendering, which uses intensity blending technique. The fact that it uses three images means that the observer can actually moves freely

in two dimensions, within the triangular shape defined by the camera.

### 1.2.2 Immersive Video

Another system called the Immersive Video was developed by a number of researchers from University of California (Moezzi, Katkere, Kuramura & Jain 1996). This system is based on building dynamic 3D reconstruction of the scene using information from multiple cameras to produce the Environment Model.

The Environment Model has a global view of the scene, including camera locations and parameters, and the state of all objects in the environment. The system consists of four components. The video data analyser is aimed to detect and track dynamic objects, which can be configured to observer's interest. The Environment Model Builder uses the static information of the scene (i.e. the objects that are not moving) and the information from previous component to obtain the model for the environment. Finally observer can select viewpoints using the viewer interface and see the result produced by the visualiser.

The advantage of this approach is that it can produce higher quality results because of the sophisticated environment builder. It is not only reconstructing the 3D shape but also making estimation on the world position of the objects and finally performing object identification and tracking. They use voxel representations to model the dynamic 3D objects. It is performing prediction on where each pixel on the image would lie in the 3D world of voxel or volume elements. Voxel is the equivalent of pixel in the 3D space.

## 2 Design and Implementation

The goal of this project is to build a system that can be used to generate in-between view from a pair of images taken by camera. The in-between view will be an approximation of what would be observed if there is a real camera between the existing cameras. The position of the virtual viewpoint is parameterised by the ratio of the distance between the virtual viewpoint and one of the source images with the distance between the two source images.

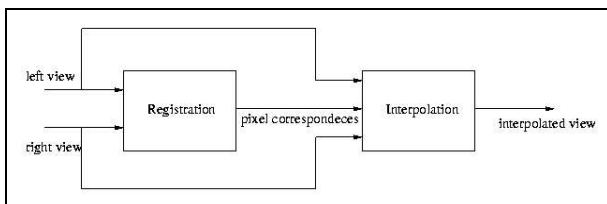


Figure 1: High level view of the system

### 2.1 Methodology

The process of view synthesis described in this thesis is broken down into two main stages (see Figure 1). The registration stage is responsible on retrieving the correspondences between the image pairs. For this project, a simple image-mapping algorithm is used. More

emphasis will be put towards the second stage. Note that the first and second stage is more or less independent. This means that no matter how the registration is performed, as long as the data passed from the first stage to the second stage is in consistent format, the algorithm will work.

The second stage is to actually produce the novel view or in-between view using linear interpolation technique. This is in conjunction to the work done by Pollard *et al.* (Pollard, Pilu, Hayes & Lorusso 1998). Once again, due to the independence nature of the two processes, a change in the second stage method should not affect the first one. Thus, both stages could be improved or reworked in the future quite independently or with little adjustment.

### 2.2 Image Registration

The aim of registration is to obtain some kind of relationship between the image pairs. This relationship could be interpreted in various ways. In all cases, it is necessary to get some kind of correspondences between pixels on the left image and pixels on the right image. In the simplest case, the correspondences would be a full pixel-to-pixel mapping.

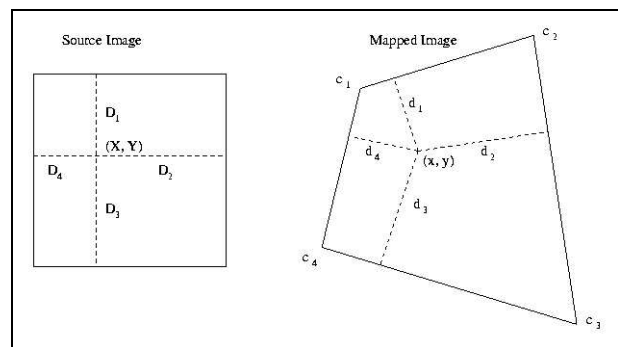


Figure 2: Image Mapping

Image mapping (Tang 2001) is used to produce the full pixel-to-pixel mapping between the first image and the second image. Image mapping is a process to map an image (in rectangle shape) into another image of arbitrary quadrilateral shape (see Figure 2). Performing image mapping is like stretching the original image into the new image defined by the quadrilateral. The quadrilateral is specified by the observer through the four corner points. With respect to Figure 2, the following equations must hold for all mapped points:

$$\frac{d_1}{d_3} = \frac{D_1}{D_3} \text{ and } \frac{d_2}{d_4} = \frac{D_2}{D_4}$$

With this property in mind, it is actually easier if the mapping is done reversely. That is, for every pixel  $p$  inside the quadrilateral in the destination image, it calculates the respective pixel in the source image that will map to  $p$ . The calculation is as follows:

$$X = \frac{d_4}{d_4 + d_2} * (D_4 + D_2)$$

$$Y = \frac{d_1}{d_1 + d_3} * (D_1 + D_3)$$

The values of  $d_1$ ,  $d_2$ ,  $d_3$ , and  $d_4$  are calculated using shortest distance (perpendicular distance) formula from the point  $(x_0, y_0)$  to a line  $A*x + B*y + C = 0$ . The formula is:

$$d = \frac{A*x_0 + B*y_0 + C}{\sqrt{A^2 + B^2}}$$

The overall process consists of the following step:

- Obtain the image to be mapped and four corner points defining the quadrilateral.
- Determine the pixels inside the quadrilateral.
- For each pixel inside the quadrilateral, calculate the corresponding pixel in the source image.
- Retrieve the colour of the pixel in the source image to be the colour of the pixel in the quadrilateral.

Determining that a pixel is inside an arbitrary quadrilateral is not straightforward if it is to be done efficiently. A technique usually used for polygon filling in computer graphics, scanline algorithm, is used for this task (Lambert 2001).

The formula to get the pixel coordinate in the source image from the pixel in the quadrilateral defined earlier produces non-integer coordinate. One way to solve it is to simply round the real coordinates into integer coordinates. This is normally called nearest neighbour sampling. While it is simple to do this, the result is not satisfactory and in order to produce smoother image, one needs to do some interpolation. This is described in the next section.

### 2.3 Image Sampling

Occasionally, program wants to access a pixel at non-integer coordinate. The image mapping procedure is one example and the image interpolation view synthesis in 2.4 is another one. The nearest neighbour sampling simply converts the non-integer coordinates into integer coordinates by rounding process. This is not satisfying in a lot of cases. The other way is to use the colour of neighbour pixels to generate the colour at the non-integer coordinate of interest. A reasonably popular method is the bilinear interpolation (Tang 2001).

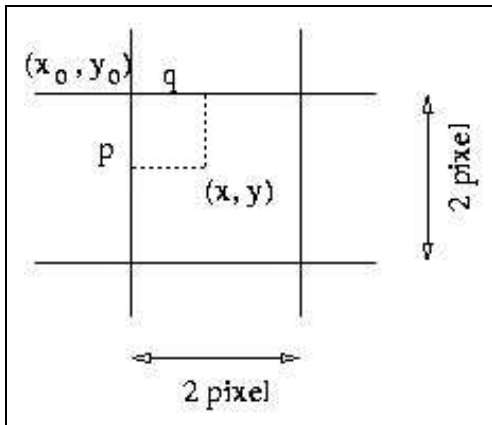


Figure 3: Bilinear Interpolation

Bilinear interpolation linearly interpolates along each row of the image and then uses the result in a linear interpolation down each column in the image. This means a linear interpolation is performed in two directions. With this method, each estimated pixel in the output image is a weighted combination of its four nearest neighbours in the input image according to the following equation (refer to Figure 3):

$$f(x, y) = (1-p)*(1-q)*f_{00} + q*(1-p)*f_{10} + p*(1-q)*f_{01} + p*q*f_{11}$$

where  $p, q \in [0,1]$

The colour of the neighbour pixels is calculated as follows:

$$f_{nm} = f(x_0 + n, y_0 + m)$$

### 2.4 Image Interpolation

The image interpolation process requires four inputs. Firstly, it needs to have the image pairs, which is needed for rendering pixel colours. Secondly, the output of the image registration process, which is the dense pixel mapping. Finally, it needs to know where observer intends to see the scene. As noted earlier, this will be defined by the ratio of the distance between the virtual viewpoint and one of the source images with the distance between the two source images. Assume this is called lambda,  $\lambda$ , which is ranged from 0 to 1 inclusive.

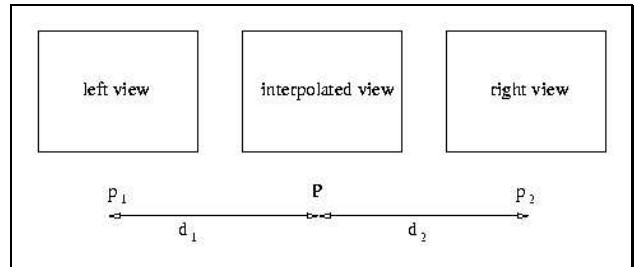


Figure 4: Image Interpolation

For each pair of points from the set given by the first stage process, a linear interpolation is performed. Let say the point  $p_1$  in the first source image maps to  $p_2$  in the second source image. Then the point where they end up in the virtual viewpoint defined by  $\lambda$  is calculated using linear interpolation as follows:

$$P = (1-\lambda)*p_1 + \lambda*p_2$$

Once the interpolated point  $P$  is known, it is a matter of determining what colour this point should be. This is done by combining or blending the colour of the pixel of the first source image (at point  $p_1$ ) and the pixel of the second source image (at point  $p_2$ ), as suggested by Pollard *et al.* (Pollard, Pilu, Hayes & Lorusso 1998). The contribution of each colour is determined by the distance of the virtual viewpoint to each of the source images. Thus, if the colour of pixel  $p_1$  is  $a$  and the colour of pixel  $p_2$  is  $b$  then the resulting colour of pixel  $P$  is:

$$C = (1-\lambda)*a + \lambda*b$$

Notice the similarity of the calculation procedure of the interpolated point and colour.

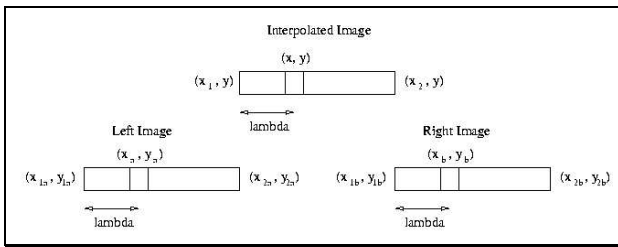
It is clear that the success of this method of view synthesis depends a lot on the accuracy of the pixel mapping produced by the registration process. A mapping of  $p_1$  and  $p_2$  is said to be accurate if pixel at  $p_1$  in one image is actually the same object (or part of object) as the pixel at  $p_2$  in the other image. If this condition is satisfied for all mapping, then the interpolation result will be correct.

Note that due to the linear interpolation routine, the distance and angular displacement between the two cameras must not be too large, otherwise it is necessary to re-project the source images to a parallel camera plane before performing the interpolation (Seitz & Dyer 1995). This is to ensure the validity of the interpolated shape of objects.

## 2.5 Interpolation Correction

Without any further work, the result of the interpolation procedure would be incomplete in the sense that not all pixels in the interpolated image are filled.

The incompleteness is due to the rounding effect of the interpolated coordinate. For example, there may be some pixel-to-pixel mappings that are interpolated to the same pixel destination. On the other hand, there are some pixels in the interpolated image that no pixel-to-pixel mapping would interpolate to.



**Figure 5: Image Interpolation correction**

Due to these missing pixels, it is necessary to perform correction routine. In this project, for each unfilled or missing pixel, it determines the corresponding pixels in both image sources. Using the colour of these pixels, it performs interpolation as usual to get the colour for the missing pixel.

Figure 5 illustrates a situation in the interpolated image, where a series of consecutive pixels (in scanline order) needs to be filled after the interpolation routine. The first points at each end that have been filled are denoted by  $(x_1, y)$  and  $(x_2, y)$  respectively.

Since the calculation of the point in the image sources are based on the relative distance from the end point of the missing region, namely  $(x_1, y)$  and  $(x_2, y)$ , then it is required to know which pair of pixels produced  $(x_1, y)$  and  $(x_2, y)$  in the first place. Since this correction procedure is done after all mapping in the dense correspondence set is interpolated completely, the intersection must be stored somewhere. For this purpose, there is a table that serves just for that. Thus, during the

first interpolation stage, this table is filled throughout the process. This table can be realised as an array of the size of the interpolated image. Each entry is a pair of the pixel origins. Un-initialised entry can be interpreted as if the pixel has not been filled. With the knowledge of the end points in all three images (the sources and the interpolated image), it is now straightforward to complete the procedure.

Suppose the left image source (see Figure 5) has end points  $(x_{1a}, y_{1a})$  and  $(x_{2a}, y_{2a})$ . Note that the segment in the interpolated image is always on the same scanline, however the segments in the source images is most likely not a horizontal line, i.e.  $y_{1a} \neq y_{2a}$  and  $y_{1b} \neq y_{2b}$ . The coordinate of pixel in the left source image that should map to  $(x, y)$  will be:

$$(x_a, y_a) = (x_{1a} + (x_{2a} - x_{1a}) * \lambda, y_{1a} + (y_{2a} - y_{1a}) * \lambda)$$

$(x_b, y_b)$  can be calculated in similar fashion. After this, the colour of pixel  $(x_a, y_a)$  in the left image is interpolated with the colour of pixel  $(x_b, y_b)$  in the right image using the interpolation formula as before.

## 2.6 User Interface

The purpose of the user interface is to guide the user to perform the step-by-step routine to do view synthesis. This begins from loading the image until viewing the interpolated novel view. The mapping procedure is quite difficult, as it requires the user to choose the best corner points by trial and error means. This user interface helps user to easily compare the result of the image-mapping routine and the source image that it should be mapped to. The graphical user interface is developed in Java using Swing library.



**Figure 6: The graphical user interface**

The main interface consists of two-canvas panel placed side by side. They are used to load the image source pairs. Both panels have the basic ability to scroll image (move image in two dimensional) and zoom the image (in and out). For the right panel, there is additional feature that allows user to choose the corners required for the image mapping. For this reason, user must load the image to be used for image mapping source on the left and the other image on the right.

The panel on the right has the ability to load two images simultaneously, although they can only be viewed one at a time. This is very helpful to compare the result of mapping to the target image. Remember that the goal is to produce a mapped image from left source image that is as close as possible to the right source image. Direct comparison can be done by toggling the image using the available button. The two images will be shown alternately on the same position and at the same scale factor.

Finally, after a reasonably good mapping has been obtained, the interpolation can be performed. User needs to specify the interpolation factor, namely  $\lambda$ . The result of the interpolation will be shown on a similar panel in a new window.

### 3 Experimental Results

#### 3.1 Test Configuration

All testing are performed under an Intel Pentium III 733 MHz machine with 256 MB PC133 SDRAM. The operating system used is Red Hat Linux 7.1. Some interpolation results can be found in Figure 7 and 8. Images are in gif format with size of 352 x 288 pixels.

#### 3.2 Result Analysis

##### 3.2.1 'painting' scene

This test case presents a painting hung on a wall (Figure 7). The objects here are relatively flat and the result of the mapping is quite good. Because of that, the interpolated views are mostly fine. Note that the right part of the image contains a non-flat region (beyond the wall) and this causes bad effects in the interpolated views. Image mapping is not aware about this and hence the mapping in those regions is simply wrong. This test shows how the system works on a real flat scene and produces reasonable in-between views out of the image pairs.

##### 3.2.2 'desk'scene

This time, the scene is quite complicated (Figure 8). It shows a lot of 3D objects taken from a close range. The mapping is chosen with the monitor shape as reference. As a result, there are a lot of artefacts in the objects surrounding the monitor. The most prominent ones are the keyboard drawer and the desk lamp. The shadows are quite obvious. It appears that there are two objects each in the interpolated view. It is simply because of the mismatches in the registration produced by image mapping. The lamp shape from the left view ended up in different position than the lamp shape from the right view.

The test shows that this system is not really suitable for complicated non-flat 3D images taken from close range. This is because at short distance, a little camera displacement causes a lot of changes in the way object is seen. By relying on direct image mapping for the registration, there will be a lot of mismatches and the resulting interpolated views can not be good.

Thus, it is really necessary to insert additional stage before the interpolation, which is responsible to re-adjust the image mapping result. Despite all that, in general the interpolated views are pretty reasonable. The effect of looking at the scene from the intermediate in-between virtual viewpoint is really shown. This means that the simple linear interpolation is actually a reasonable method to generate the novel views. Improvements need to be made on the registration part and once it is done, with the same interpolation process, the resulting interpolated views could be made better.

### 4 Conclusion

This project has successfully implemented a system to generate in-between views from a pair of images. In order to get a satisfying interpolation result, it is important to produce a good mapping of pixels between the image pairs. The current solution, which uses image-mapping algorithm, is generally acceptable for a flat scene, but not so well in real 3D scene. Even for the conventional flat scene, it is quite difficult to get the best mapping possible.

This system is an early phase of a comprehensive flexible viewing system. Obviously, there are a number of aspects that could be improved in order to get more accurate result and more efficient usage. To name a few of them:

- Automation of mapping procedure is essential to increase the system usage efficiency.
- Mapping correction procedure for non-flat scene.
- Alternative solution to obtain pixel correspondences, since image mapping has its own limitation.
- View synthesis by 3D objects reconstruction.
- Application of view synthesis to generate 3D video.

### 5 References

- Moezzi, S., Katkere, A., Kuramura, D. & Jain, R. (1996): Immersive Video. *Proc. IEEE Virtual Reality Annual International Symposium*.
- Pollard, S., Pilu, M., Hayes, S. & Lorusso, A. (1998): View Synthesis by Trinocular Edge Matching and Transfer. *Proc. The Ninth British Machine Vision Conference*.
- Seitz, S. & Dyer, C. (1995): Physically-Valid View Synthesis by Image Interpolation. *Proc. IEEE Representation of Visual Scenes*.
- Tang, T. (2001): Software Based Video Processing Using Microsoft DirectShow. Master of Information Technology thesis. University of Sydney, Australia.
- Web3D Consortium (2001): Web3D Consortium. <http://www.vrml.org/>
- Lambert, T. (2001): Polygon Filling. <http://www.cse.unsw.edu.au/~cs3421/slides/bres/ScanLine.html>



**Figure 7: The interpolation results of 'painting' scene with  $\lambda = 0.2, 0.5, 0.8$**



**Figure 8: The interpolation results of 'desk' scene with  $\lambda = 0.2, 0.5, 0.8$**