REAL-TIME VIEW MORPHING OF VIDEO STREAMS

BY

KARL WALTER TIMM
B.A., Saint Olaf College, 1974
M.S., University of Wisconsin – Milwaukee, 1987

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2003

Chicago, Illinois

This thesis is dedicated to my brother, “Otis”, who had an endless curiosity about all things technical.

ACKNOWLEDGMENTS

I would like to thank my thesis committee, Tom DeFanti, Dan Sandin, Andrew Johnson, Rashid Ansari, Martin Biernat, and Stephen Eick, for their patience, support, and confidence in me over these many years.

I would also like to express complete gratitude to my wife, Traci, whose prompting started me down this academic journey. It is by the endless support and sacrifices that she made over the years that I was able to achieve this accomplishment.

KWT

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF ABBREVIATIONS

CAVE	Cave Automatic Virtual Environment. The CAVE is a multi-person, room-sized, high-resolution, 3D video, and audio environment.
EVL	Electronic Visualization Laboratory
IBMR	Image-Based Modeling and Rendering
IBVH	Image-Based Visual Hull
MIP-map	multi-resolution data structure for storing textures
NCP	Normalized Cylindrical Projection
RIP-map	Rectangular MIP-map
SSD	Sum of Squares Difference
UIC	University of Illinois at Chicago
VR	Virtual Reality
XMIP	eXtended MIP-map (see RIP-map)

SUMMARY

This dissertation describes a real-time virtual camera application based on view morphing. This system takes video input from multiple cameras aimed at the same subject from different viewing angles. After performing real-time pattern matching, the system generates synthetic views for a virtual camera that can pan between the real views. The approach of this dissertation differs from the more common “depth from stereo” approach for generating virtual views in that it does not attempt to reconstruct the 3D structure of the original scene. Instead it takes two 2D images and directly generates the 2D output image by performing only planar operations. At the heart of the system are algorithms and data structures that support the fast inter-image correlation needed for the real-time view morphing. The contribution of this dissertation is that it demonstrates, through the use of innovative algorithms and data structures, that view morphing can be used as the basis of a real-time video avatar system running on commodity PC hardware.

1. INTRODUCTION

An important goal of computer graphics is to create realistic world scenes. One of the more challenging tasks is creating believable human faces. Because of the subtleties of expressions, synthetic faces are easy to identify even when the rest of a computer-generated scene looks believable.

Tele-immersion of an individual into a synthetic scene is the objective of many virtual reality systems. Modeling the face of an individual is difficult and time-consuming work. Having created a static model of the individual, creating believable facial expressions to convey emotion adds to the complexity of the task. When the individual is an active participant in the virtual world, video streams are used to create images that are more creative and realistic than synthetically generated images (Wang, 1998), (Yura et al., 1999), (Leung et al., 2000), (Ogi et al., 2000; 2001), and (Rajan et al., 2002).

This dissertation describes a method of creating a *video avatar* based on view morphing (Seitz and Dyer, 1996). By taking synchronous video streams of an individual from different viewpoints and manipulating them in real-time, a virtual video camera is created that can pan smoothly between the real cameras, see

Figure 1. A system such as this, when used in conjunction with a virtual reality world, creates a better sense of immersion than just video inserted into the scene from a single point of view.

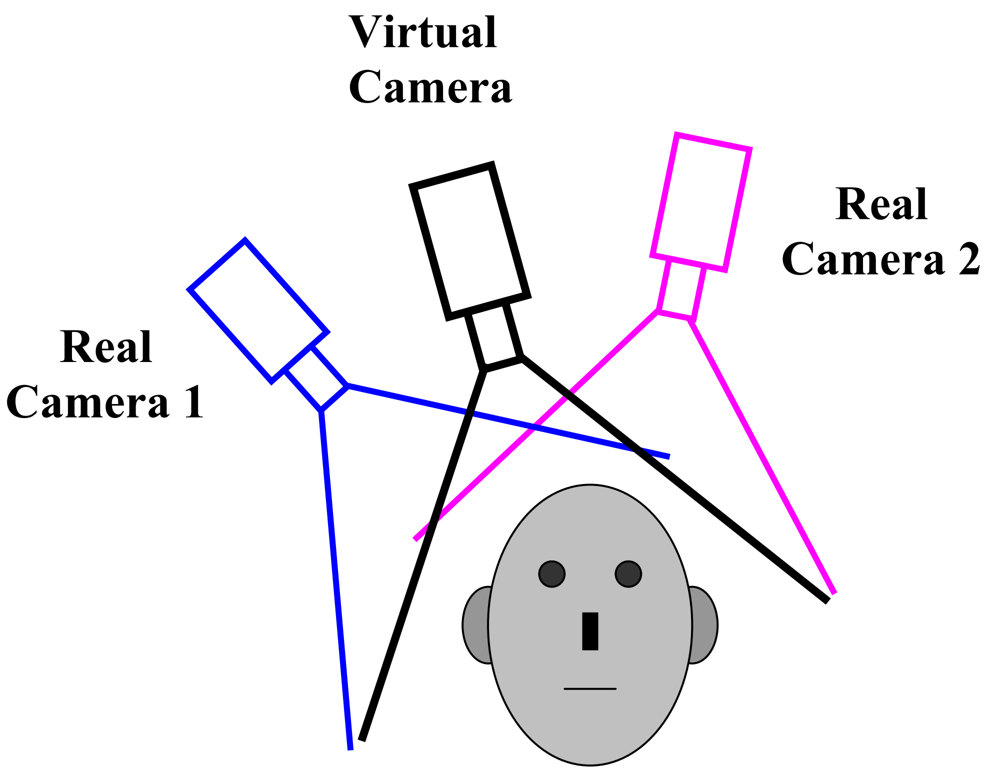


Figure 1 Diagram of the virtual camera system.

1.1 Motivation

Two examples where the view morphing system would be useful are an enhanced video teleconferencing system and a virtual reality (VR) world. Conventional video conferencing systems have the limitation of only providing a single viewpoint of the remote participant. Consider the setting shown in Figure 2 of a room with several local participants and a single remote individual. The remote person typically stares ahead, independent of who is talking. This is because a single camera is capturing the remote participant's gaze towards his/her video display of the remote group. The lack of interactivity of the remote individual's gaze breaks the illusion of proximity. This dissertation presents a system that can overcome this limitation. It takes video input from two or more cameras and combines them in real-time. A virtual camera is then manipulated to provide orientations of the individual's head from either of the extreme camera views or anything in between. Thus a video conferencing system could be created which would, theoretically, provide a stronger sense of locality of the remote participant.

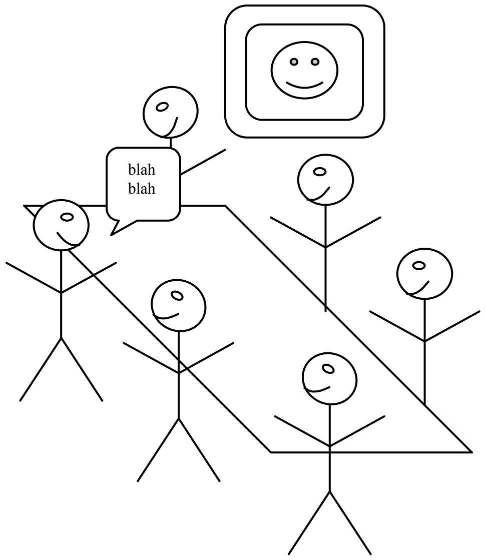


Figure 2. Traditional Video Conferencing System.

In the configuration shown Figure 3, the remote individual is presented in a more engaged manner. By augmenting the display hardware in the conference room with directional microphones, the remote participant's gaze can automatically be directed to track the current speaker in the conference room.

In the previous example, a single remote individual was placed in a conference setting. Consider the situation where all participants are remote. Such a system has been put together in the Coliseum project at Hewlett-Packard's research center (Baker et al., 2002). In their work, a completely virtual world is created where video-based 3D images of the participants are placed around a synthetic conference table. Each participant is able to view all the other participants.

Their work makes use of the Image-Based Visual Hull (IBVH) research of (Matusik et al., 2000). The IBVH approach consists of first identifying and then combining the silhouettes of various camera views of the subject to create the visual hull volume and then texture mapping view specific video images onto those hulls. The use of the view morphing approach of this dissertation in their system could produce a more realistic direction of gaze than their current IBVH approach.[1]

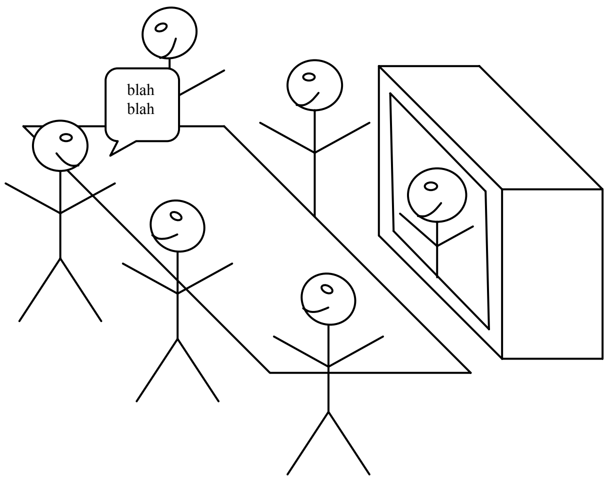


Figure 3. Virtual Camera Enhanced Video Conferencing System.

In a virtual reality system, avatars are often used to represent individuals in interactive, immersive environments. In today's systems, one typically creates a completely synthetic avatar that provides for arbitrary orientation and positioning in the 3-D scene. However it does not provide either the realism or the nuance of communication of a human face captured on video. Alternately, one can insert a video feed into the scene but realism of the video is limited because

[1] Based on discussions the author had with the researchers of the Coliseum system.

of the inability to direct the individual's gaze. The virtual camera presented in this dissertation allows video streams to be viewed from a range of different viewpoints, see Figure 4.

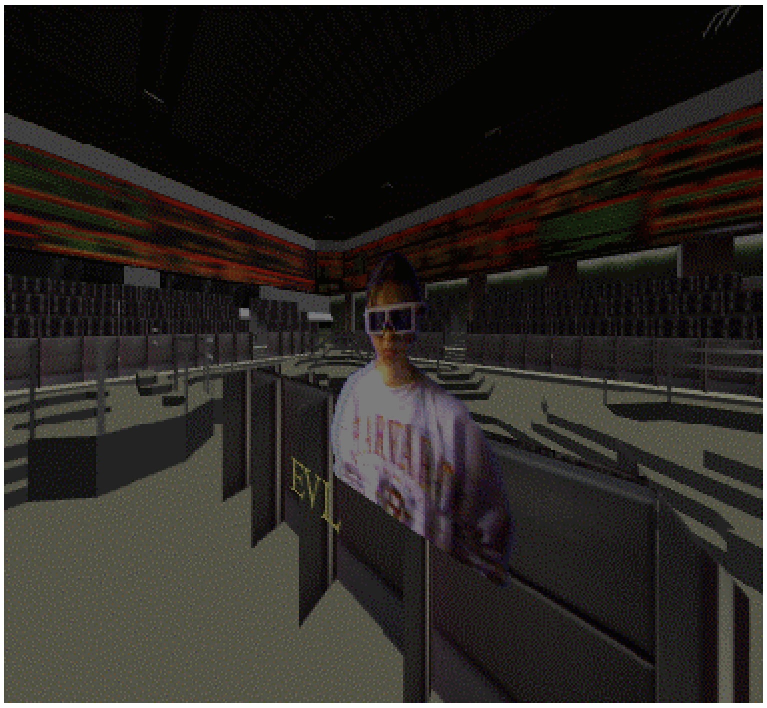


Figure 4. Virtual Camera Enhanced Video Avatar System.

(Image courtesy of Fang Wang, EVL, UIC, (Wang, 1998))

1.2 Virtual Camera

The goal of this dissertation is to describe how to intelligently blend the views of two video streams so that an image from an in between point of view can be generated. In Figure 5 below, are example images generated from the left and right video streams used as input to the system. In Figure 6 below, one can see the results of naively blending the two video streams compared to a view morph of the subject.

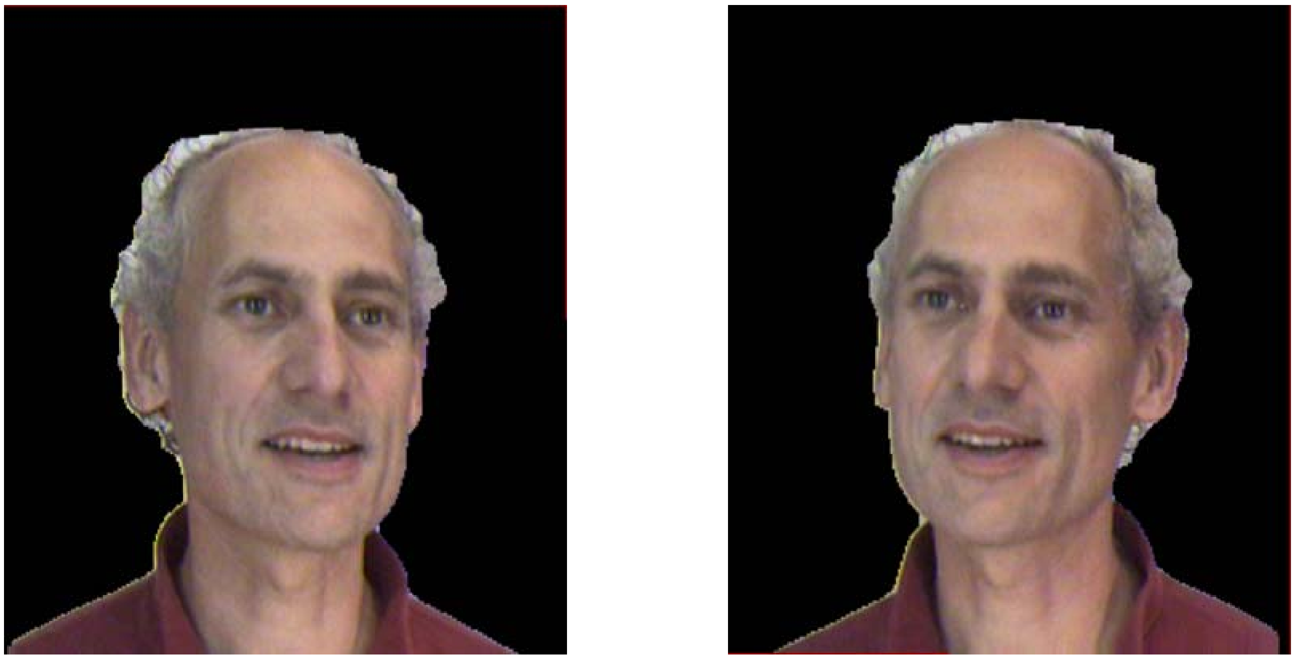


Figure 5 Left and right views from input streams.

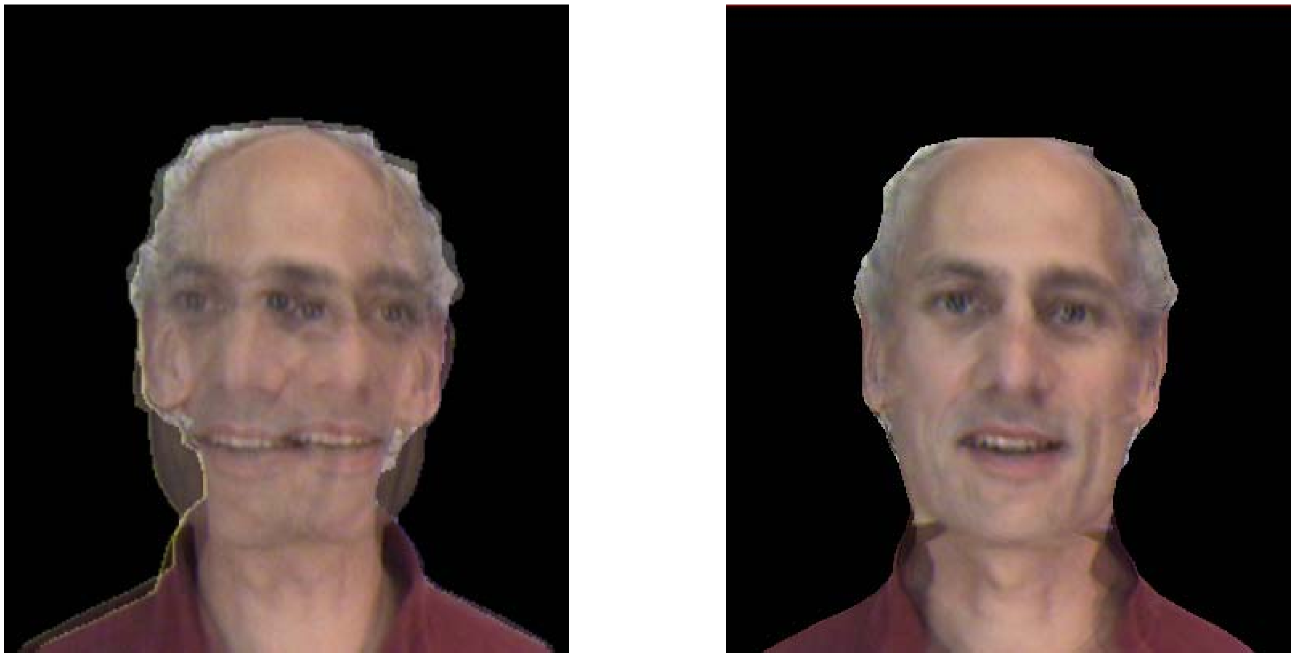


Figure 6 Naive blending of images, left, compared with a view morph, right.

1.3 Problem Statement

The problem to be tackled by this dissertation is to implement a system with the following characteristics:

- To take input from multiple video streams and output a new video stream of physically valid images of the subject that can be panned continuously between two real views.
- To run in real-time on conventional PC hardware.
- To use only the following hardware to implement the system: two (or more) video cameras, a video capture card, and a PC.

The view morph process chosen as the basis for the virtual camera depends fundamentally on establishing correspondences between images. As a result of running in real-time, an implication of the above problem statement is the following:

- The pattern matching system must be completely automated.

This dissertation describes an architecture and a set of algorithms to implement the problem statement. The application was implemented as a stand-alone system and was not integrated into a larger environment such as a virtual world.

1.4 Organization of the Dissertation

This dissertation is organized as follows. Chapter 2 gives an overview of related work. Chapter 3, *Algorithm,* presents the core ideas of the dissertation. If one were to read only one chapter of this dissertation, this would be it. Chapter 4, *Implementation*, explains various details of the systems and explains some practical considerations that had to be dealt with. Chapter 5, *Further Analysis*, investigates improvements or low level details of the system. Chapter 6, *Software Architecture*, explains the organization of the program that was implemented to demonstrate the ideas of this dissertation. Chapter 7, *Results*, discusses how well this implementation of the system performed. Chapter 8, *What Could Be Done Better*, talks about various ways the system could be improved. Chapter 9, *Future Work*, suggests various areas of future research that could be spawned from this work. Finally, Chapter 10, *Conclusion*, gives some summarizing thoughts about the dissertation.

2. RELATED WORK

The organization of this section is as follows. The first section of the related work deals with fundamental techniques, which are the building blocks for implementing the system. Those components include texture mapping, morphing, computer vision, and view interpolation. The next section surveys the field of Image Based Modeling and Rendering. The last section discusses systems that are similar in nature and include: "Morph-Based Facial Rendering", "Model-Based Facial Rendering", and "Video Avatars". The last section, Video Avatars, is a presentation of competing techniques that implement the same goal, to take input from live video and redisplay it in a 3D manner in real-time.

2.1 Fundamental Techniques

This section surveys some of the more fundamental methods used as the basis for the more advanced techniques described in the later sections of related work. The topics discussed in this sub-section include texture mapping, morphing, computer vision, and view-interpolation.

2.1.1 Texture Mapping

Ultimately, the system of this dissertation can be described as discretely sampling real world scenes, manipulating that data, resampling the resulting image, and then sending it out to a display. The use of one- and two-dimensionally sampled data, to enhance the realism of images has long been a fundamental technique of computer graphics (Catmull, 1974; Blinn and Newell, 1976). With the introduction of texture mapping came the problem of how to efficiently deal with aliasing (Oppenheim, 1989) when rendering the textured regions. Two important contributions dealing with aliasing are MIP-maps (Williams, 1983) and "Summed-Area Tables"

(Crow, 1984). Today, most computer graphics systems, even commodity PC systems, provide hardware acceleration of texture mapping and support MIP-maps to deal with aliasing.

To deal with the aliasing issues introduced with the multi-resolution image correlation described in this dissertation, a variation of traditional MIP-maps will be introduced. The trade-off, between aliasing and the loss of information from blurring, is controlled better by using more memory. And, the computational complexity is kept the same.

2.1.2 Morphing

Fundamental to the algorithm described in this dissertation is the technique of morphing (Beier and Neely, 1992). Morphing consists of two distinct steps:

1. Warping – the distorting of the geometry of the images.
2. Blending – the combining of the images by summing a percentage of their light intensities.

By varying the extent of the warp in coordination with the contribution of the light intensities of the images, an animation is produced, gradually transitioning from one image to the other. The original technique of blending two images can be extended to multiple images. The "Polymorph" (Lee et al., 1998) describes the blending of multiple faces.

Consider the morph of one view of a rigid body into another view of the same object. In general, the in-between views of the object are not physically valid. This problem has been studied by (Seitz and Dyer, 1995; 1996) and they have determined the class of "image morphs" that do produce physically correct views of objects that they refer to as "view morphs". By limiting the morphs to geometric interpolations of corresponding points of two views of an object for which the image planes of the cameras capturing the views are parallel, the in-between views produced by the morph will produce a valid view. This valid view will be the same as that taken from another camera located in-between the two cameras and whose image plane is parallel with the other cameras.

The challenge of this technique is to determine the location of corresponding points in the two images. In the case of Seitz and Dyer, this was done manually. The system in this dissertation is dealing with a steady stream of images, so manual correspondence is not possible. Fully-automated real-time vision techniques will be used instead.

2.1.3 Computer Vision

This section will not attempt to survey the field of computer vision but will rather point to a couple of tutorial references and discuss the papers with the strongest contribution to the system that was implemented. A good introduction to three-dimensional computer vision is (Trucco and Verri, 1998). A more mathematically rigorous reference that is strong on homogeneous geometry is Faugeras' "Three-Dimensional Computer Vision: A Geometric Viewpoint" (Faugeras, 1993). A very practical work describing how to efficiently manipulate images with operations such as the perspective transforms used in computer vision is Wolberg's "Digital Image Warping" (Wolberg, 1990). An example of combining vision with graphics is Szeliski's "Image Mosaicing for Tele-Reality Applications" (Szeliski, 1994). In this example Szeliski takes as input calibrated and uncalibrated images and textures to be used for computer graphical operations.

The image correlation algorithm uses a multi-resolution or coarse-to-fine approach. This approach is not new to the area of computer vision. The algorithm also has the structure of the fast wavelet transform. A survey article that discusses the multiple-resolution approach to vision along with wavelets is Mallet's "Wavelets for a Vision" (Mallat, 1996).

This dissertation developed its own matching routines. There exist many fast alignment algorithms. Recent work has been seen in the medical industry (Tokuda et al., 2002; Meijering et al., 1999). The approach taken by this dissertation, in order to optimize performance, was to focus on scan-line algorithms. An early example of inter-image scan-line approach is that of (Ohta and Kanade, 1985). After some initial investigation of these approaches, they were abandoned in favor of a simpler, lighter weight approach developed by the author.

2.1.4 View Interpolation

The system presented in this dissertation depends fundamentally on the use of *view-interpolation*. View interpolation is the process of creating novel, physically valid 2D projections of 3D scenes by warping and blending 2D images of the scene (Chen and Williams, 1993), (Laveau and Faugeras, 1994). In particular, this work was inspired by the technique of Seitz and Dyer that they termed View Morphing (Seitz and Dyer, 1995; 1996). Similar work includes (Havaldar et al., 1996) which performs view synthesis by exploiting epipolar geometry to find correspondence between views of a static scene. The work (Xiao et al., 2002) performs view interpolation on dynamic scenes.

What distinguishes this work from other systems is that it performs the image analysis and synthesis on video stream in real-time. Given its need to process many frames a second, it is, necessarily, completely automated.

2.2 Image-Based Modeling and Rendering

This relatively recent sub-field in computer graphics deals with new graphical primitives which consist of acquiring images from the natural or synthetic world, processing them in some manner, and then outputting them in a graphical display system, perhaps along with conventional graphical objects. This approach has several advantages over traditional graphics. First the generation of the 3-D models is a very time consuming process. Second there are some everyday objects that still elude computer graphical artists in terms of realistically modeling the nuances in shape deformation and color such as human faces. A third advantage of Image-Base Modeling and Rendering (IBMR) is that rendering time is independent of scene complexity. For example, consider an urban street scene. To model this traditionally, one would need to specify the polygons for every building, every window and door on the building, every parking meter, the cars in the scene and so on. This can quickly becomes millions of polygons that must be transformed, clipped, lighted, and rendered. The number of primitives quickly exceeds the number of pixels on the screen. Alternately, one can simply capture the scene with a camera and then quickly render it with some shape deforming transforms to give a realistic display that is independent of the geometric complexity of the objects in the scene. Below are some of the major milestones in the field.

An early technique was to take movies of a scene from every displayable view. These pre-stored images could be accessed randomly giving the user the illusion of being able to navigate through a space. These “movie maps” could be natural scenes collected with conventional cameras (Lippman, 1980) or synthetically generated scenes such as the Virtual Museum (Miller et al., 1992). To achieve smooth panning, this technique requires a large amount of storage.

Environment maps have been used to create interactive panoramas. If a camera is rotated about its optical center but not translated, then all views of the camera can be mapped into a single view-independent canonical projection. Later, views in any direction from that rotation point can be generated by reprojecting portions of the environmental map (Chen, 1995; Hodges and Sasnett, 1993).

An important step in the field of IBMR was that of “View Interpolation for Image Synthesis” (Chen and Williams, 1993). They were able to reduce the number of images needed for a “movie map” by interpolating between points of view. In their work, Chen and Williams generated a 3-D mesh of images of a virtual museum. The images were pre-rendered using progressive radiosity, a very computationally intensive and hence a non-real-time technique. Also, disparity maps were generated between adjacent images automatically from the known camera and vertex locations. Then in real-time, a continuous motion sequence was generated by morphing adjacent images together. Chen and Williams made the observation that if the camera moves in a plane that is parallel to the initial image plane, then linear interpolation of the pixel locations produces exactly correct results. This observation was also made by (Seitz and Dyer, 1996) and it is central to the algorithm presented in this dissertation.

(McMillian and Bishop, 1995) came up with a general scheme for classifying IMBR techniques. They describe a “plenoptic” function that has six degrees of freedom. The six parameters consist of three to describe a location in space, one for time, and two more to describe the location on a sphere. A complete plenoptic would describe what is visible at every point in time. With this framework for analysis, they categorize existing IBMR systems and also present their own system consisting of a collection of cylindrical projections.

QuickTime (reg.) VR is an early and well-known commercial IMBR system (Chen, 1995). QuickTime VR provides for two kinds of "movie" formats. The first is panoramic in which the viewer is in the center of a enclosed space looking outward. Alternately they also provide for object movies where the view is located outside of an object viewing towards its center. They also provide a suite of tools to allow the user to collect a range of images and then stitch them together semi-automatically.

The above papers described techniques for collecting and the viewing of panoramic images that contain no explicit 3-D information. Much work has also been done in the area of collecting images from real world scenes and then reconstructing the actual 3-D geometry of the objects using vision techniques. Once the 3-D geometry is recovered, it can be used in the traditional computer graphics pipeline. An in-between approach is to use depth maps where partial 3-D information is recovered and maintained. Richard Szeliski's "Image Mosaicing for Tele-Reality Applications" (Szeliski, 1994) is one example of recovering a dense depth map of a scene using vision techniques on a collection of video images.

Havaldar, Lee, and Medioni in their paper "View Synthesis from Unregistered 2-D Images" (Havaldar et al., 1996) describe a method for recovering a full 3-D model of a scene by correlating camera images from various points in space. In their work, they make use of projective invariants. They exploit the constraints of epipolar geometry to limit their search space when making correspondences. Their system is mostly automatic but needs some human intervention.

Moezzi, Katkere, Kuramura, and Jain present a novel method for creating an immersive video experience (Moezzi et al., 1996). In their paper "Reality Modeling and Visualization from Multiple Video Sequences", they describe a method for combining the video input from several cameras in such a way that user can move a virtual camera around the scene. Their system, "Immersive Video", depends on having a 3-D model of the static background scene which is created before the performance is recorded. The system then builds a voxel (Foley et al., 1991) representation of the dynamic elements of the scene by casting rays from each of the cameras into the scene. Areas in the scene that differ from the static scene are combined from the multiple cameras to create a model of the participants. Both the geometry and the color are recorded as part of the model. Once the model has been created, it can be displayed from arbitrary points of view.

A sub-area in IBMR deals with the application of 2-D and 3-D warps to planar images to create novel points of view near the original viewpoint. A work that provides a good summary of the previous work in this area is that of Mark, McMillan, and Bishop (Mark et al., 1997). Their technique makes use of the Z-buffer of a conventional graphics pipeline to provide depth information to facilitate the re-rendering of the image. A commercial system whose goal is to provide interactive frame rates on commodity hardware is the Talisman system from Microsoft (Torborg and Kajiya, 1996). This system breaks the 3-D scene into a collection of 2-D layers. These 2-D layers are warped between successive frames until an error threshold is exceeded at which point the layer is re-rendered using traditional computer graphics. A good reference on the mathematics of warping is Wolberg's book "Digital Image Warping" (Wolberg, 1990).

2.3 Systems for Generating Virtual Heads

This section describes systems that are the closest to view morphing system implemented by this dissertation. While the system described by this dissertation was not theoretically limited to views of the head, this is a strong practical example of its use. This section is broken down into the following sub-sections: morph-based facial rendering, model-based facial rendering, and video avatars. The section on video avatars is the most closely related to the work presented in this dissertation.

2.3.1 Morph-Based Facial Rendering

A popular application of morphing (Beier and Neely, 1992), even from its initial use, has been morphing between faces. Even morphing theoretical papers such as "Polymorph: Morphing Among Multiple Images" (Lee et al., 1998) or "View Morphing" (Seitz and Dyer, 1996) use faces as the basis for their examples because facial morphs generate realistic faces. Below is a sampling of technical contributions that make use of morphing in the synthesis of computer-generated faces.

(Ezzat and Poggio, 1996) in their paper "Facial Analysis and Synthesis Using Image-Based Models" create a computer model of the human face that consists of a 3x3 grid of facial images which does not make use of any 3-dimensional models. Their system is capable of modeling both rigid facial transformations such as changes in pose as well as non-rigid transformations such as smiles. The system takes video frames as input, performs feature analysis, synthesizes an image by performing 2-D warps on the 3x3 example network, performs feature analysis on this generated image, and then repeats the process to minimize the error in the feature parameters. Their system produces realistic images that match the pose and expression of the original face. Their system is fully automated but has the limitation that it takes on the order of minutes to analyze and generate each frame.

(Pighin et al., 1998) in "Synthesizing Realistic Facial Expressions from Photographs" collect images taken from multiple viewpoints of a variety of expressions. For each expression, they manually map critical points from the images to an existing polygonal model of the head.

Instead of creating a single view-independent texture map for each expression, they suggest a view-dependent texture mapping mechanism that combines varying contributions of multiple textures depending on the orientation of the surface normal resulting in a higher quality texture map. Their system is used to generate smooth animations between the various expressions recorded. The morphing performed by this system is based on the 3-dimensional models rather than just the 2-dimensional images resulting in very realistic animation sequences.

(Guenter et al., 1998) “Making Faces” describes a system which collects video sequences of a human actor from multiple cameras and then combines those images to create a 3-dimensional texture mapped model of the video that can be played back from novel viewpoints. This system, like the previous, uses a 3-dimensional model of the head and hence uses a 3-dimensional morph of the geometry to create the final image. This system differs in that the extraction of the geometry is automated. The authors glued 182 dots of six different colors on the participant’s face. These dots were automatically registered between the different cameras and automatically tracked between frames. An automated process removed the dots from the final image. Because of accurate knowledge of the texture registration between frames, the authors were able to achieve the relatively low bit rate of 240 Kbits per second for a 3-dimensional 300x400 animation that can be viewed from any angle.

Ezzat et al. have developed a system for generating “videorealisitic speech (Ezzat et al., 2002) animation”. After capturing video of the subject speaking, analysis of the mouth positions with phonemes is performed to create a *multidimensional morphable model* (MMM). This model can be used to generate new lip-synched speech. They use automatic methods for extracting the facial expressions for a set of phonemes once a facial map of the eyes and mouth has been defined. Although natural eye and head movements were added to the model to make it look more life like, emotional responses were not part of the system.

Video Rewrite (Bregler et al., 1997) takes video of the actor speaking and creates a video database of mouth movements associated with phonemes. Taking video clips of three phonemes, triphones, Video Rewrite uses morphing techniques to blend the transitions of phonemes to lip synch with novel speech. To account for all possible triphones, a large number of video sequences are needed.

These systems generate images of faces that are life like. Some of the systems have been used to create video conferencing systems of extremely low bit rates 0.57 Kbps if the model is already at the remote location (Eisert and Girod, 1998). After having gathered and processed the initial image capture, these systems can be used to generate new image sequences that are more life-like than traditional models.

The "Eyes Alive" (Lee et al., 2002) paper describes work to make the eyes of facial models more realistic. Using eye-tracking technology, they develop statistical models about eye movement under different scenarios, (e.g. listening versus talking). Generating a model of eye movement from the statistics, they use it to more realistically animate their synthetic head.

Pighin et al. used morphing techniques to create photo realistic textured 3D models of a range of various facial expressions (Pighin et al., 1998). Using photographs, they employed user-assisted operations to recover camera pose and to build a 3D model of the subject. Then using 3D shape morphing and blending of the textures from the photos, an interface was provided from combining various levels of the captured emotional states.

2.3.2 <u>Model-Based Facial Rendering</u>

An alternative to image-based facial coding is model-based facial coding in which just the parameters that defined the pose and facial expressions are conveyed to the facial-image rendering engine. This approach is presented to contrast it with the one implemented by this dissertation. The advantage of the model-based approach is extreme efficiency in the amount of data transmitted. (Eisert and Girod, 1998) with about 10 seconds of processing per frame and using extensive compression were able to achieve the incredible rate of 47 bits per frame. They used the MPEG-4 Synthetic-Natural Hybrid Coding (SNHC) group's proposal (SNHC, 1997) to parameterize the facial expressions.

Many of these systems use deep models for simulating the facial deformation. Deep models simulate the real anatomy of the human body by emulating the operation of muscles under an elastic skin. The image-based systems in contrast only distort the image surface. Example of deep model systems are (Terzopoulos and Waters, 1993), (Roivainen and Forchheimer, 1993), and (Takeuchi and Nagao, 1993)**.** A disadvantage of model-based systems is a lack of realism or spontaneity in the generated images.

2.3.3 <u>Video Avatars</u>

An approach to overcome some of the limitations of the model-based facial image coding systems is to show live video of the subject and immerse it into a 3D scene. Compared with the model-based image coding approaches, these systems have much greater bandwidth demands and only display what the actor has expressed. The system described in this dissertation belongs to this category. In its most naive form, placing a video avatar into a 3D scene could consist simply of inserting an unaltered 2D video stream into a region of the final rendered display.

In ClearBoard (Ishii et al., 1994) the usefulness of interacting with 2D video images of the participants was demonstrated in a collaborative white board environment. Communication was enhanced through everyday cues such as hand gestures, head movements, and gaze direction.

In the InterSpace system (Sugawara et al., 1996) live video images of the participants' faces are placed over 3D polygonal avatar bodies. While being immersed in a 3D world, remote users are able to communicate using their own voice and facial expressions.

(Wang, 1998) did work with two kinds of video avatars. The first were with 2D video avatars. By performing a blue screen segmentation of the background, a flat image is inserted in to the scene. Its perspective is distorted for various points of view. Wang investigated an augmented version of this work with a 2 1/2 D system. In the improved version, a video image was projected onto a static head model using a projective texture matrix, this allows the view from the camera to be mapped back onto the head model.

(Insley, 1997) created full body, static avatars by creating a set of silhouetted still images captured from video taken from a 360 degree rotation around the individual. When navigating through the 3D immersive CAVE® VR environment, the image nearest to the user's viewing angle is projected onto a single flat polygon.

Full body dynamic video was inserted into a Multimedia Multi-User Dungeon 3D environment (Sakamura et al., 1999). The problem of navigating around the video image was handled by displaying the image with the closest angle to the viewer of four video streams taken 90 degrees apart.

Live 2D video images of the users head have been mapped to 3D polygonal models in the CAVE® environment (Cruz et al, 1993). Precise position tracking devices are used to identify the location of the user's head. The head is segmented from the video by either chroma-keying or digital subtraction from a known background. Projecting the resulting texture onto either a generic head model or a specific laser scanned model creates a more accurate and believable 3D image (Wang, 1998), (Rajan, 2001), (Rajan et al., 2002).

Using stereo cameras and calculating a depth map of the video input, 2.5D video avatars have been created that can be rotated in 3-space giving an enhanced sense of presence (Ogi et al., 2001), (Hirose et al., 1999). The depth maps only provide a limited range that the avatars can be rotated but this has been augmented with the use of multiple video and stereo-depth cameras.

Another example of using a depth map to create video avatars was developed at the Electronic Visualization Laboratory (Sandin et al., 2000). The 3D model of the person is obtained using depth from stereo. The range information obtained is used to segment the background. Using a 3D meshing algorithm, the surface of the 3D avatar is reconstructed from the point cloud data of the foreground. Since the model of the avatar is constructed in real-time, the 3D shape or size of the person need not be known. Thus this method can be used to generate more realistic video avatars

A hybrid system that combines a model-based approach with video images has been used in a video conferencing system (Leung et al., 2000). A subset of the video of the user's face (eyes and mouth) is sent over the transmission link allowing for lower bandwidth communication. In this implementation, both the original 3D model along with targeted regions of the user's face were manually segmented.

3. ALGORITHM

The main steps of the algorithm for this dissertation can be summarized as follows. First, the images from two video streams must be acquired and the subject segmented from the background. Next is the most important piece of the view morphing process, establishing a correspondence between the images. After the correspondence has been created, each image is warped to an in-between view, blended with each other, and then displayed to the screen. These steps can be separated into either belonging to computer vision or computer graphics.

Computer Vision (analysis)

1. capture synchronized images
2. segment out the subject
3. establish a correlation between corresponding parts of the images

Computer Graphics (synthesis)

4. view morph the desired image
5. display

Of the five tasks above, only item three is not straightforward, establishing the correspondence between the two images. The challenges for this operation are two-fold. First an accurate correlation must be performed. Second, the analysis must be preformed fast enough so that it is completed before the next frame needs to be rendered. Realistic looking view morphing has been demonstrated in the past by using manual human identification of the inter-image correspondences (Seitz and Dyer, 1995; 1996). To meet the needs of real-time streaming video, a completely automated solution is needed.

3.1 Foundational Techniques From Computer Vision

The field of computer vision has a long heritage with a rich mathematical basis. In developing this dissertation, that heritage was used as a foundation. This following set of subsections briefly describes the most important theories related to this work.

3.1.1 View Morphing

The key technique that serves as the basis from which the work in this dissertation is built upon is View morphing. The view morphing term was coined in the paper by Seitz and Dyer (Seitz and Dyer, 1996). It is a technique for generating physically valid in-between views of a 3D scene by only using two 2D images of that scene. The view morphing technique is similar to a variety of other work done in the same period known as view-interpolation. To generate the in-between image, the original 3D model is not constructed. Instead, the 2D images are first rectified, morphed, and then re-projected onto the destination image plane. By performing only 2D operations, valid novel views are created.

At the heart of the view morphing technique is the need to find where points from one image map to in the other image. In (Seitz and Dyer, 1995; 1996), this finding of corresponding points was a manual process using human guidance. The rendering of the images was an off-line process that took many seconds to perform.

The goal of this dissertation is to take this technique and apply it to video streams. A pre-requisite of processing video streams is that the process be at least semi-automated given the large number of images. A second goal of this dissertation is to perform this work in real-time. This second goal necessitates the complete automation of the matching operation. The task of fully automating the matching process is, in general, an unsolved problem. Constraints were put on the input to make the system more realizable, (see section 3.1.3 that deals with monotonicity). However, it is the real-time constraint that presents the most difficulty in deriving the final solution. Through a variety of innovative techniques, a viable solution to performing real-time view morphing on video streams is found.

3.1.2 Matching

Vision algorithms can generally be divided into being either feature-based or correlation-based. In feature-based algorithms, a sparse set of specific features such as lines and edges are detected in the images. Measures are made of these features in the source and destination images to choose the set of correspondences that possess the minimal distance. In contrast, correspondence-based systems measure the differences between regions in two images by perturbing the location of a region in one image and comparing it to a fixed location region in the other image in an iterative process. The system of this dissertation belongs to the family of correspondence-based systems. Choosing to implement a scan-line algorithm, in the interest of performance, greatly limited the set of useful features that could be extracted.

Feature-Based Methods

- Restrict search for correspondences to a sparse set of features
- Typical features include edges and lines
- Characteristics of features include length, orientation, and average intensity, contrast, and gradients
- More complicated features could include eyes, nose, and mouth
- Matching is very domain specific
- Generally more immune to lighting differences, both ambient and spectral, than correspondence based methods

Correlation-Based Methods

- Perform a dense disparity search across a limited region
- Correspondence measures are defined for the correspondence windows
- Can be used over a broad range of subjects
- Generally simpler to implement
- Without normalization, lighting differences can generate errors

The goal of the proposed matching system is to find the mapping that will take points from one image to the other. In general, this mapping is arbitrary as long as it adheres to the monotonicity constraint, see section 3.1.3. When viewing the effect of the projection of the 3D surface to the 2D image planes at the level of a scan-line, it can be seen that the image content is both shifted and possesses a variety of dilations and contractions. Hence the goal of the matching system is to find that initial shift and then to find the distortions that map one line to the other. To compute the correlation between scan-lines, a Sum of Squared Differences (SSD), operation is performed.

The SSD operation is a mechanism for calculating the error between two entities. It can be thought as a method for calculating the distance between two points in a N-dimensional space. The N intensity values in a scan-line can be thought of as an N-tuple representing the coordinates in N-space. The SSD operation can be thought of as a generalized Pythagorean Theorem and is described below. Consider two scan-lines whose values are represented as N-tuples S0 = (p01, p02, p03, … , p0N) and S1 = (p11, p12, p13, … , p1N). The SDD operation is defined as follows.

SSD = sqrt((p01-p11)^2 + (p01-p11)^2 + … + (p0N-p1N)^2)

3.1.3 The Monotonicity Constraint

An important assumption in the implementation of view morphing is that of the monotonicity of the images (Seitz and Dyer, 1995). Simply stated, two images are monotonic with respect to each other if whenever point P_0 is to the left of point P_1 in image I_0, then point P_0 is to the left of point P_1 in image I_1. This constraint is met by objects whose width to depth dimensions are roughly the same or less and do not possess extreme protrusions or indentations. When panning left to right, this constraint is met when viewing a finger pointing straight up but fails when the finger is pointed towards the camera.

This constraint fails for points that are occluded in one image but not the other. In terms of the human face, the nose can produce this problem. Fortunately, failure of the constraint in a localized section of the input images only produces failures locally in the output image. If only small regions of the total image fail the constraint, the results are generally acceptable.

The monotonicity constraint has a positive implication on the search complexity of the problem. Consider the point $\mathbf{p}_{L0}$ to the left of point $\mathbf{p}_{R0}$ in image I_0. Once having found the point $\mathbf{p}_{L1}$ in image I_1 corresponding to point $\mathbf{p}_{L0}$, one knows that they only need to search to the right of $\mathbf{p}_{L1}$ to find the point corresponding to point $\mathbf{p}_{R0}$.

3.1.4 Epipolar Geometry and Scanline Search

The general correspondence problem between two images is computationally quite large. A single pixel P_0 in the first image I_0 could map to an arbitrary point in the second image I_1. If **N** is the number of pixels along either dimension of a square image, then the general search problem is an $O(N^2)$ problem. An initial step in speeding up the search is to transform the images so that search is can be restricted to that along a scan-line thereby reducing the problem to a complexity of *O(N)*. In practice, the whole scan-line (or whole image) would not need to be searched. The range of the search depends on many parameters such as the angle between the two cameras and the geometry of the subject, but is generally much smaller than the full theoretical search area.

By knowing the location of the camera centers, orientation, and image planes this problem can be reduced to search along a line by using the properties of epipolar geometry, see Figure 7, (Trucco and Verri, 1998). The three-dimensional point **P** in the real scene and the optical centers of the two cameras $\mathbf{O_0}$ and $\mathbf{O_1}$ define a plane. The lines l_0 and l_1 that are formed where this plane intersects images I_0 and I_1 are the epipolar lines of **P**. Epipolar geometry states that the projection of **P** onto images I_0 and I_1 (i.e. points $\mathbf{p_0}$ and $\mathbf{p_1}$), must lie on the epipolar line of that point in the corresponding image. Therefore, given the point $\mathbf{p_0}$ in image I_0 and the ability to create its associated epipolar line in I_1, one only needs to search along that line. Hence, by exploiting epipolar lines, what was a 2-dimensional search problem is reduced to that of search along a line thereby vastly improving the efficiency.

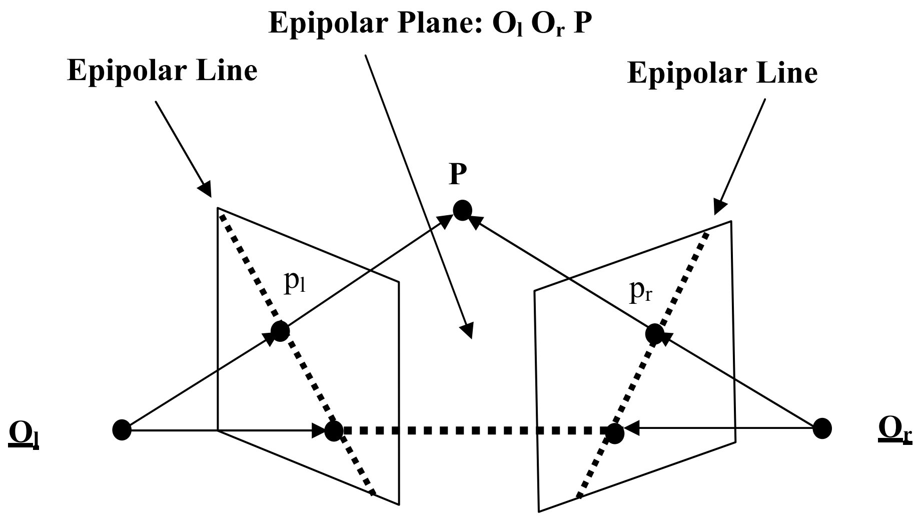


Figure 7 Epipolar geometry.

3.1.5 Scan-Line Rectification

The epipolar lines in the previous section do not, in general, lie on horizontal scan-lines. This means that searching the epipolar line involves jumping from scan-line to scanline. When moving diagonally though the image, the original image data with respect to the epipolar line is not uniformly sampled. This will likely incur the additional processing of re-sampling the data to get it into some kind of regularized format.

However, it is possible to re-project the images such that the epipolar lines align with the scan-lines of the images in memory thereby facilitating efficient memory referencing on a computer (Seitz and Dyer, 1996). From a programming point of view, this has the effect of simply traversing a 1-dimensional array in the search of correspondences.

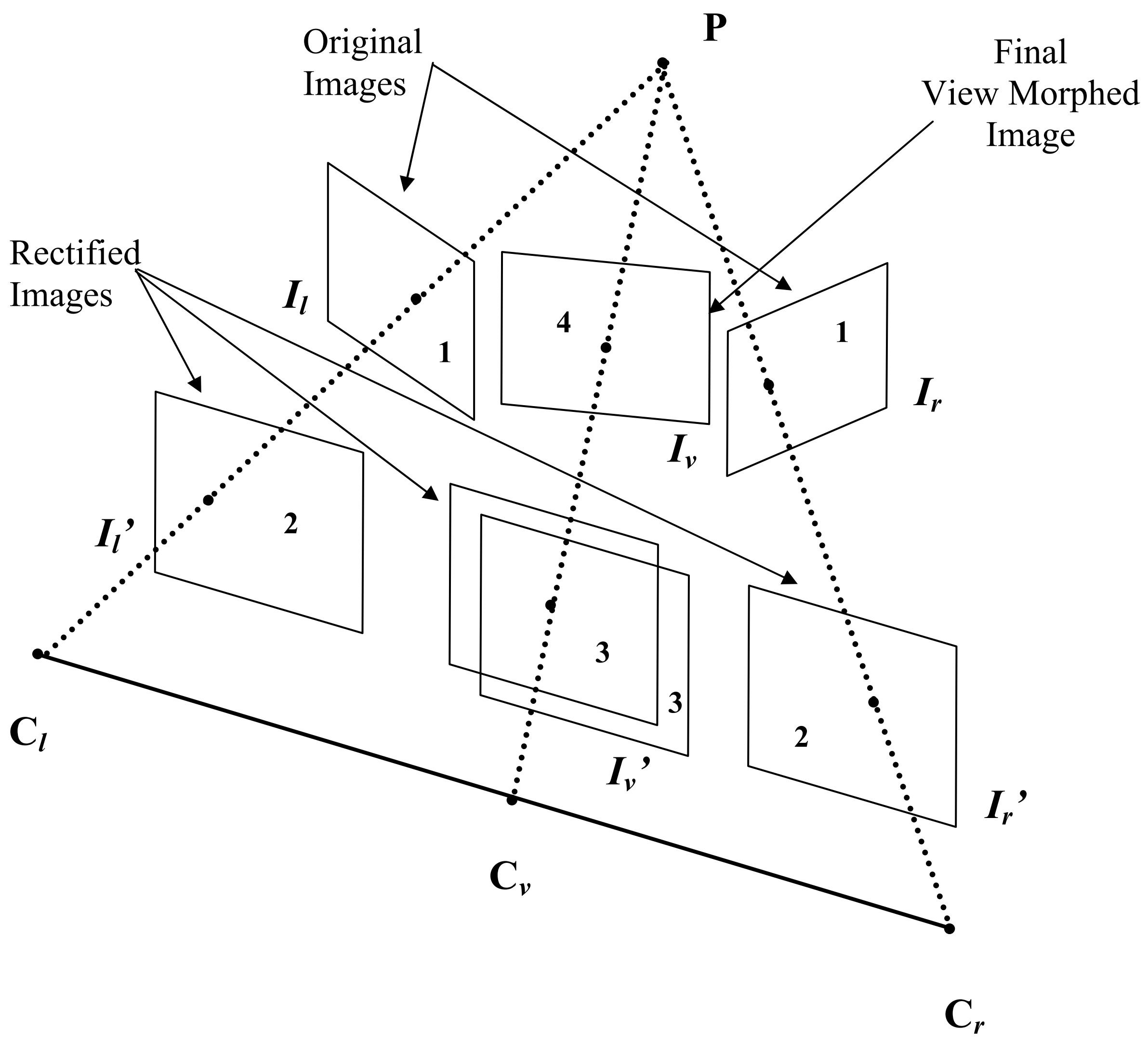


Figure 8 Diagram of the image rectification and view morphing processes.

3.1.6 <u>The View Morph Process</u>

The view morphing operation is accomplished through a series of steps as diagrammed in Figure 8. The types of images generated as part of the view morphing process are classified as follows.

1. The images as captured by the cameras.

2. The images after they have been rectified to lie in the same plane as each other.
3. The images after corresponding pixels have been linearly interpolated in proportion to the location of the virtual camera.
4. The alpha blended image of step three after it has been re-projected to the image plane of the virtual camera.

Let the subscripts ***l***, ***r***, and ***v*** refer to the left, right, and virtual images/cameras respectively. Let $\mathbf{C}_l$ and $\mathbf{C}_r$ denote the left and right real cameras and $\mathbf{C}_v$ denote the virtual camera. In step 1, the images $\boldsymbol{I}_l$ and $\boldsymbol{I}_r$ are acquired into the system from the video cameras. In step 2, in order that efficient scan-line processing be performed in the matching phase, the images are rectified producing images $\boldsymbol{I}_l'$ and $\boldsymbol{I}_r'$.

After a correlation has been established for corresponding points in the two images, these points are geometrically linearly interpolated in proportion to the location of the virtual camera. This creates two separate images that have been warped. These images are then alpha blended in proportion to the location of the virtual camera producing $\boldsymbol{I}_v'$.

The final step of the view morphing process is to project the rectified image of the virtual camera to the image plan that is in orientation with the virtual camera producing $\boldsymbol{I}_v$.

3.1.7 <u>Scan-line Search</u>

The key challenge of performing view morphing is finding the correspondence between points in the two images. When dealing with the 2D projection of a 3D scene of an unknown object, the correspondences will be distributed, in general, is some kind of non-uniform manner. Point correlations are determined by systematically distorting one scanline and then checking how well it matches the second scan-line. Having no specific knowledge about the subject, how they have oriented their head, if they are wearing glasses, the size and shape of the nose and other structures, this distortion can be arbitrary. Fortunately, by taking advantage of the monotonicity constraint (section 3.1.3), the search along the scan-line is reduced.

Figure 9 shows the nature of the matching problem. While the order of the points on the scan-line projection is the same, the distance between them varies. The goal of the algorithm is to distort the scan-lines from one camera so that they appear like the scan-lines recorded in the other camera. Once two scan-lines appear the same, the mapping between them is known. In the example in Figure 9, **AB** in the left image is much smaller than **AB** in the right image, while **BC** is about the same in both images, and **CD** is larger in the left image.

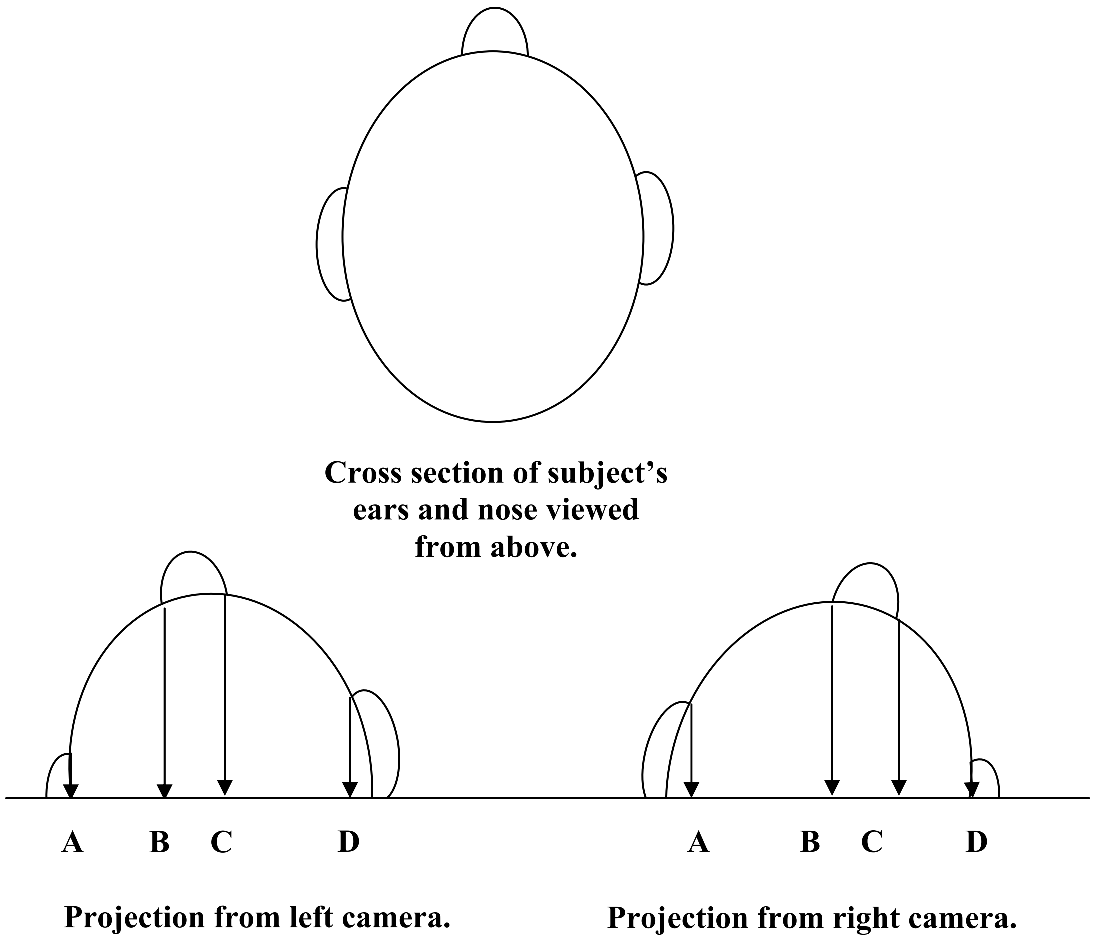

Cross section of subject's ears and nose viewed from above.
A
B
C
D
Projection from left camera.
A
B
C
D
Projection from right camera.

Figure 9 Projections from the left and right cameras of the subject onto a scan-line illustrating the nature of the search problem.

The advantages of scan-line algorithms are well known. Various reasons for choosing scan-line operations are: they are easier to implement than multi-dimensional procedures, they have smaller memory requirements, and they generally run faster. There are also disadvantages to using a scan-line algorithm for matching. The main concern is that because less information is available to the matching routine, there is little context for performing the match causing ambiguities or errors to occur.

3.2 Problem Space Transformation – Normalized Cylindrical Projection

Performing matching on images of the head that are separated by any significant angle proves problematic not only because features go in and out of view when rotating around the head, but also because those features become distorted when the view of the feature changes from straight on to askance. To mitigate this problem, the images are transformed into a data representation where the correspondence between them is more easily identified.

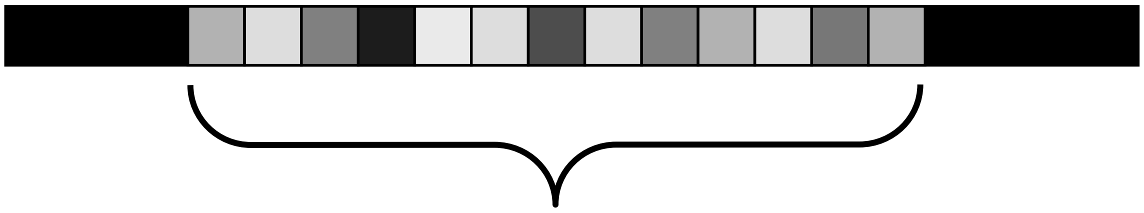


Figure 10 Simple scan-line geometric metric, the diameter of the segmented region.

Viewing a single arbitrary scanline of the segmented subject, the only *geometric* measurement that can be made with confidence is the length of the scan-line, see Figure 10. Taken from a camera that is oriented towards the subject’s head, that scan-line would correspond to the projection of a cross-sectional slice of the head. A simple assertion is now made: “A cross-sectional slice of a head is more like a circle than a straight line.” This assertion is the basis of the Normalized Cylindrical Projection (NCP) transform and will be referred to as the “NCP Assumption”, see Figure 11.

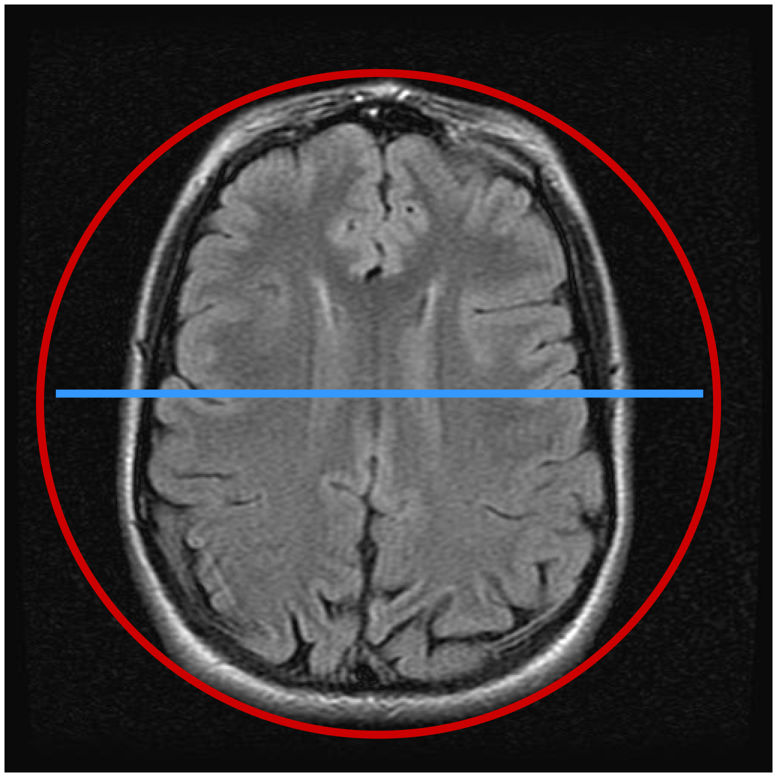


Figure 11 The "NCP Assumption", a cross-section of the head is more like a circle than a line.

An implication of the NCP assumption is now derived. If a cross-sectional slice of the head were a circle, then half of that circle would be projecting on to the scan-line. Using this logic, one can generate a more accurate representation of the original object by taking the length of the scan-line of the segmented object and using it as the diameter.

As a first step in the NCP operation, each of the scan-lines is projected onto a semi-circle and then laid flat. To account for possible differences in the focal lengths of the cameras, and to more optimally use the memory in a 2-dimensional data structure in RAM, a normalizing step is performed. Whereas the original segmented object possesses background pixels on its left and right, the normalization step independently scales and shifts each original scan-line so that it fills an entire row in the data structure for the transformed image. In summary, the original segmented image is, on a per scan-line basis, projected onto a semi-circle, flattened, and then scaled so as to occupy a full scan-line, see Figure 12 and Figure 13.

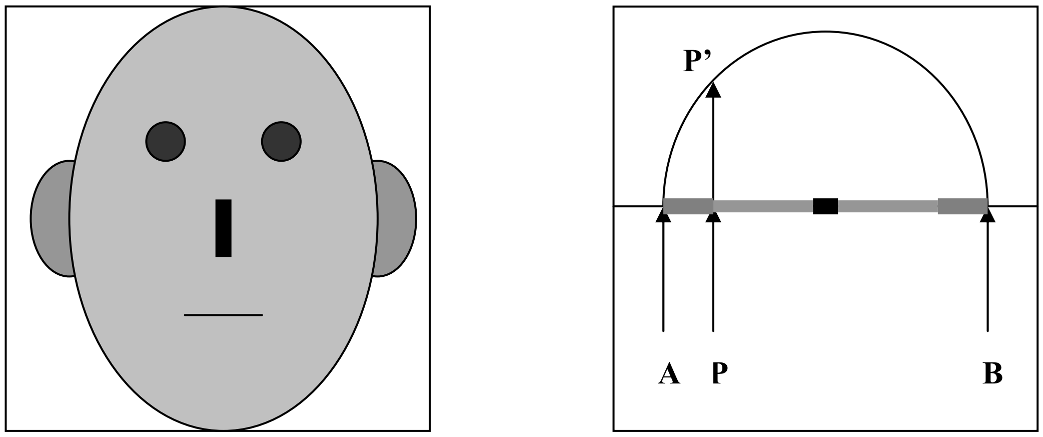


Figure 12 Left: diagram of the subject. Right: scan-line that passes through the ears and nose. Point P is projected onto a semi-circle at point P'.

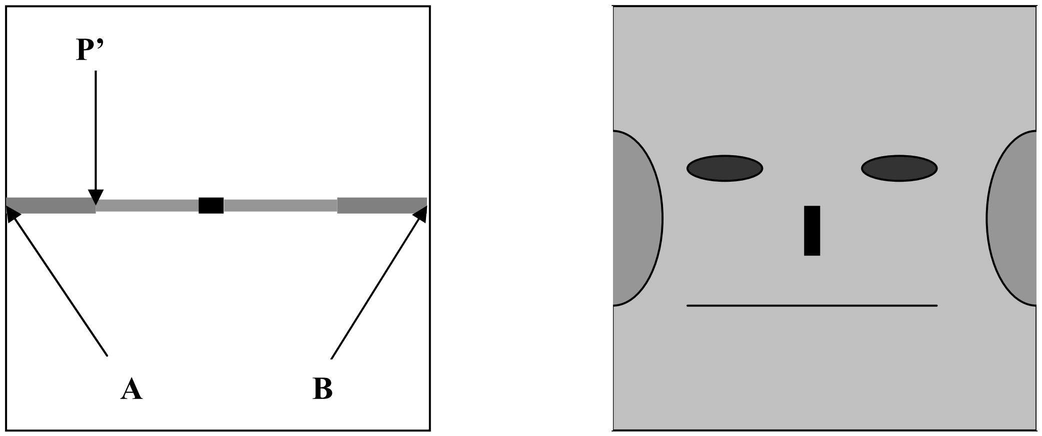


Figure 13 Left: the newly rendered scan-line after the semi-circle has be laid flat. Right: the resulting image after all the scan-lines have passed through the NCP transformation.

The effect of the transformation is to severely stretch the regions near the outer edges of the segmented subject but to leave the regions near the center relatively undisturbed. The algorithm possesses geometric simplicity but is surprisingly effective at providing a much improved starting point at which to perform the pattern matching. Consider the images below generated for a real subject, Figure 14 and Figure 15.

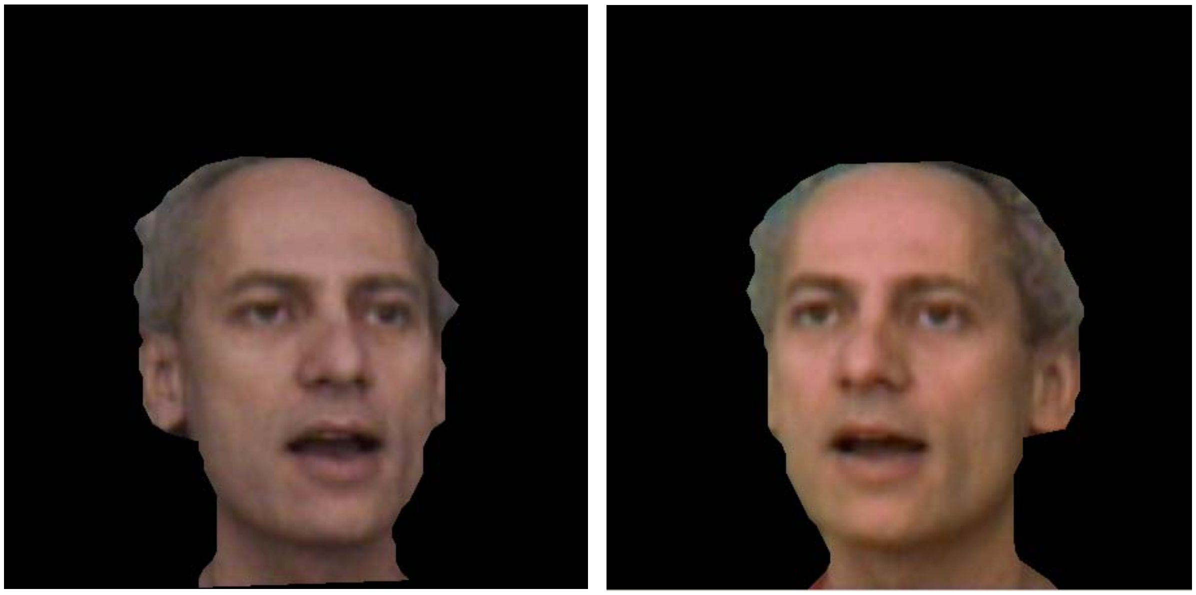


Figure 14 Left and right view of the subject.

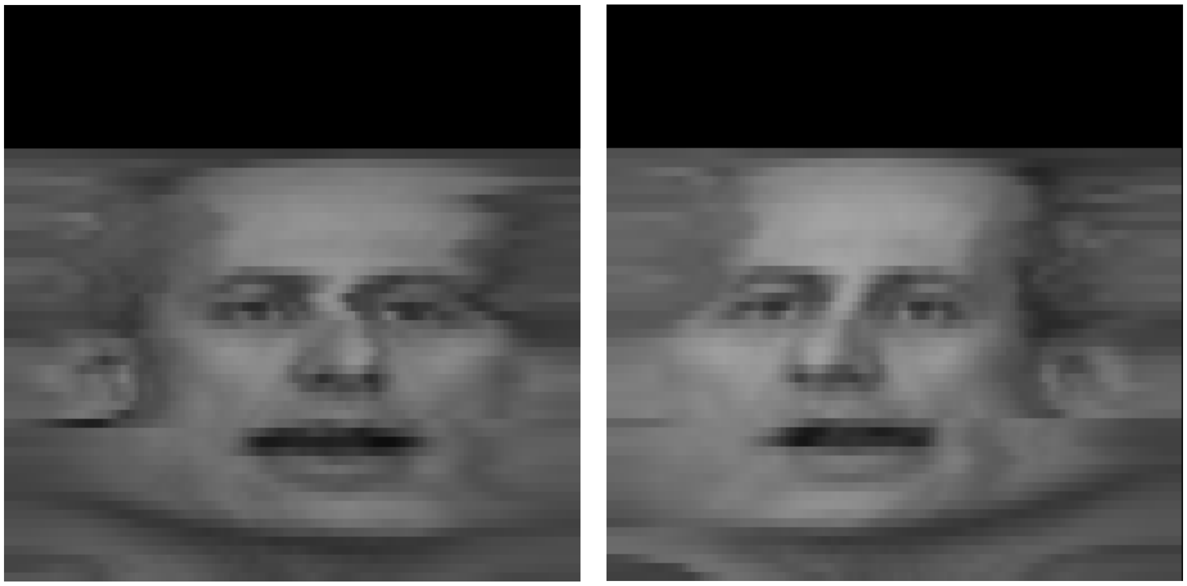
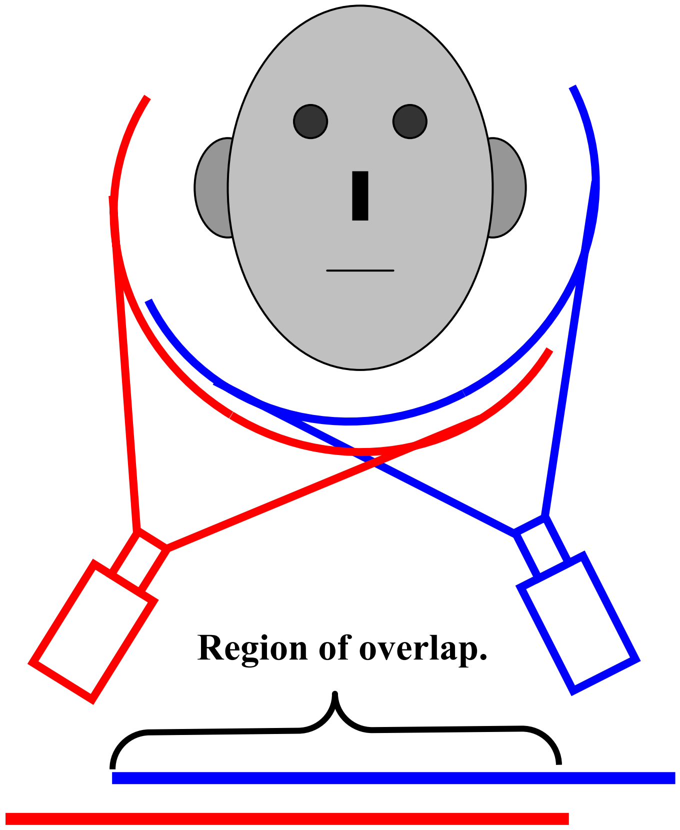


Figure 15 Left and right views of the subject after the NCP transform has been applied.

Taking the corresponding transformed scan-lines from a pair of images and shifting them to account for the rotational differences of the two views, creates a better starting point for performing pattern matching than simply laying the corresponding scan-lines on top of each other before the transformation, see Figure 16 and Figure 17. Before trying to perform the image correlation process, the images are shifted relative to each in proportion to the angle between the fixed cameras as implied by the NCP transform. The shift amount is directly related to the amount of a semi-circle the camera separation covers. For example, if the video cameras were separated by 45°, that would be 25% of a semi-circle. . The images should then be shifted linearly in NCP space by 25%. The effectiveness as a starting point for performing image alignment can be seen in Figure 17 where the *Null* NCP match is performed. This consists of merely aligning the images in NCP based on the default shift and then applying the inverse NCP transform.



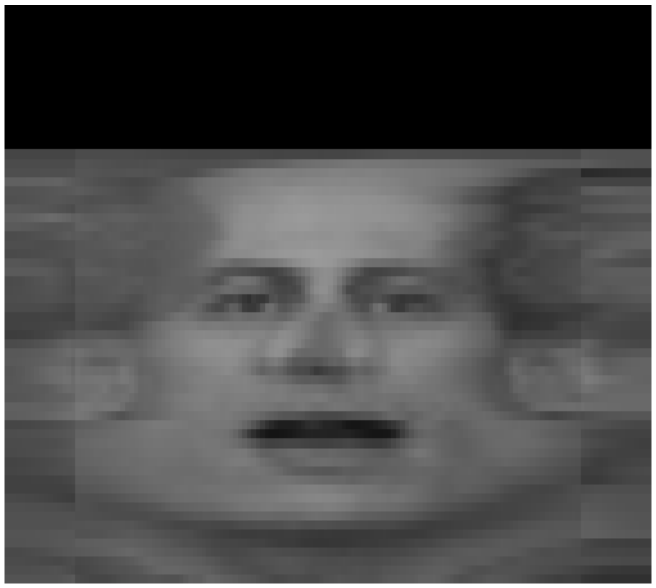


Figure 16 Process of applying NCP transformation with simple shift and then blending the left and right images.

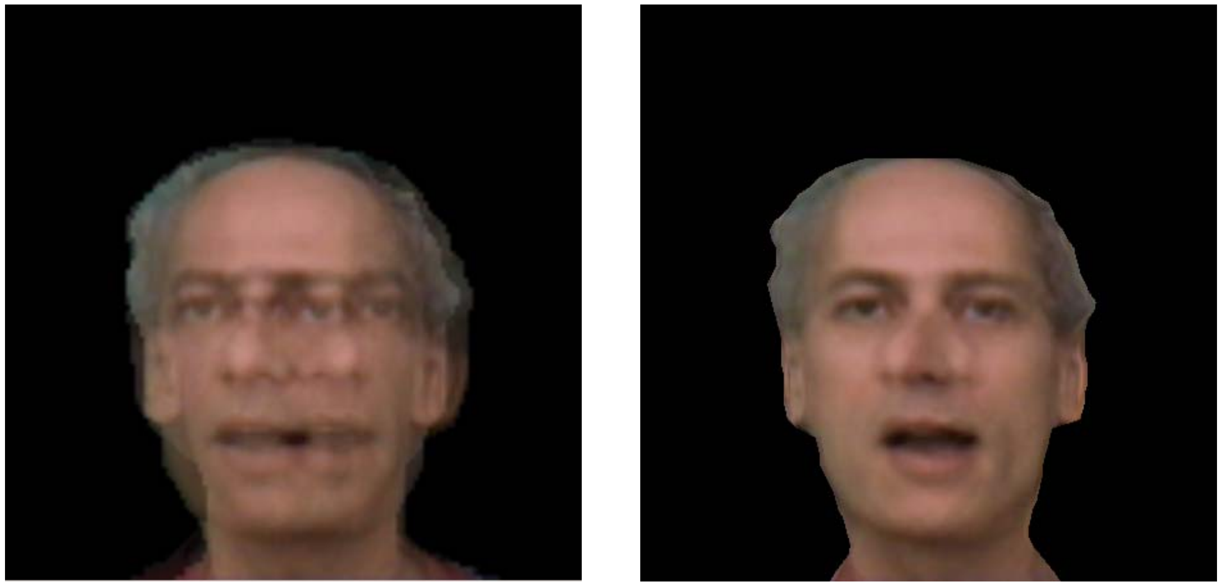


Figure 17 Comparison of video stream overlap, left, versus "Null" NCP alignment, right.

The NPC operation is viewed as a *transform*. The problem being solved is that of finding the corresponding points of one image in the other. The NCP operation does not solve that problem. What it does do, importantly, is make that problem easier. That easier problem is first solved and then the inverse NPC transform applied to get that information back into the original image space. The steps for performing the image alignment are:

1. Convert the input data from image space to NCP space.
2. Perform the correlation matching in NCP space.
3. Apply the inverse NCP transform to get the match information back into image space coordinates.

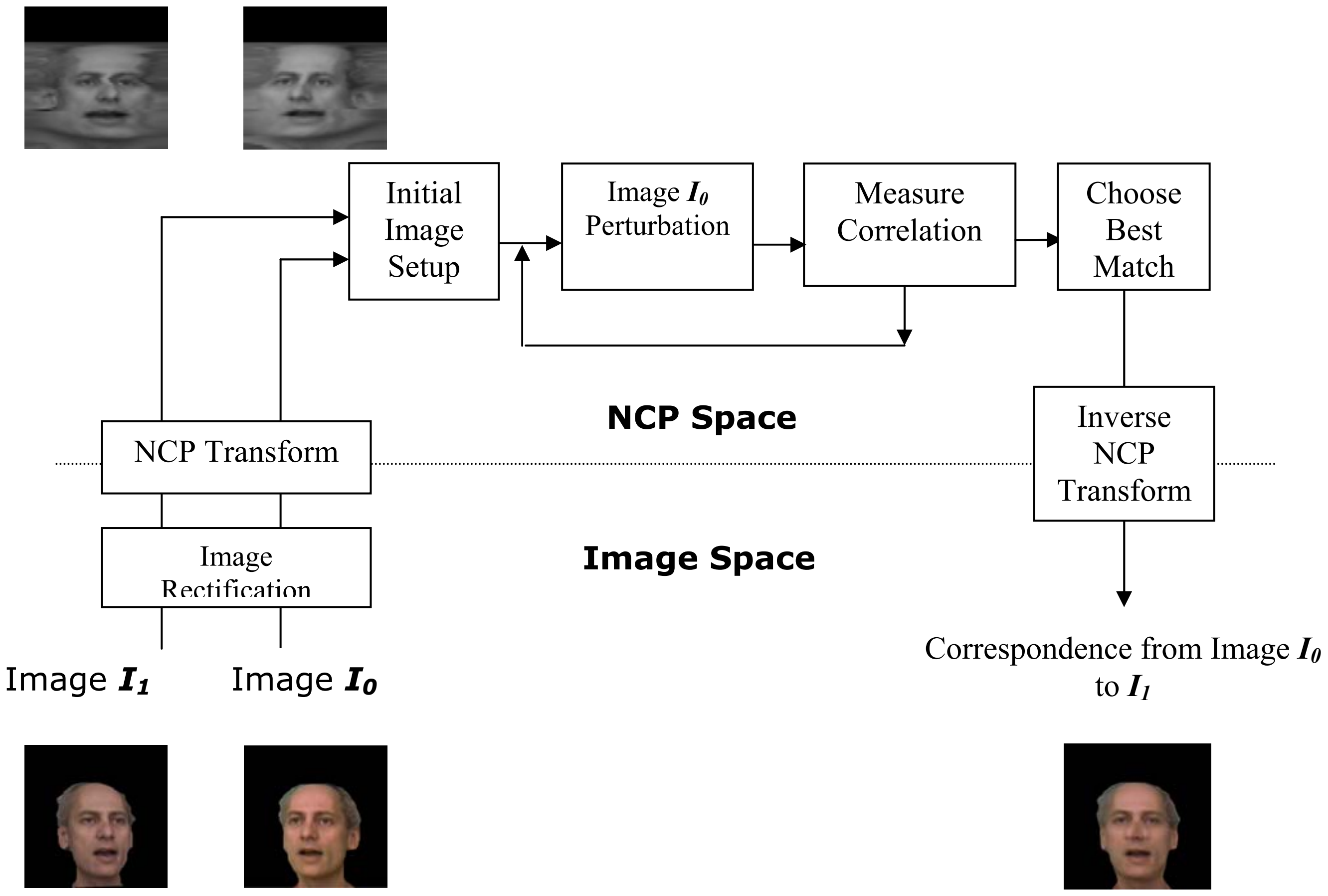


Figure 18 Diagram of the NCP transform and image correlation process.

3.2.1 <u>Per Scan-Line Initial Shift Adjustment</u>

The performance of the system depends explicitly on making good guesses as to what should be the starting point for the correlation generation. The default initial shift is based on the assumed geometry of the subjects (see section3.2). This initial starting point is only a crude guess. Empirical tests have shown that making adjustments to this default shift on a per scan-line basis is an inexpensive way to create a better starting point for the more complicated matching operations to follow. Adjustments are made to the default shift by making multiple adjustments to the left and right of this starting point and choosing the one that has the smallest SSD value.

3.3 Combinatorial Complexity

Having aligned the images so that corresponding epipolar lines lie along scan-lines, one now needs to search along the scan-lines to find corresponding points. That is, one needs to find a set of perturbations to the scan-line that will map it to the corresponding scan-line in the other image. Assuming that NCP space gives a reasonable correlation between the images and that the initial shift operation has succeeded in good alignment of the scan-lines, then a natural set of perturbations would be to shift pixels to the left, right, and center (none). An analysis of the computation is now performed to determine the tractableness of this approach.

Given the standard size of video images being 640 x 480 square textures of the size 512 x 512 can be created if the sides are cropped and the top is padded. The approximation of the NCP is used determine the region of overlap. If the cameras were separated by 45 degrees, or ¼ of the semi-circle, the NCP assumption states that the images should be shifted linearly by ¼ leaving ¾ * 512 = 384 pixels overlapping. Using the simple three-shift strategy of left, right, and center, this would give an expression of the form 3 x 3 x . . . x 3 = 3^{384}. This is more than a googol and hence is not worth pursuing.

The observation is also made that shifting pixels that the resolution left and right by one pixel at that resolution would probably not be enough to distort the image into a match. One approach would be to work with a smaller number of samples. This would both lower the computational demands and provide for larger shifts that would allow for a broader range of distortions.

Working with a shrunk scan-line of 32 pixels and assuming a ¼ offset, then there are 24 pixels overlapping. Perturbing each by three values would result in 3 x 3 x . . . x 3 = 3^{24} or about 282 billion variations for each scan-line. While this many matches may be realizable in an off-line process with today's computers, it is still not feasible for real-time applications. Working with scan-lines of 8 pixels gives us 6 pixels of overlap or 3^6 = 729 variations per scan-line. Given the expense of rendering a scan-line and computing the SSD, this is a much more reasonable number to work with. However, a scan-line of 6 pixels does not give us enough feature detail to perform to perform accurate comparisons.

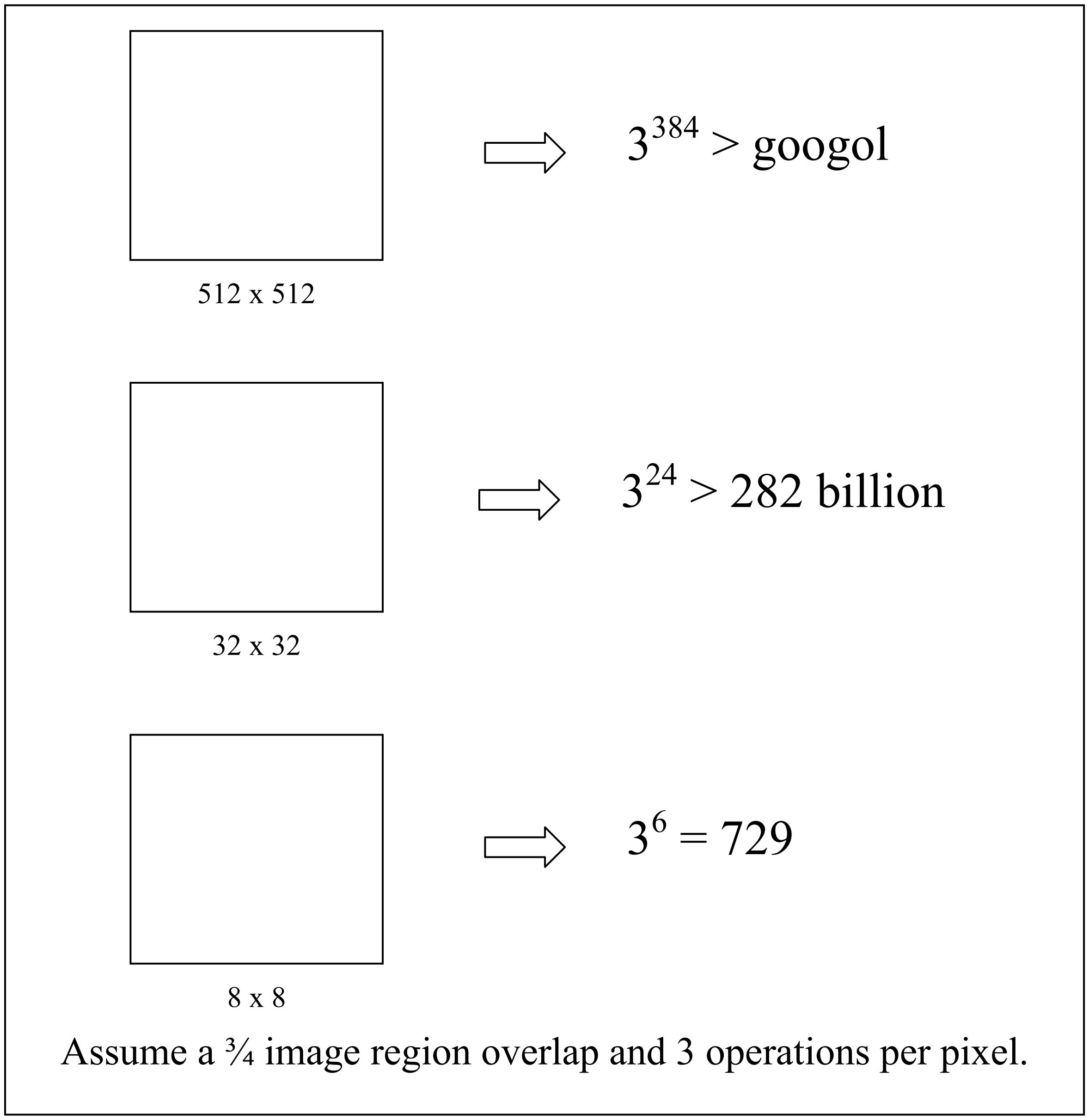


Figure 19 Assuming 3 match operations per pixel, what is the total number of operations per scan-line?

The solution to this dilemma is to move away from perturbing pixels to perturbing control points. The variation or frequency content of the shape of a face on a scan-line basis is relatively low. This frequency content is made lower by the NCP transformation. There is a general roundedness to a cross-section of a head, the nose produces a protrusion, and the eye sockets produce minor indentations. Hence, about a half a dozen control points are chosen to represent the shape of the head. Those control points are used to render scan-lines at a high enough resolution so that enough feature content is exposed so that effective SSD calculations can be made.

These control points are simply texture coordinates on the NCP space scan-lines. To generate a scan-line of 24 pixels, first the 6 control points are perturbed and then they are used as the texture coordinates for rendering the line.

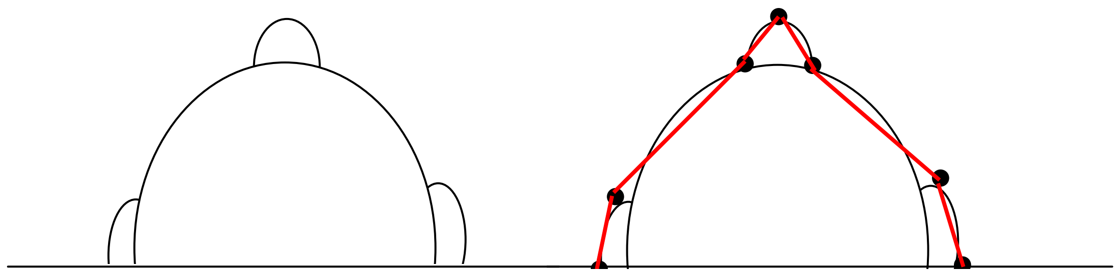


Figure 20 Diagram demonstrating the ability to represent the geometry of the head with a small number of control points.

An interesting observation about the combinatorial complexity of various matching configurations is made. The growth in the computational complexity is completely different when changing the vertical resolution versus changing the horizontal resolution. As it has just been discussed, small changes in the horizontal resolution creates large changes in the complexity due to the exponential nature of the growth. Changing the horizontal resolution from 8 to 16 makes the matching about 1000 times slower. On the other hand, changing the vertical resolution from 8 to 16 only increases the computation by 2. The growth in complexity by changing the vertical resolution is linear in behavior. The difference in this growth guides one in the choice of resolution to use for matching. One can take advantage of these differences to increase the resolution that matching is performed. The lowest level used for matching has twice the number of rows as it does columns. In this way one can have a more accurate base image to perform the matching while still keeping the performance of the program in bounds.

3.3.1 Course-to-Fine Correlation Matching

After the initial shift phase, the starting point for performing matching is still a rough approximation. An enhancement to the algorithm, investigated but not fully implemented in the current version of the software, is to have a multi-pass approach to refining the image correlation. The initial choice of perturbations of the matching phase must be large enough to span the range of possible adjustment. Using 6 equally spaced control points in NCP space, experiments indicate that using adjustments of +/- 1/3 are sufficient to align the image correctly. Having found an initial course distortion of the source scan-line to match the target scan-line, the algorithm proceeds by a series of successively smaller refinements. For example, the second round would use perturbations of +/- 1/6, then +/- 1/12, and so on until the benefit of the adjustment no longer justifies the computational cost.

Refinements can proceed separately in each dimension. After adjustments are made at one level, the resolution is vertically increased by doubling the number of scan-lines and the matching is performed again. Next the horizontal resolution is doubled while at the same time halving the fractional size of the control point perturbations. This process is repeated until the resolution of the final mesh used to render the view morph is reached.

3.4 XMIP – Multi-Resolution

A well-known approach to the speed up of the search for features in an image is that of data reduction or a multi-resolution analysis. For features that are large or have a low frequency characteristic to them, this technique works well. One can make use of this approach to first generally find where major features correspond and then successively refine their search at higher resolutions.

A conventional approach to creating the multi-resolution images is to make successively smaller images that are each a power of two smaller. This produces a series of images that are similar to the MIP-map decomposition of computer graphics, see Figure 21. When rendering images, MIP-map's work well when the features being rendered are scaled down about the same amount in both the X and Y dimension. When this is the case, the MIP-map that is closest in size to the destination image is chosen as the source for the texels for rendering the final image.

MIP-maps have a weakness when the destination image is significantly more skewed in one of the X and Y axes than the other. Consider the case where a texture map of the cover of a book has been generated as a MIP-map. If in the scene one places a book on end with the cover facing the viewer and the book is moved farther and closer, it is easy to choose the closest MIP-map to use as the source of texels for rendering. Now rotate the book along the vertical axis. Which MIP-map is best? Choosing the texture closest in the Y dimension will cause aliasing along the X dimension. Choosing the texture that best matches the X dimension will cause blurring along the Y dimension.

To solve this problem a previously discovered data structure that is known as a RIP-map or a "rectangular" MIP-map was independently developed (Heckbert, 1986;

Larson and Shah, 1993). Throughout the remainder of dissertation this data structure will be referred to as an Extended MIP-map or a XMIP. The XMIP is a full set of reduced images scaled along both the X and Y dimensions, see Figure 21. This data structure performs well in the cases where rotations are purely along the X and Y axes. (To handle the cases along other axes, a more advanced solution is needed such as anisotropic texture mapping is needed (McCormack et al., 1999)). Fortunately, after the transformation to make the search space along a scan-line, the stretches and scaling is purely along the X-axis. The XMIP data structure handles both the general image scaling done to reduce the image search space and the stretching along the X-axis done during the correlation search.

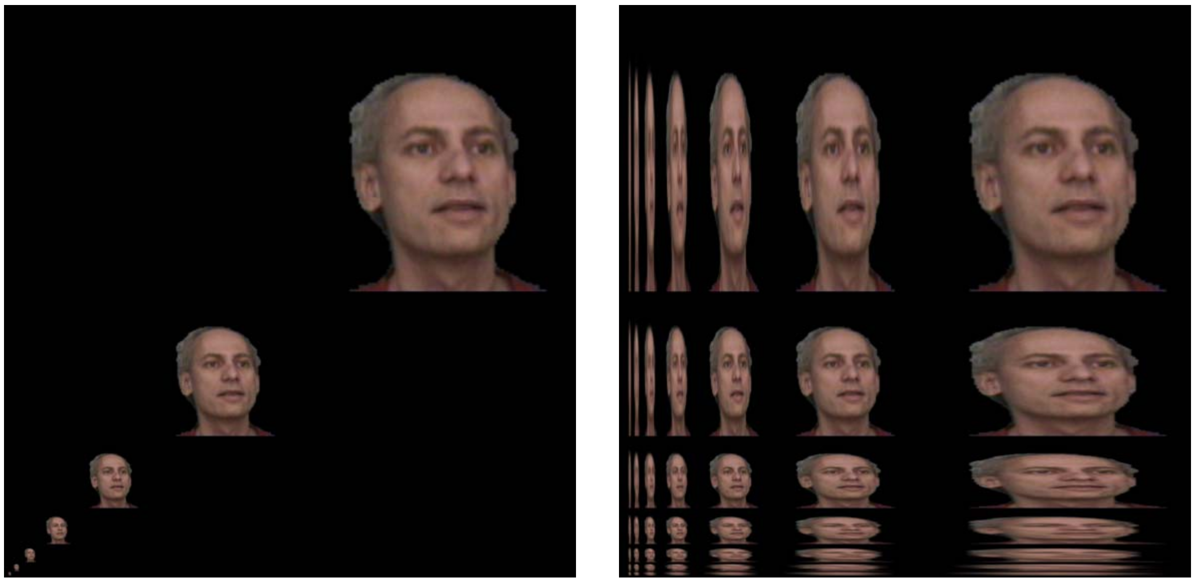


Figure 21 Conventional MIP-map images versus extended MIP-maps (XMIP). The XMIP data structure provides better control of aliasing and blurring.

3.5 XMIP-NCP

Combining the two data structures gives us a powerful method for efficiently finding correspondences between the images.

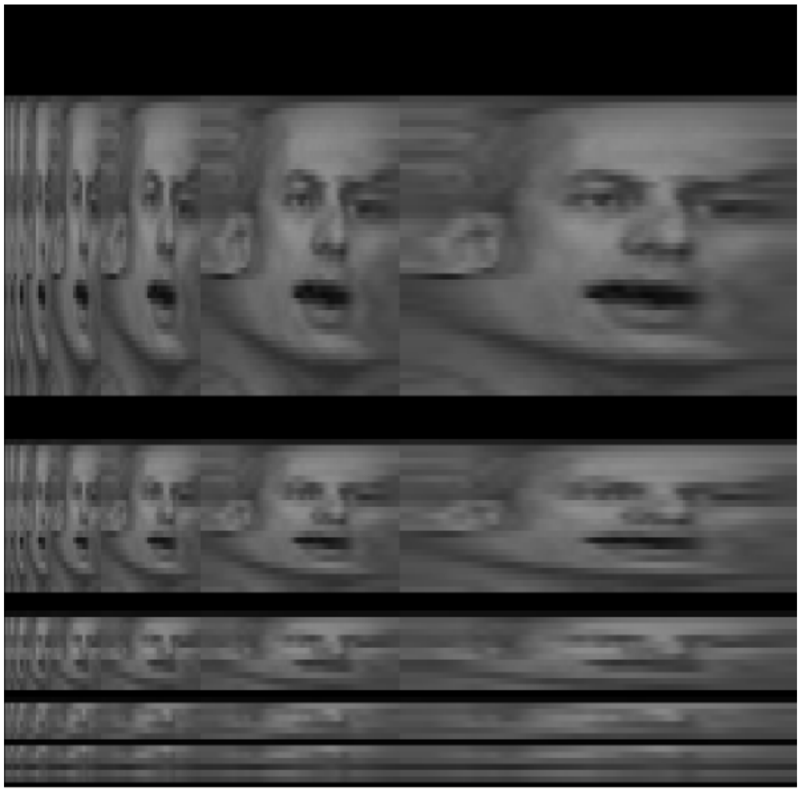


Figure 22 Example of combining the NCP and XMIP data structures.

3.5.1 Advantages

When combined, the XMIP and NCP complement each other to create a more effective data structure. The NCP operation causes blurring in the left and right regions of the image when they are stretched. When viewing a lower resolution NCP in the XMIP along the diagonal, the left and right regions are stretched relative to the reduced image but are derived from the higher resolution original image. These two factors have a compensating effect that provides for a more constant sampling of these reduced NCP images. In summary, the advantages of the NCP-XMIP include:

- Full rectangular area, all image points, used in finding correspondence
- Without XMIP, creating the NCP causes blurring of the regions on the left and right sides of the rectangular area. By combining the XMIP with NCP, the image detail of the left and right regions of the lower level NCP regions is improved.
- The XMIP-NPC is efficient in both space and time both in its creation and also in its use for accurately rendering of scan-lines.

3.5.2 Data Structure Generation

Given the real-time nature of this application, it critical to be able to construct the XMIP-NCP with minimal computation. Fortunately, this data structure can be constructed with the time complexity of the number of texels it contains.

The first step of creating the XMIP-NCP is to create the NCP at the top level. This is done with the help of lookup tables of texture coordinates that are generated when the application initializes. Consider an **n** x **n** matrix. Scan-line diameters can range from 1 to **n**. The NCP mapping is to first project the scan-line onto a semi-circle and then scale it so that its length is **n**. A table of **n** sets of texture coordinates is sufficient to generate the NCP mapping. Rendering the texture coordinates using linear interpolation generates the transformed scan-lines. Because the linear interpolation runs at ***O(N)*** complexity, the first stage runs at ***O(N)*** complexity for the number of output samples it generates.

The procedure is to first make the scaled columns of the half height NCP and then make the scaled rows of the full width NCP. Let **m** be the total number of texels in the **n** x **n** image. To create the first column that is half as wide as the full original data, some constant times m operations is performed. The second generated column only references pixel from the first reduced column. Because it only make half the references and produces half the data as the first reduced column, it time complexity is half. Algebraically this is represented as:

$$
\begin{aligned}
&\mathbf{Cm + Cm/2 + Cm/2^2 + Cm/2^3 + \ldots} \\
&\quad \mathbf{= Cm\ (1 + \tfrac{1}{2} + \tfrac{1}{4} + \ldots)} \\
&\quad \mathbf{< 2\ C\ m} \\
&\quad \Rightarrow \mathbf{O(m)}
\end{aligned}
$$

The number of operations to create the XMIP scaled rows from the partial XMIP containing just the scaled columns is four times that needed to create the scaled columns. Adding these together keeps the time complexity at **O(m)**.

3.5.3 <u>Fast Scanline Rendering</u>

The image correlation process consists of testing the best match from a set of control point variations. The test for a particular control point variation consists of rendering the scan-line specified by the control points and then performing a SSD to measure the closeness of the match. Given the large number of correlation tests to be made, having a fast scan-line renderer is a key component to speeding up the overall matching. The rendering of the scan-line is effectively a texture-mapping operation. To get any kind of reasonable quality, a linear interpolation rendering must be used. Using the nearest pixel would cause too many artifacts to be useful for the pattern matching. Texture mapping is a very computationally-intensive task. This is the reason that texture mapping was one of the first operations that early generation graphic cards used to offload the CPU. Therefore, a fast algorithm has been developed to render the variations of the scan-lines.

3.5.3.1 Pre-Rendering Pixels

After analyzing the structure of the texture coordinates generated, the following approach was taken. Consider the case where one has 32 samples and they want to make adjustments of +/- 1/3. By rendering a scan-line at three times the resolution, that is, 96 samples, one could create the set of rendered scan-lines by adding adjacent samples. For the unperturbed case, adding multiples of three sub-samples will create the rendered line. The groups of sub-samples to be added range in size of one to five. If the left sample is shifted to the right by a third and the right sample is shifted to the left by a third, then the distance between them is one third. Likewise, if the left sample is shifted to the left by a third and the right sample is shifted to the right by a third, the distance between the samples is five thirds.

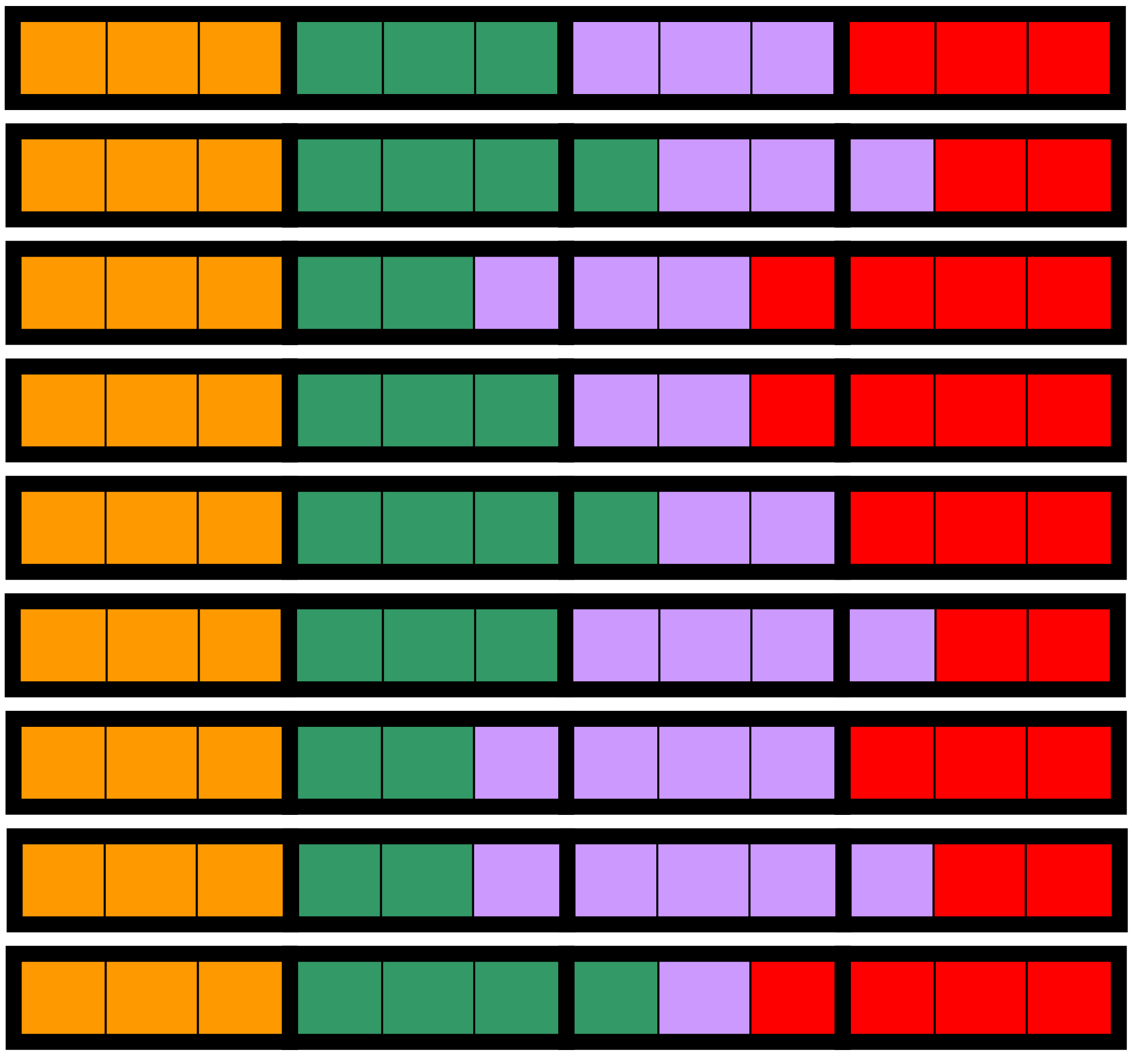


Figure 23 All combinations of the lavender pixel when restricting variations to thirds.

In Figure 23, the pixels have been divided into thirds. The first line is the original scan-line unmodified. This figure shows all the variations of the lavender pixel (second from the right). Lines 2 and 3 show the pixel retaining its original size but shifted to the left and right by a third. The fourth and fifth lines show the two possible combinations of the pixel when it has been shrunk to two-thirds of its original size. Lines 6 and 7 show the two possibilities when the pixel has been stretched by a third. Line 8 shows the single possibility of a five-thirds wide pixel and line 9 shows the single value of the one-third wide pixel.

It is possible to pre-render these pixel variations and place them into an array. From Figure 23, it can be seen that the range of pixel widths varies from 1/3 to 5/3s. If one were to render the set of all 1/3, 2/3, 3/3, 4/3 and 5/3 wide pixels, then they would have the full set of possible shifted and scaled pixel combinations with respect to the test model.

To create this set of possible pixel widths, first the original scan-line is rendered three times as long. Then the set of all one-third pixel widths is created by simply copying the pixels from the 3X scan-line. To create the set of two-third pixels widths, the first two pixels of the 3X scan-line are added and divided by two. The algorithm proceeds by shifting by one pixel in the 3X scan-line and then adding the second and third pixels and dividing by two. This process continues until the end of the scan-line is reached. Continuing, all the three pixel combinations from the 3X scan-line are computed and then each sum is divided by three. The four and five third combination sums are computed in the same manner. Let a[x] represent the pixels of the 3X scan-line.

One-third: a[1], a[2], a[3], a[4], . . . a[n]
Two-third: (a[1]+a[2])/2, (a[2]+a[3])/2, (a[3]+a[4])/2, . . .

. . .

. . .

. . .

Five-third sums: (a[1]+a[2]+a[3]+a[4]+a[5])/5, (a[2]+a[3]+a[4]+a[5]+a[6])/5, . . .

A data structure that can efficiently reference these pre-calculated pixel values is created. Let the dimension of the 3X scan-line be **N**, (i.e. **N** is three times the length of the original scan-line). A one-dimensional array organized as follows is created. The first **N** elements are dedicated to the one-third sums, the second **N** number of elements are dedicated to the two-thirds sums, and so one. Note, actually there are **N** one-third elements, **N**-1 two-third elements, **N**-2 three-third elements and so on. The entries are padded so that each set of X-third elements starts on a multiple of **N** boundary. The formula for indexing into this pre-calculated pixel array is as follows. The **x**th entry of the **y**-third element is **(N * (y-1)) + x.** The process of rendering a scan-line now becomes simply a matter of indexing into the pre-rendered pixel array.

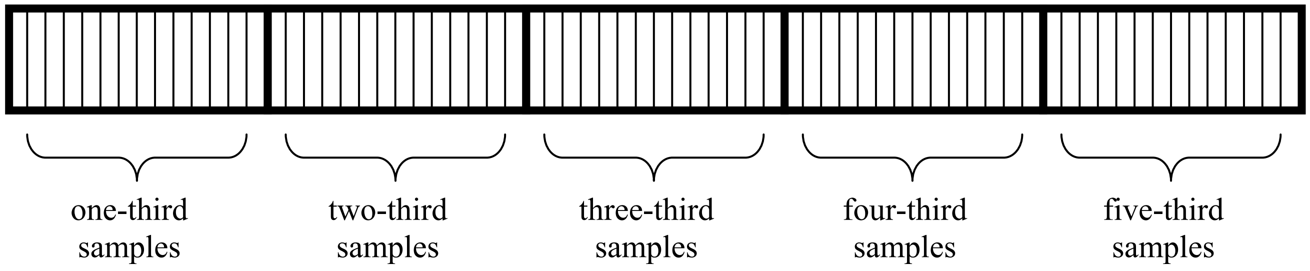


Figure 24 Array for holding pre-rendered scan-line values.

3.5.3.2 *Generating Sets of Scan-Line Variations*

Note, given a scan-line of fixed length and the restriction that the scan-line variations described above be limited to adjustments by thirds, the set of indexes that describe the scan-line variations do not depend on the content of the scan-line. This means that the generation of the indices can be computed at initialization time. When the program starts, a table is created where each row is a different scan-line set of perturbations and the columns are the indices into the table of sub-sampled sums that are created dynamically when the program is running. Rendering a texture-mapped scan-line then becomes just a loop of indirect references into the array of pre-computed sub-sampled sums.

What is the set of indices to generate the scan-line variations? It has been noted earlier that not all combinations of all pixel sizes are relevant. For example, if pixel A shifts to the right, then pixel B on its right must either shift to the right also or shrink in size. Refer to Figure 23 to help visualized these limitations. The set of constraints is as follows.

1. The entire scan-line must be covered.
2. No two pixel sums are allowed to overlap.
3. The sum of all pixel fractions must equal the original length of the scan-line.

3.5.3.3 <u>Optimizing the Scan-Line Variation Generation Process</u>

With the table lookup used for the scan-line rendering, the scan-line generation process is very efficient. However, the scan-line search algorithm entails generating tens of thousands of scan-line variations for every image comparison. In addition to the cost of generating the scan-line variation, the SSD of the variation with the target must be calculated also. In fact, this problem space search to find the best distortion of the scan-line, depending on the resolution chosen to work at, can consume all the computational power thrown at it. In view of the system overall, the computational resource usage of everything else is of little concern. This motivates one to pay special concern to this computational loop. Certainly it would be a good candidate for explicit code optimization, assembly language, or the use of SIMD (Single Instruction Multiple Data) operations. However, before taking on these kinds of optimizations, it would be prudent to perform some analysis of the algorithm.

The problem space is traversed by systematically generating variations of scan-lines. It is noted, however, that the difference between any two scan-lines in the search space is likely to be much smaller that the length of the scan-line. This similarity between adjacent scan-lines can be easily seen in Figure 26. Here it is seen that the pixels on the left not changing from scan-line to scan-line. If these pixels are not changing, then first they do not need to be rendered and second, the SSD values with the target image do not need to be calculated. The question is then, how much computation can be saved by only rendering the parts of the scan-lines that change. An analysis of algorithm is performed on the naive approach and then on the minimal rendering approach.

The time order of complexity of naive approach can be calculated as follows. First, for every pixel, there are three states: unshifted, shifted to the right, and shifted to the left. Given **N** pixels to be rendered, then there are 3^**N** possible lines. Because each line contains **N** pixels, the total time order of complexity is ***O(N 3^N)***.

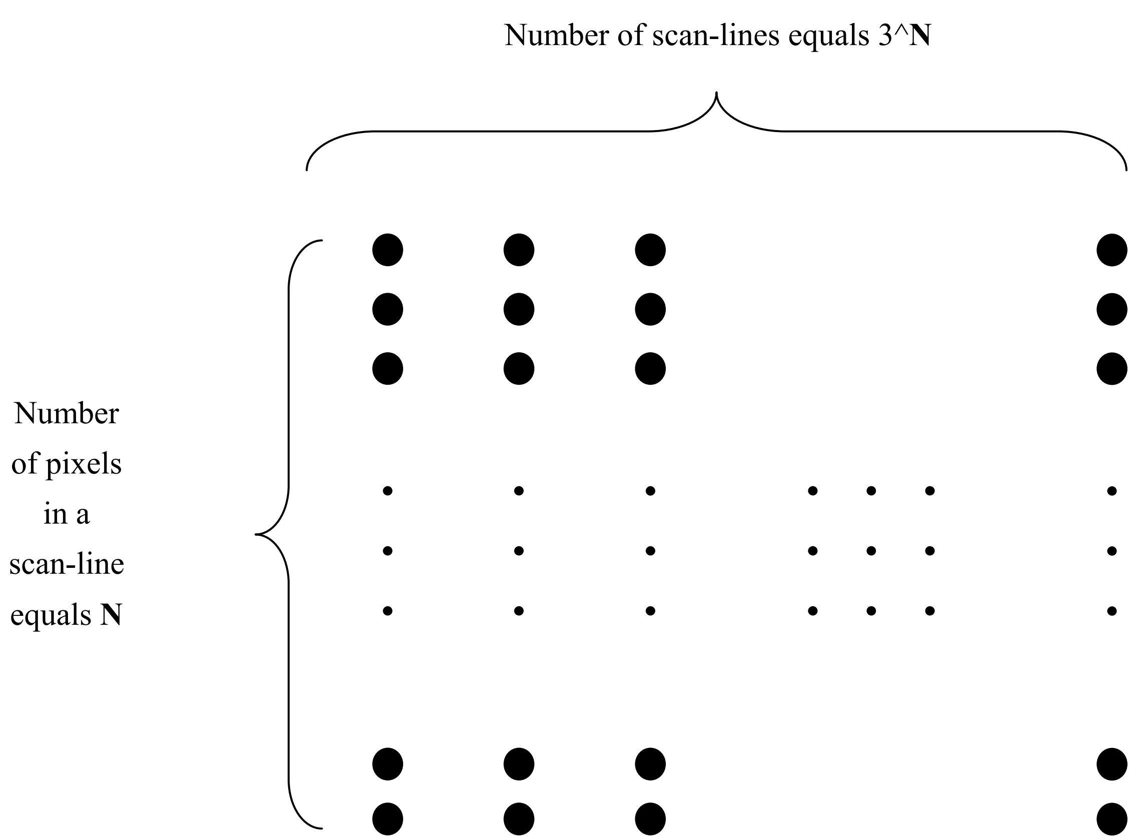

Number of scan-lines equals 3^N
Number of pixels in a scan-line equals N

Figure 25 The total number of operations in the naive problem space generation.

First, before calculating the time order of complexity of the render only what changes approach, let's consider what the theoretical lower limit of any algorithm would be. It can easily be seen, that because there are 3^**N** number of scan-line variations, there must be at least 3^**N** operations. By inspection, it is clear that no algorithm can run faster than 3^**N.** However, it is not known, without further analysis, that any algorithm can achieve that efficiency. For example, when sorting **N** items, one can infer that no algorithm can have a time order complexity of less than ***O(N)*** but that does not guarantee that one can reach that efficiency. In fact, in the case of sorting, it is mathematically provable that the smallest time complexity of any algorithm would be ***O(N log(N))***.

One approach to representing the minimal sub-scan-line re-render is by a tree, see Figure 27. Each path from the root to a leaf node represents a unique path in the search space. One can easily design an algorithm that visits each of the leaf nodes but only visits each node in the tree once. This could be done by saving the partial sums of the SSD calculations as each node is visited. When backtracking up the tree and then moving down again, there would be not any need to re-calculate the SSD starting from the root. The algorithm just uses the saved partial sum previously calculated. Thus, the time order complexity of the total number of operations is on the order of the number of nodes in the tree.

The question is then, how many nodes are there in the tree? The observation is made that in a fully populated regular tree, of any type e.g. binary, tertiary, decimal, **. . .** , there are more leaf nodes in the tree than all the other nodes put together. Specifically, in a tertiary tree there are 3^0 + 3^1 + 3^2 + **. . .** + 3^N number of nodes. It is seen that this expression is a sum of polynomials. Time order complexity theory tells that when one has an expression that is the sum of polynomials, all the lower order terms can be thrown away. This means that the time order complexity of visiting each node only once algorithm is 3^**N**. Indeed, from earlier observations, this is the most efficient algorithm that could ever be designed.

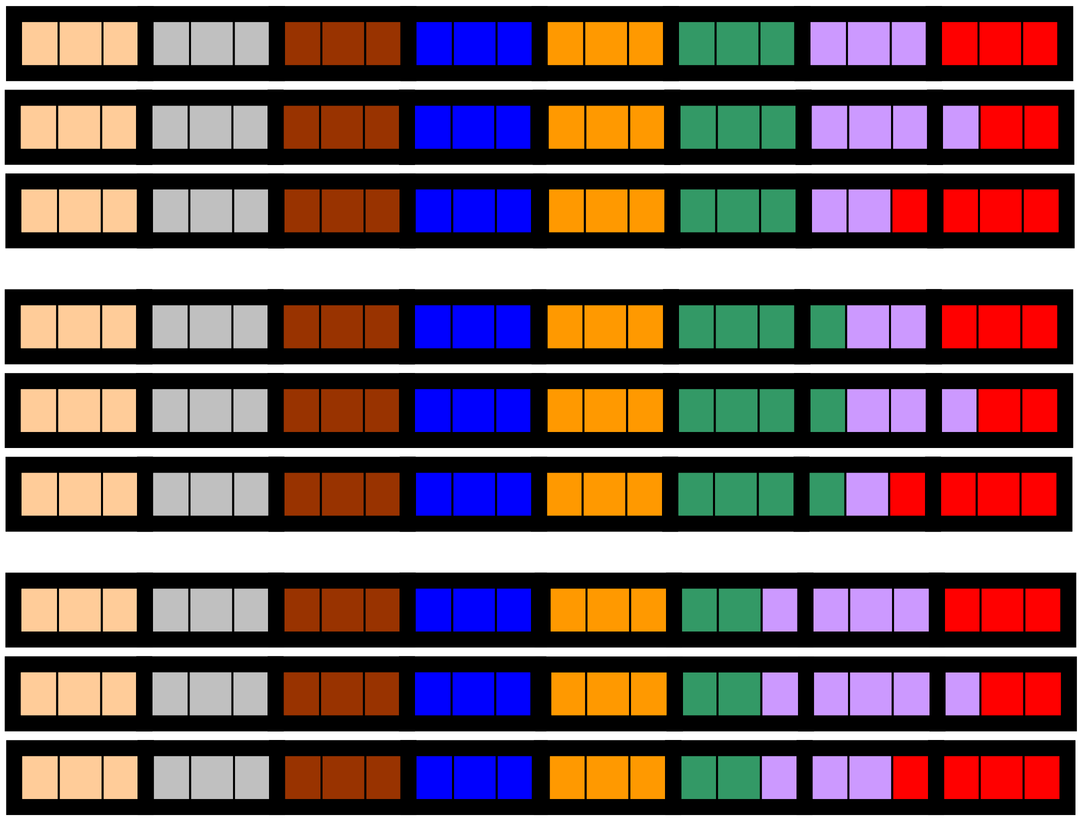


Figure 26 Diagram to expose the similarity between adjacent scan-lines in the search space. Left portion of scan –lines remain unchanged while the right side varies.

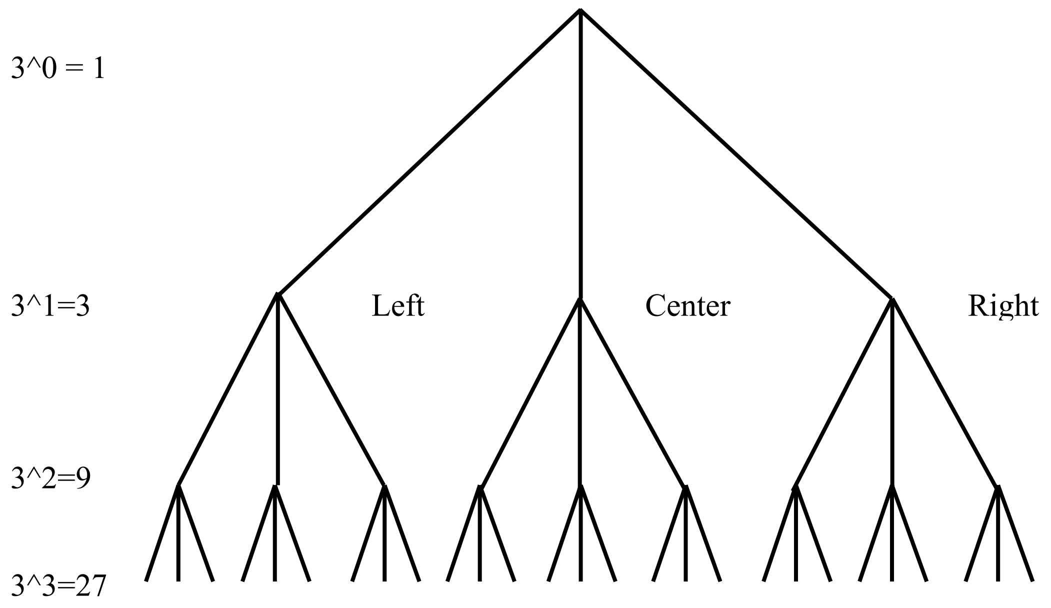


Figure 27 Tertiary tree representing the scan-line generation process.

3.5.4 NCP to Rectified Space Mapping

Having completed the matching in NCP space, the coordinates of corresponding points must be mapped out of NCP space back into rectified space. Because the NCP mapping is just a scaled projection onto a semi-circle on a per scan-line basis, there exists a straightforward procedure to perform the inverse map. One can compute the coordinate positions in rectified space by unnormalizing the scan-line to scale it back to its original length, taking the inverse cosine and adding an offset.

Because the number of coordinates to compute is in the thousands for each matched frame and there are tens of frames per second, this step was optimized for performance. Instead of computing the inverse cosine, a table for lookup of 256 values was created. To perform the lookup, first the closest entry in the table was chosen and then the fraction interpolated to improve resolution.

3.6 Synthesis

Having chosen the texture coordinates for the best match, this correlation is used to create the final rendered image. Image generation consists of the following steps:

- Geometric interpolation in *rectified* space
- Projection, by texture mapping, using the warped geometric vertices to the destination image plane
- Blend of images from both cameras

3.6.1 Geometric Interpolation in Rectified Space

Before the geometric location of the image mesh vertices can be interpolated, they must be transformed to a rectified image plane. The location of these rectified vertices are retained from the matching step. Because of the constraints of rectified image space chosen for matching, interpolation of vertices is a straightforward and consists of linearly interpolating the matching X coordinates of the two images.

3.6.2 Projection to Destination Space

Having generated the interpolated coordinates in rectified space, the location of these coordinates in the final destination image plane must be determined. In practice this can be done by either mapping from rectified space back to the original image plane and then to the destination image plane, or directly from rectified spaced to destination space. Regardless, the texture coordinates associated with vertices are those from the original match and are specified in the original image plane of the texture. For details on the calculation of the destination image plane transformation, see section 4.4.2.

3.6.3 Blend of images

The final rendered image is a blend of the two warped images. This blend is a simple linear alpha blend operation with the intensity level inversely proportional to the rotated angle. See section 5.1 for details of the alpha value calculation.

3.7 Summary of the Algorithm

3.7.1 Algorithm Overview

A general understanding of the image processing flow of the program has now been determined. The following sequence of operations is a list of the individual steps.

1. segment the region of interest (ROI)
2. warp the images to a parallel projection
3. perform the NCP
4. assign texture coordinates and vertices evenly dispersed in NCP space before performing the match
5. perform the inverse mapping of "default" vertices and texture coordinates to the original image, these texture coordinates will remain unchanged after the matching process
6. perform the match and record the destination location of the matched vertices (this is the location of the corresponding points of the "source" vertices)
7. given an angle, calculate the location of the view morphed vertices in the original image plane by performing a linear interpolation in parallel plane space
8. calculate the post warp transformation (done by linear interpolation of the quadrilateral control points)
9. using the original texture coordinates, take the new vertices and transform them by the post warp
10. alpha blend the images and display

3.7.2 Algorithmic Strategy

To meet the demands of real-time performance, a couple of design criteria were kept in mind while considering the overall program flow. First, given the large number of pixels in an image, it was decided that a multi-resolution approach was essential to making the time constraints feasible. Making this decision probably had the biggest effect on the overall performance.

An important goal of choosing algorithmic sub-tasks was to focus on operations that ran in linear time. That goal has been met in two important sub-tasks. The first is the image reduction operation that is the result of the XMIP generation process, see section 3.4. The other important area where linear performance was strived for was the generation of the scan-line variations, see section 3.5.3.3.

Another strategic guide was a derivation of the multi-resolution approach. At the lowest level of the multi-resolution analysis, there exist the least number of pixels to deal with. This gives an opportunity to expend additional computational energy on these pixels with only modest increase in the overall cost of the program. Extra matching is performed at the lowest level of the XMIP chosen to work with. As one works up to the larger levels, incremental improvements with less computational complexity are performed. Figure 28 shows the general overview of the algorithm.

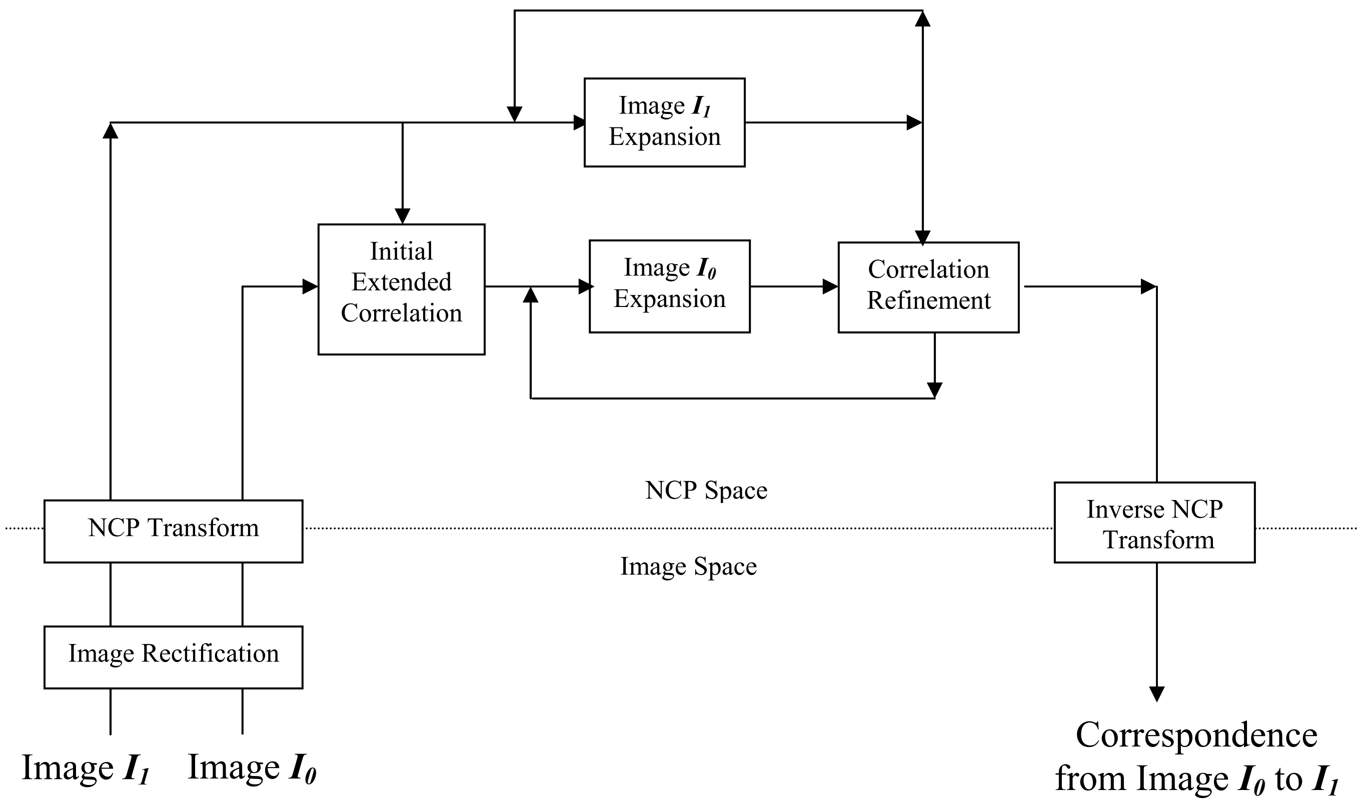


Figure 28 Diagram of an overview of the strategic algorithm.

4. IMPLEMENTATION

This section deals with the details of the system implementation. Whereas some of the previous sections emphasized the theory of operation, this section gets into some more practical aspects involved in creating the system.

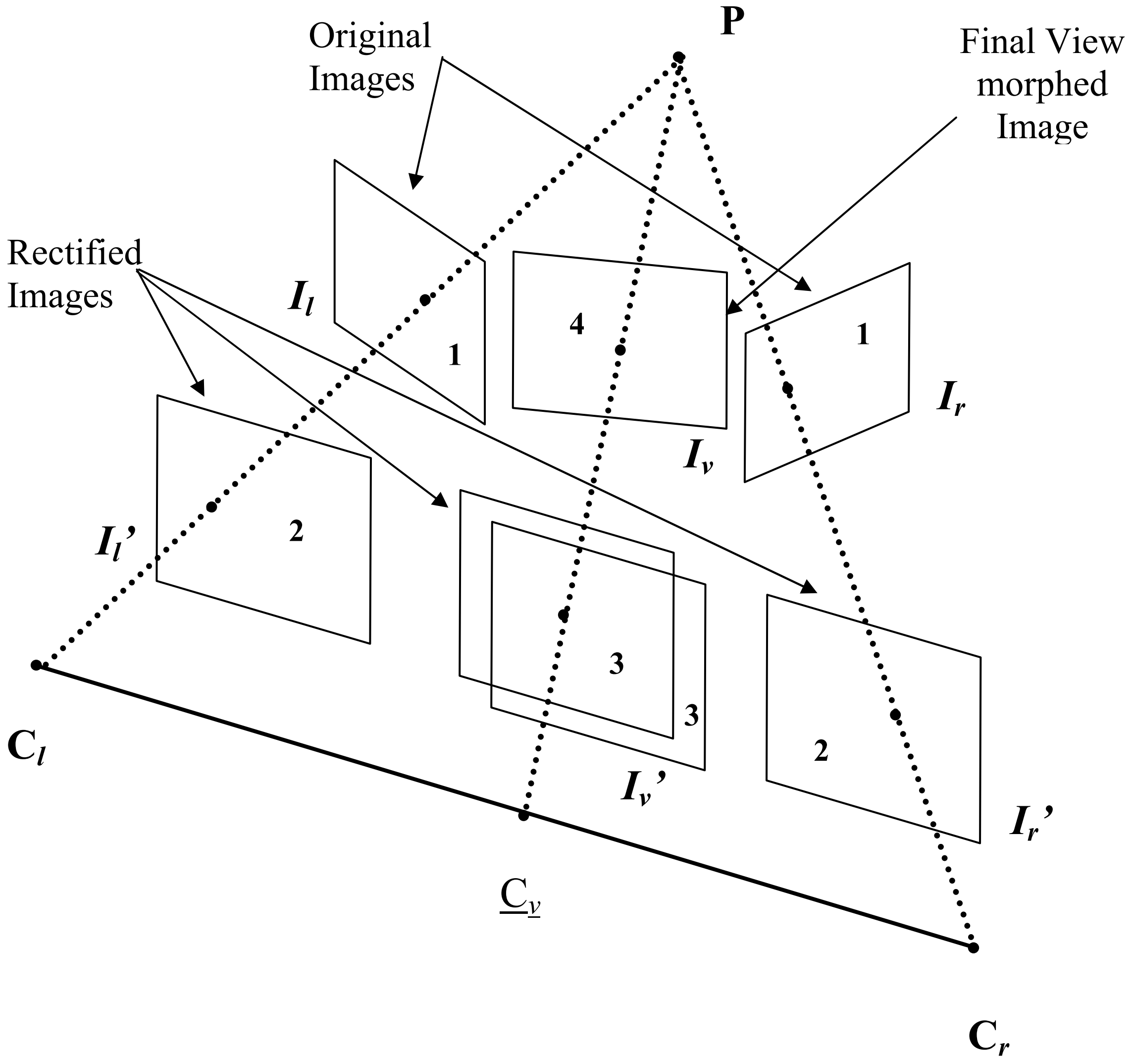


Figure 29 Diagram of the image transformation pipeline (see section 4.1).

4.1 Image Transformation Pipeline

In Figure 29 the overview of the image transformations involved in the general view morphing process can be seen. In this section, the details of how to implemented the transformations will be described and the issues that had to be dealt with discussed. Below is the terminology for the different image types.

1. **Video Images:** the original source images from the video cameras.
2. **Rectified Images:** the video images after they have be transformed so that they lie in the same plane.
3. **Interpolated Images:** the images created in rectified image space by warping the original rectified images.
4. **View Morphed Image:** the final end product image of the system.

The video images are the source images into the system. They are captured from conventional video cameras and handed to image-matching/view morphing system in real-time as fast as the system can process the images.

While pattern matching could be performed between the two source images, it could not be performed efficiently because the images lie in different planes as seen in Figure 29. Fortunately, there is a simple process for aligning the images after they have been captured. This involves performing a 2D perspective transformation.

4.2 Video Capture

One of the goals of this dissertation was to create a system that could be implemented on commodity hardware. There is a component of the video capture requirements of the system presented in this dissertation that is not part of the consumer market and might never become so. It is a key aspect of the technique so that capturing from multiple video sources is performed simultaneously. It is not clear that the average home video enthusiast would ever have a desire to capture video from multiple sources simultaneously and dynamically merge them. Therefore, the all commodity system is violated on this account.

As time went on, the author's understanding of the demands of a multiple video system increased. Having built an initial system, the quality limitations of it became frustrating. Therefore the work of this dissertation was built using two different video capture configurations. These two systems will be described below and their strengths and weaknesses discussed.

Ideally, the video capture would be live and the system would process it without needing to access the disk system. The system as it was actually implemented, consisted of capturing the video to disk as a separate process. This data was then processed off-line to create a proprietary video file format. When the view morphing system was run, it would read in the proprietary video file into RAM at initialization time and process the videos data by chroma-keying it. At run time the system would just read these chroma-keyed video frames out of RAM.

From a development point of view, there is a huge advantage to be able to work with pre-captured data. As a developer, one can run the program again and again on the pre-captured video and concentrate on the programming task rather than needing a subject to model continuously.

Another important advantage to having the system run from video files on disk is portability. While the bulk of this development was performed on a PC running Windows®, at home, the computer systems at school were SGI® systems running IRIX® version of UNIX®. Video capture software is something that does not possess a cross platform standard. In the Windows environment, Microsoft has developed a couple of standards. The legacy standard that is currently deprecated is VFW (Video for Windows). The current standard belongs to Microsoft's DirectX® family of components and is called DirectShow®. While using DirectShow components provides for inter-operability between various vender's hardware, it does so only on Window's operating system platforms. The 3D graphics operations for this dissertation were written in OpenGL®. This allowed for platform independence. The system was able to be developed on a Windows PC and to be ran on a SGI system by simply performing a recompilation of the code. The software also ran Sun Solaris® hardware by simply recompiling it.

In addition, reading the video data from disk not only provided for hardware and operating system independence, it provided for portability of the data. Even if all locations were running the same hardware and operating system, setting up the video cameras to capture the data is a non-trivial effort. The camera setup with it chroma-keying background takes up a lot of physical space. Taking a CDROM to a new location was much easier than setting up the camera capture system.

4.2.1 Video Capture System One

The choice of this initial system was influenced by the author being self-funded and trying to achieve minimal cost. An initial technical challenge was how to capture the video from multiple sources simultaneously. The support for multiple capture cards was limited at the time. In addition, because the ability of PCs to handle the data rates of a single video stream was an issue, capturing from multiple streams would have been even more unlikely. Compression of the video streams would mitigate the demands on the system for data transfer but does so at a loss of image quality. For the matching process to perform well, the absence of any artifacts in the images is of critical importance.

To deal with the issue of capturing the video from multiple cameras at the same time, a video multiplexer was obtained. Quad video multiplexers are popular in the security surveillance industry. It is common for a store or a business to have multiple video cameras to be placed at various locations. Having separate monitors for each video camera is not practical so quad video multiplexers are often used to take then input of up to four cameras and display them as a 2x2 grid. Having combined the video of multiple cameras, it can be captured into the computer using a single capture card. The quad multiplexer purchased took conventional NTSC composite signals as input and produced a composite signal as output. No audio was processed. The multiplexer costs about $750.

The initial cameras for the setup were conventional consumer camcorders. The advantage of using camcorders is that they provide a variety of features. Useful features include auto-exposure, auto-focus, and the ability to zoom the focal length. One disadvantage of consumer camcorders is that they are not normally designed to remain on for extended periods of time without the user recording to tape. This has the effect of the camera turning itself off at inconvenient times[2].

Two separate pairs of cameras were purchased for video camera system one. The first pair consisted of the family camcorder of one brand with a second unrelated brand.

[2] It was discover that this annoyance could be overcome with one brand by not placing videotape in the camera.

This produced images that possessed wildly different characteristics such as contrast, brightness, hue, and saturation. Fortunately, the video multiplexer had the ability to adjust these characteristics on a per camera basis. While proceeding manually by trial and error, much improvement was made in the making the images possess similar chroma characteristics, however, these improved images were still less than ideal. These disparate video cameras were later replaced with a pair of identical consumer hi-8 camcorders in the $200 price range. While these cameras produced similar images, the quality was less than acceptable. The inexpensive price certainly was one factor in the low image quality. A lot of chroma bleeding distortion was seen in the images. In retrospect, after dealing with camera setup two described below, it is now believed that some of the low quality was due to insufficient lighting. While the camera continues to produce images in low lighting situations, the quality degrades considerably. Most notably the images contain a lot of noise and the color is greatly distorted.

The issue of lighting is one of the most important. Of fundamental importance is the need for sufficient lighting as noted in the previous paragraph. A second issue is that of the uniformity of the lighting. The creation of shadows can affect the effectiveness of the performance of the chroma-keying system. Depending on the settings of the chroma-keyer, dark shadows might fall out of range. Another lighting concern is that of specular highlights. The nature of specular highlights is that their location on the subject is dependent on the viewing angle. That means that the two cameras will see the specular highlights in different locations. Excessive specular highlights will confuse the pattern matching system. Using very diffuse lighting can mitigate specular highlights. The use of florescent lighting helps in this regard.

Throughout the duration of the project, several video capture cards were purchased. The first three were combination computer graphics and video capture cards. These combined cards were less expensive than buying the cards separately and seemed to offer efficiencies when processing the incoming video and displaying it on the screen. The disadvantage of combination cards is that, in recent years the rate of improvement in performance of graphics capabilities has been tremendous. This has encouraged the replacement of the graphics card in the system on a regular basis. Combination cards are more expensive than single function cards and are more temperamental in terms of drivers. This has lead to generally unstable systems as operating systems change. New versions of DirectX are delivered by Microsoft, and the hardware has changed. With the use of video overlays and DMA access directly to the graphics card's video memory, the efficiencies offered by combination cards are no longer an important factor. In video capture setup two described in the next section, a system where the capture card and the graphics card are separate is described.

The following combination graphics and video capture cards were used over the course of this project. The first was the Matrox Marvel 200®. This provided good quality video capture and performed hardware MJPEG compression. The graphics support, while competitive at the time, mainly provided texture-mapping support; the transformation of vertices was performed by the CPU. Next an Asus V6800 Deluxe® based on the NVidia® Gforce 256 chip. This card performed the vertex transformations on the card itself there by freeing the CPU to concentrate on the matching operations. The last card purchased was the All in Wonder – Radeon® 7500. This card possessed both advanced graphics capabilities along with hardware support for video compression. In terms of video quality, the Matrox® card had the best with the other two being disappointing.

Another consideration of the graphics card is the ability to have a TV-out connection so that the output of the program can be archived. Choices for the video out are either composite or S-video. The composite signal combines the Y, luminance, and the two U-V, chrominance, signals onto a single wire. This has the effect of some of the color information being lost in the process and colors of the composite signal are not as vivid or separated. The S-video signal separates the signal into two parts, the Y and the U-V. This provides for better color in the transmitted image. There is a third choice where three signals are output R-red, G-green, B-blue which is referred to as a "component out" system. This provides for the best signal but is only just becoming available in graphics cards to provide better DVD playback. Given the nature of equipment being dealt with and the lack of high fidelity inherent in the view morphing process, a composite signal is adequate for the system's needs.

In the current implementation, the chroma-keying of the background was not processed in real-time. Video from the two cameras was captured into the computer and then keyed in an offline process. High-end video capture and processing cards provide chroma-keying support in hardware. Alternatively, a state-of-the-art PC could probably chroma-key in with the main CPU(s) in real-time.

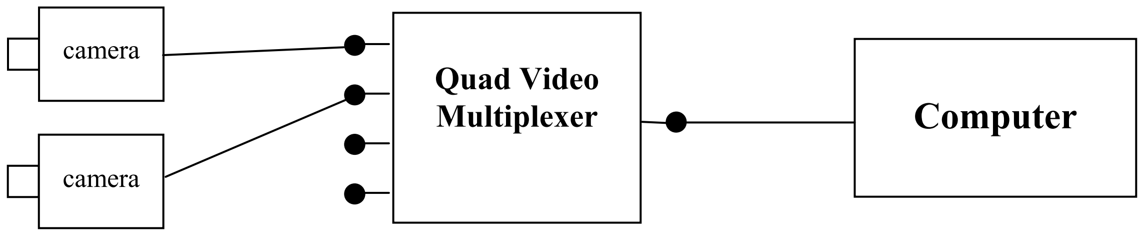


Figure 30 Diagram of Video Capture System One

This video capture setup has a variety of weaknesses as listed below.

- Cameras are not synchronized
- On a two-camera system, half the captured data is wasted
- The image processing performed by the multiplexer necessarily adds some degradation to image quality.
- The quality of cameras was not good.

The timings of the image capture of the camcorders were independent of each other. The original thoughts about the system were that that quad multiplexer provided synchronization that was needed. In early versions the video was captured only at the rate of 15 frames per second. It was thought, that given the coarseness of the capture rate, some fraction of a thirtieth of second would not matter. Upon visual inspection of the paired images, significant shift of the subject in the frame such as mouth and head movement was viewable. A system where the video image capture is synchronized is needed to remove these anomalies that will certainly cause interference with the pattern matching system. Conventional camcorders on the market do not provide for any mechanism for synchronizing the signals.

Another problem with this system is that the quad multiplexer generates output that is based on getting four inputs. To get a proper layout of the four input video streams, a 2x2 format is used. In a view morphing system that only uses two video inputs, half the image region is unused. That is, the video input stream to the system is twice what it needs to be. Because of the large amount video data generated, it is a strain on the system to process it, especially for the disk system to save.

Finally, the quad multiplexer is a source of image degradation. While a high-end system might provide excellent results, the more modestly-priced system purchased had a variety of artifacts. First the overall image was blurrier. Second, when the subject moved quickly to the left and right, a visible tear between the top and bottom halves was visible. Such a tear would cause great havoc with the image correlation system. Finally, with age, the system seemed to degrade in terms of the offset and in the borders between the images.

4.2.2 Video Capture System Two

To address the deficiencies of the video capture system described in the previous section, an improved system was created. First, the camcorder cameras of system one were replaced by higher quality cameras that provided only video. The money that was spent towards the taping mechanism, view finder, special effects, and other consumer features were concentrated only on the video capture system. A Sony® SSC-DC393 camera producing 480 lines of resolution was purchased for $215. A fast, auto-iris, zoom lens was purchased to go with it for another $150. These prices make the camera competitive in price with consumer camcorders.

In addition to the better image quality, these new cameras also possessed the capability of synchronization. This was achieved by the cameras synchronizing with the line voltage of their AC source.

The quality of the camera was only one issue in the video capture system. The second problem was with the quad multiplexer. To overcome this limitation, a single board PCI card that possessed four independent video capture ports on it was purchased. Each of the video ports could be individually accessed simultaneously. This new video capture card relieved two problems caused by the quad mult-iplexer. First, the new system had direct access to the high-resolution quality of the cameras directly with no resampling of signal. Second, all the video data processed by the system was relevant. This is in contrast to the half the image data in the quad multiplexer that was blank.

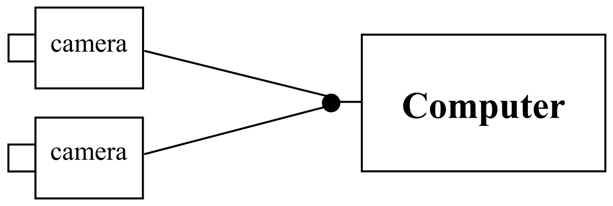


Figure 31 Diagram of Video Capture System Two

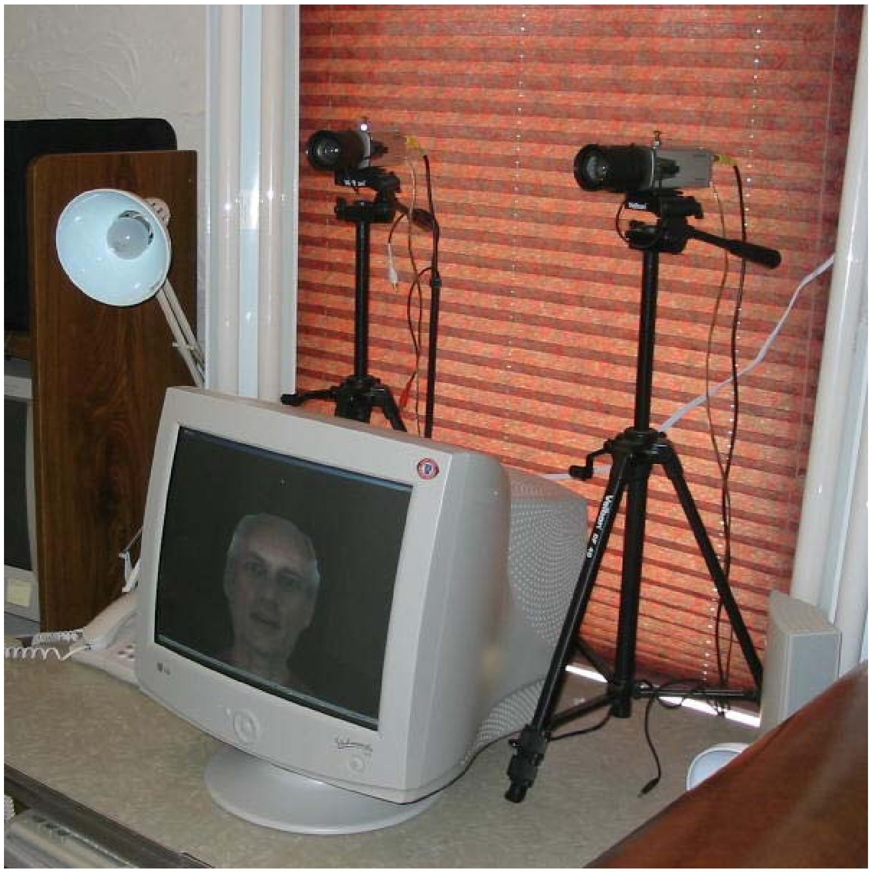


Figure 32 Photograph of the image capture system.

A technical complexity of this second system was how to capture the data from multiple video inputs. In system one, there was just one video signal that is easily captured by any conventional video capture software program. Conventional video capture systems are not designed to handle input from two cameras simultaneously let alone trying to synchronize them. There were two choices with the second capture system. One was to write original software that would process the data from the two systems at the time. The second choice was to run two conventional single video capture programs simultaneously. The image sequence would need to be aligned after the fact to handle the issue of capture not starting at the same time slice. After some preliminary investigation of the Microsoft's DirectX9 interfaces, the choice of writing a proprietary dual video input program was abandoned in favor just running two capture programs at the same time. While the DirectX interfaces make it easy to perform conventional operations, using these interfaces to perform unconventional operations appeared rather involved. While the single program to capture both videos streams would be more desirable, it was not pursued due to the lack of time at the stage of the dissertation when this new system was purchased.

4.3 Video File Format

A proprietary movie file format was developed to store the images. The beginning of the movie file consisted a header describing the file. Included were conventional movie file information such as image dimensions, color depth and format, number of frames, frame rate, and the like. In addition, unconventional file information needed for view morphing was kept also such as the number of images per frame, how those images were laid out (e.g. left to right or up and down), and information on how to chroma-key the images. The remainder of the file consisted of the movie frames, each of which was composed of two or more images synchronized in time from different viewpoints.

The video capture was performed on a Windows-based system. This would suggest the use of encoding the video information into the Windows AVI video file standard. While a dual video stream encoding might be possible with the AVI multi-stream format, an alternate simpler scheme is possible. One could encode the two images as a single stream by making the image dimension double tall. This would allow one to use the top half for one image and the bottom half for the other image. In the end, the AVI format was abandoned due to its dependency on the Windows operating system and the libraries it provides for referencing the files. An important goal of the project was that it be able to run on a variety of operating systems, the most important of which are UNIX and Windows.

Standard textures in OpenGL are square and powers of two. Typical video acquisition is a 4:3 format, e.g. 640x480 or 320x240. The video images are made into square power of twos by cropping the sides and extending the top region by duplicating the last line. For example, for a 320x240 image a texture size of 256 was created by cropping 32 pixels from both the left and right sides and adding16 lines to the top.

4.4 Camera Calibration

A general pre-requisite for image-based rendering systems is camera calibration. This project used a constrained setup for the cameras that simplified this process. In addition, view morphing does not require knowledge of 3D shape. (Seitz and Dyer, 1996). This allows us to perform a simplified camera calibration process while still achieving good image results.

4.4.1 Physical Setup

The setup consists of two (or more), cameras orientated towards a common point. Typically, the camera would be located a few feet from the object of interest although an configuration where the cameras were farther back with a longer focal length would be feasible and would have the advantage of providing images with a more orthogonal or weak perspective nature which would minimize errors in the camera calibration. A background of single solid color for chroma-keying was used in this implementation.

Alternatively a known static background to be digitally subtracted would be equally suitable to provide easy segmentation of the object of interest from the background.

The two cameras were aligned to be approximately oriented towards the same point in space with roughly the same focal length and at a predetermined angle. A quick and dirty method for aligning the cameras was performed by the following steps. First the two cameras were placed on tripods at the same height. Then a template was used to place the cameras at a known angle. Next a ball was placed at the center of interest of the two cameras. The cameras were rotated until the ball was in the center of the images of the two video cameras and they were zoomed until the ball was about the same size. Next a rectangle was inserted in the center of the scene and its image captured by the two cameras to be used for the camera calibration step.

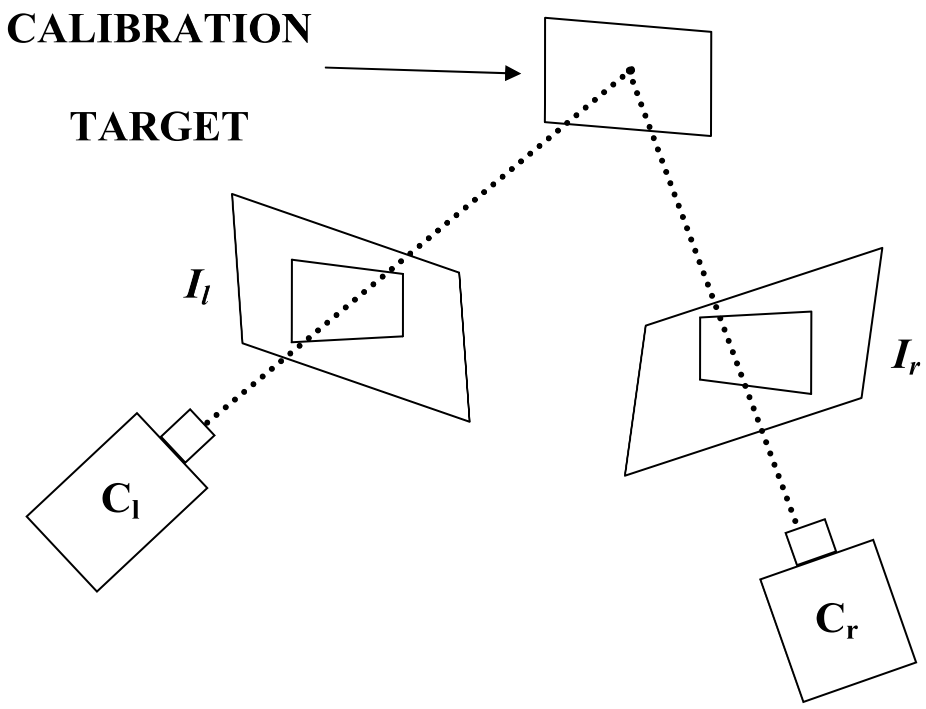


Figure 33 Camera calibration setup.

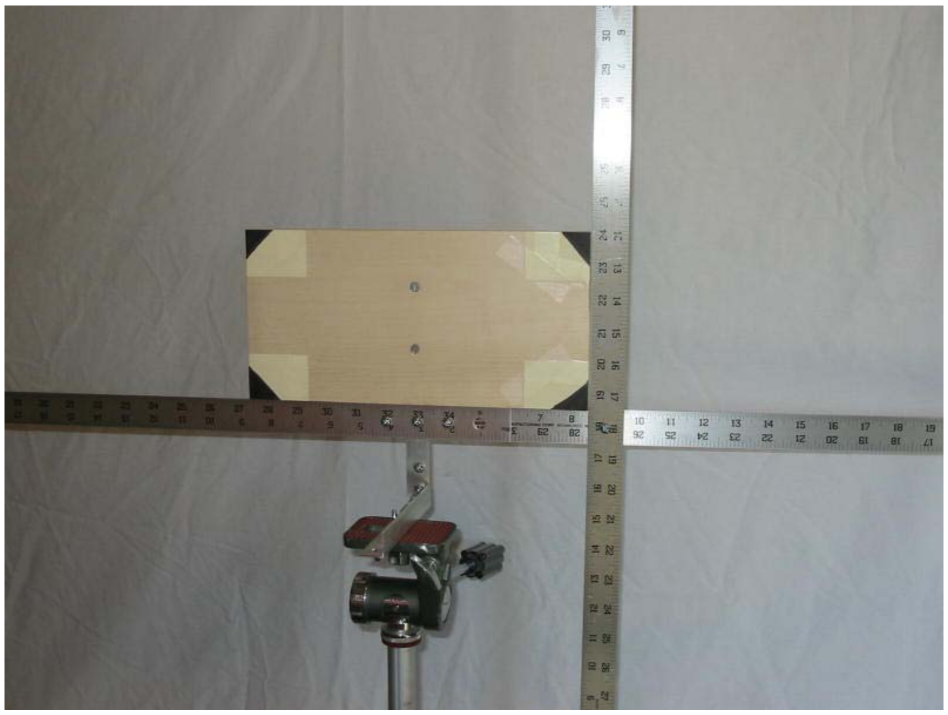
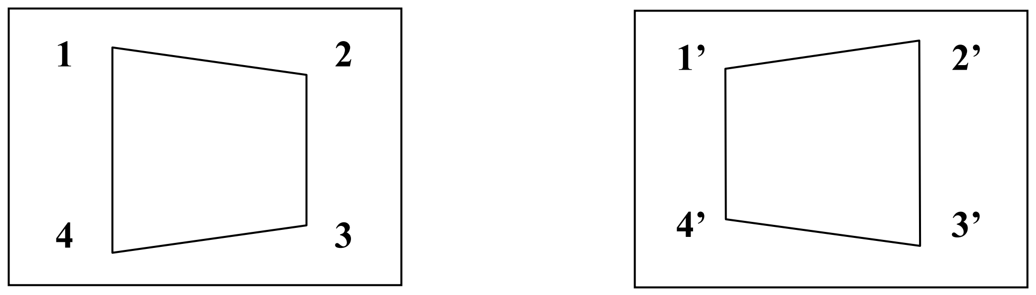


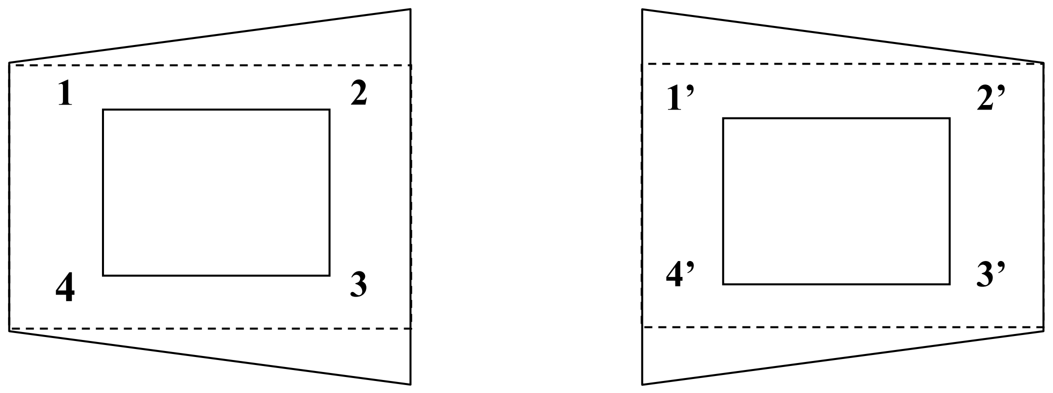
Figure 34 Photograph of the calibration target.

4.4.2 Rectification Matrix Calculation

In order to perform the pattern matching, the image planes of the both cameras need to be rotated so that they are parallel and that their scan-lines aligned. Knowing either the full set of camera model, both extrinsic and intrinsic camera parameters, or knowing 8 or more corresponding points in each image was suggested by Seitz, (Seitz and Dyer, 1996). This dissertation uses a simpler approach of selecting 4 corresponding points in each image. As mentioned above, to facilitate calibration, a rectangular object was placed in the scene. The corners of the rectangle were identified by clicking on them with a mouse. These four point correspondences in each image is enough to give use 8 equations that can be used to define a 3 x 3 matrix that will transform one image place to the other (Wolberg, 1990). In order to not favor one image plane over the other, neutral image plane was defined by calculating the midpoints of corresponding points and mapping the original image planes to this intermediate plane. Because of the highly specified orientation of the cameras, there was no need to worry about the case of where any three points of the rectangle could become collinear in any particular in between transition. See Figure 35 for a diagram of the alignment.



Original left and right camera images.



Images after correction distortion applied to scan-line corresponding points.

Figure 35 Diagram of the scan-line alignment distortion process.

4.4.3 Post Warp Matrix Calculation

After performing the match with parallel views, the image plane of the final view had to be determined. Because the full camera parameters were not known, a simplified method to generate the final image plane was used. All the processing was done in 2D. The same 4-point correspondences that were used to find the parallel matching plane were used again. The location control points were interpolated and then used to define a new 2D projection matrix. Applying this post-warp transform to the new views has the effect of performing a keystone operation (see Figure 8). It also smoothly handles differences in scaling, translation, and orientations of the cameras.

4.5 Segmentation

After the initial capture of the video, the next step is to segment the subject from the background. Once the region of interest is separated, it can be further processed. Various techniques exist for segmentation. One popular approach for segmenting faces is flesh tone matching (Pentland and Choudhury, 2000). This approach is useful if nothing is known about the background of the image. This approach has various drawbacks such as large flesh areas such as legs being falsely identified, it is racial skin tone dependent, and it does not segment the hair and clothing. In the case of this dissertation, the environment was controllable and the background could be manipulated. This allowed for either of two major approaches to be used: chroma-keying or digital subtraction.

4.5.1 Chroma-Keying

Chroma-keying or blue-screening, is the popular technique in the TV or movie industry of placing a mono-chromatic background behind the subject. Later, in post processing, this single color can be removed and replaced with a different background that can have the effect of placing the foreground in a different location. Another technique that accomplishes the same results is digital subtraction. First the background is captured without the subject and then the subject is placed in the scene. The newly captured image with the subject is compared to that of just the background and the foreground subject extracted.

Both chroma-keying and digital subtraction suffer from various problems. One problem common to both is that of holes. In the case of chroma-keying, if some part of the foreground has the same color as the background, it will be segmented out. Choosing a background color that does not naturally or typically appear in the subject can mitigate this weakness. Historically a common color used for the chroma-keying background was blue (hence the term). In practice these days green is more often used because it has a smaller component of flesh tones. Analogously, digital subtraction can produce holes if the color of pixel in the foreground is the same as that of the background.

Both schemes can also generate errors due to variation in intensity levels. One cause of intensity variation is the auto-exposure features of video cameras. Placing the subject in the scene can cause the total light entering the camera to change. As the subject moves around, the change in distance to the lights can cause exposure changes. Another problem is that of shadows. If the subject casts shadows on the background, the chroma of the background does not change (assuming consistent lighting sources) but the intensity level can change dramatically.

Dealing with variation in intensity levels can be handled by using an appropriate color space. Computer graphics programs typically work in the RGB color space. Here each of the color components deals with chroma. The range of intensities that are considered to be a match cannot be adjusted without changing the chroma of the color accepted. Alternative color spaces exist that factor out the intensity level of a color sample from its chroma components. Two popular choices in the video domain are YUV and YCrCb. In both these formats, the Y stands for a pure intensity component and the second two terms express the chroma values.

Arbitrarily, chroma-keying was chosen here to segment the foreground subject. When performing the chroma-keying, the application converts the image data from RGB that is used by OpenGL for texture maps to YCrCb space. Working within this color space, a rectangular box of acceptable color values for the background was created. Knowing that the background was monochromatic, the range of the two chroma terms was kept small while allowing for a much broader range of the intensity values.

4.5.1.1 Color Selection

Having identified the mechanism for segmenting the foreground, a tool that would allow for the easy identification for the background colors was created. The goal was to choose the dimensions of the chroma-key rectangular box to be large enough to include all background colors but no larger than necessary to inadvertently include any of the foreground pixels. Having used interfaces of other applications, which allowed users to explicitly choose the color components based on the numeric values in color space, it was determined that a more intuitive interface was needed. By just viewing on image, no one can accurately specify the range of numeric color values. Being presented with dials to control the numeric ranges, individuals struggle with much trial and error to adjust the settings to suitable values. Each chroma value has a center value and a range giving the user a total of six values to adjust to achieve the ideal key specification.

It was decided that a better way to determine the color values was to read them from the images themselves. An interface was created where the user could specify the color by sweeping rectangular regions using a mouse on the video images. All colors in the region swept are added to the range of colors of the background. By definition, no unnecessary colors are included in the keyed out range.

In a system that processes pre-recorded video only, one can select all background colors by cycling through the images and selecting regions until all the background is keyed out. On a system that processes live video, with no opportunity for a per subject chroma-key phase, the creation of the key range could proceed in the same manner with a sample subject. Then the range of the chroma-key could be expanded slightly to account for changes in the auto-exposure processing of the camera and other lighting interactions. Because the chroma of the background would be stable, most the most the expansion of the chroma-key region would be in the intensity range only.

4.5.2 Shape Smoothing

High-quality chroma-keying is difficult to do without expending a effort in the placement of consistent high-quality lighting in a well-dispersed manner. The amount of time and money available for a highly effective chroma-key environment was limited. To compensate for the low quality of the lighting environment and the simplicity of the keying approach, a post-keying step was added to the algorithm. Given knowledge about the foreground object that a general algorithm would not have, a shape smoothing operation was applied.

Working with a human subject viewed from the shoulder on up, it is assumed that the foreground shape has these characteristics.

1) On any horizontal scan-line, there exists at most a single foreground region.
2) The subject extends to the bottom of the image.
3) The top of image may or may not be empty.
4) The shape of the subject changes smoothly.

An implication of the above assumptions is that the outline of the foreground region defines a curve whose height is monotonically increasing starting from the left, reaches a peak, and then is monotonically decreasing (see Figure 36).

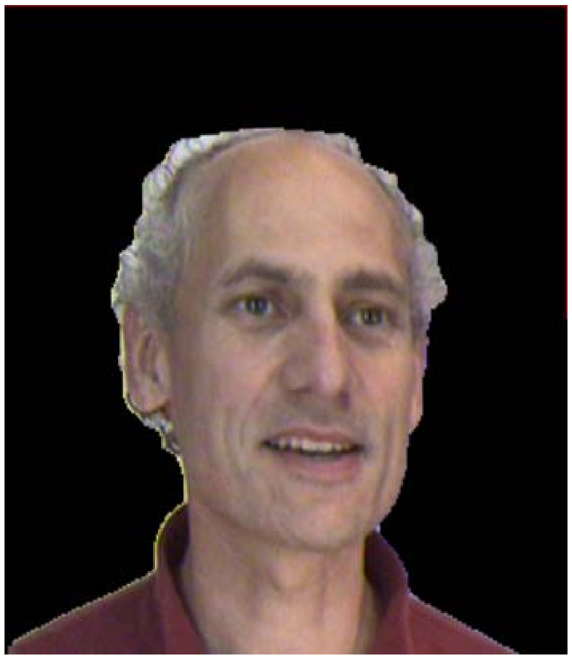


Figure 36 Typical Image Segmentation.

The shape smoothing is broken into the steps of first removing outliers and then controlling the raggedness of the shape. Given the first assumption above, the chroma-keying occurs on a scan-line basis by starting from the left and proceeding to the right edge until a non-background color pixel is hit. Given the noisiness of the image, the algorithm needs to check if the pixel is truly an edge. It therefore continues until it finds two or three background pixels depending on the settings of a system parameter. The process is then repeated from the right. The entire scan-line section between these two edges is considered to be foreground.

Assumption 3 states that the top of the image might be empty. With this knowledge and starting at the bottom of the image, the chroma-keying and shape smoothing can be stopped once an entire scan-line has been crossed without seeing a foreground pixel. This knowledge is exploited in various other sections of the program to provide for early termination of processing.

Two kinds of problems happen with chroma-keying: either some of the background is brought into the image or some of the foreground is keyed out. The leaving of foreground out, producing holes, could show up either on the edges of the subject or in the middle of it. Given assumption 1 above and the way that the foreground is found by processing to the center first from the left and then from the right, the process does not suffer from holes in the middle of subject. It can, however, suffer from holes along the edges.

Given the occurrence of noise in the image, the keying algorithm often suffers from bringing in too much of the background. In fact, due to the stopping of the processing of scan-line from the outer edges to the center when non-background color is found, a couple of pixels of noise can sometimes create an erroneous long thin line extending out of the subject. An analogous error happens when a particular scan-line extends too far in to the foreground. To deal with these outlier scan-lines, extending too far out of or too far into the subject, a post-keying step is applied to limit how much a scan-line edge can jump left or right from its neighboring scan-lines. An empirically derived value of limiting the delta between two adjacent scan-line edges to two pixels has been found to lead to reasonable shapes in practice.

4.5.2.1 <u>Shape as a Signal</u>

Applying the simplistic, and efficient, technique mentioned in the previous paragraph of limiting the change of where scan-lines start and stop based on a fixed number of pixels greatly improves the results of the keying process. However the results are still less than desirable which has led to the need for further processing. Because this is a real-time application, the addition of extra processing into the image creation stream needs to be evaluated for its effect on the computational budget.

Given the large number of pixels processed, even algorithms that run in linear time with the number of pixels stress the capability of the system. Fortunately the shape smoothing operations do not process all the pixels in the image. In fact, because they only process pixels on the edges of the subject, the number of pixels is in a different time order of complexity class. The number of pixels on the edges of the subject can be approximated by the number of pixels on the edge of the image. The number of pixels on the edge of the image is equal to the square root of the number of pixels in the image. Because the processing of the edges is linear with the number of pixels in the edges, the time order complexity of processing the edges is *O*(sqrt(n)) where n is the number of pixels in the image. This allows us to ignore the cost of the extra processing of the edges on the computational budget.

The removal of outliers still left images that left excessively ragged edges. When viewed as a video sequence, this raggedness was exaggerated. One cause of the raggedness was resolution of the geometric mesh used to manipulate the image. Increasing the mesh resolution made for a more smoothly varying image but the image outline still appeared unnatural and displayed great oscillations when viewed as a video. Therefore, methods were investigated to smooth the shape.

The goal can be thought of as trying to control the rate at shape changes. A reasonable first approach was to look at the rate of change of **x** with respect to **y**, i.e. the derivative. After a brief initial investigation, this derivative, the slope of the line between edge pixels on successive scan-lines, proved to unmanageable. Large changes in the **x** values of edges that were nearly horizontal resulted in only small changes in the slope value whereas no change in the **x** on two successive lines resulted in an infinite slope. This quickly led to using the change in the angle between two adjacent scan-line pairs to be the variation to be managed. While working with the change in angle would have been a feasible approach to take, an alternative and simpler solution was found.

The values of **x** between successive scan-lines can be thought of as a signal. This signal, as it comes out of the previous processing step, oscillates too wildly or can be thought of as having too high of a frequency component. To produce a more pleasing and realistic shape, the processing step now becomes that of band-limiting the signal. Given the crudeness of signal, it was determined that only a simple low-pass filter was needed. Therefore, a 3-element box filter was applied. That is to say, the value of **x** was replaced with the average value of itself with its neighbors above and below. This computationally inexpensive task produced good results.

4.6 Warp Mesh Generation

Modern computer graphics libraries provide a construct called a vertex array for handling large numbers of vertices in a regular fashion. The structure of the data as a rectangular mesh of vertices was a prime candidate for this data structure. The vertex data was arranged as rows of quads stacked on top of each other. Typically, the rendering of polygons in graphics libraries is implemented by combinations of triangles. The use of triangles ensures the satisfaction of the constraint that all points be co-planar.

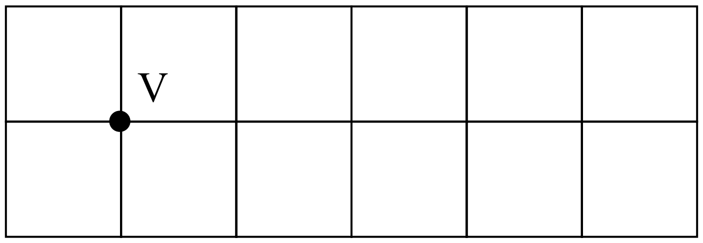


Figure 37 Example of the use of quad strips in the program.

Figure 37 can be used to see the efficiency of using quad strips for specifying the polygonal mesh. It can be seen that vertex **v** can be used for the construction of four quads. Thinking about the internal implementation, each vertex **v** is used for the rendering of eight triangles. Using the OpenGL interface, it is possible to place each vertex into an array only once. The quad strips are specified by passing lists of the indices of the vertices in the array. Given the fixed layout of the quad strips, these lists of indices are independent of the texture content. The list of indices are created once at initialization time and then used throughout the program's duration. Only the values of the vertices change.

Another important point to note is the domain where the warping mesh is created. Ultimately it will be used in the final rendering phase of the graphics pipeline. This suggests that the proper space to create the mesh is in the standard image space. To determine where the mesh should be defined, the behavior of the image as it is distorted is observed. The region in the horizontal center of the image only undergoes a mild distortion. The regions in the left and right parts of the image undergo great distortion. This suggests that a regular sampling in image space would not properly handle the two kinds of behavior. The use of a regular rectangular mesh in NCP space was investigated. With the mesh based in NCP space, it more accurately represents a regular sampling of the surface area of the subject. Indeed, this was the motivation of performing the NCP transform in the first place.

Below are some images exposing the kinds of stretches the mesh undergoes. To generate these images, the normal output of the program for a single view was taken. Instead of displaying the polygons filled in, only the wire mesh of the polygons were displayed. The upper right image is the view with the subject rotated to the extreme left, the bottom left image is the subject in the in between view, and the bottom right image is the subject rotated to the extreme right. Given the resolution of the mesh, it may be difficult to see the varying size of the triangles. However, by noticing the whitish haze of the mesh on the left edge of the upper right image and on the right edge of the lower right image, one gets a sense of how the triangles are compressed as the image rotates.

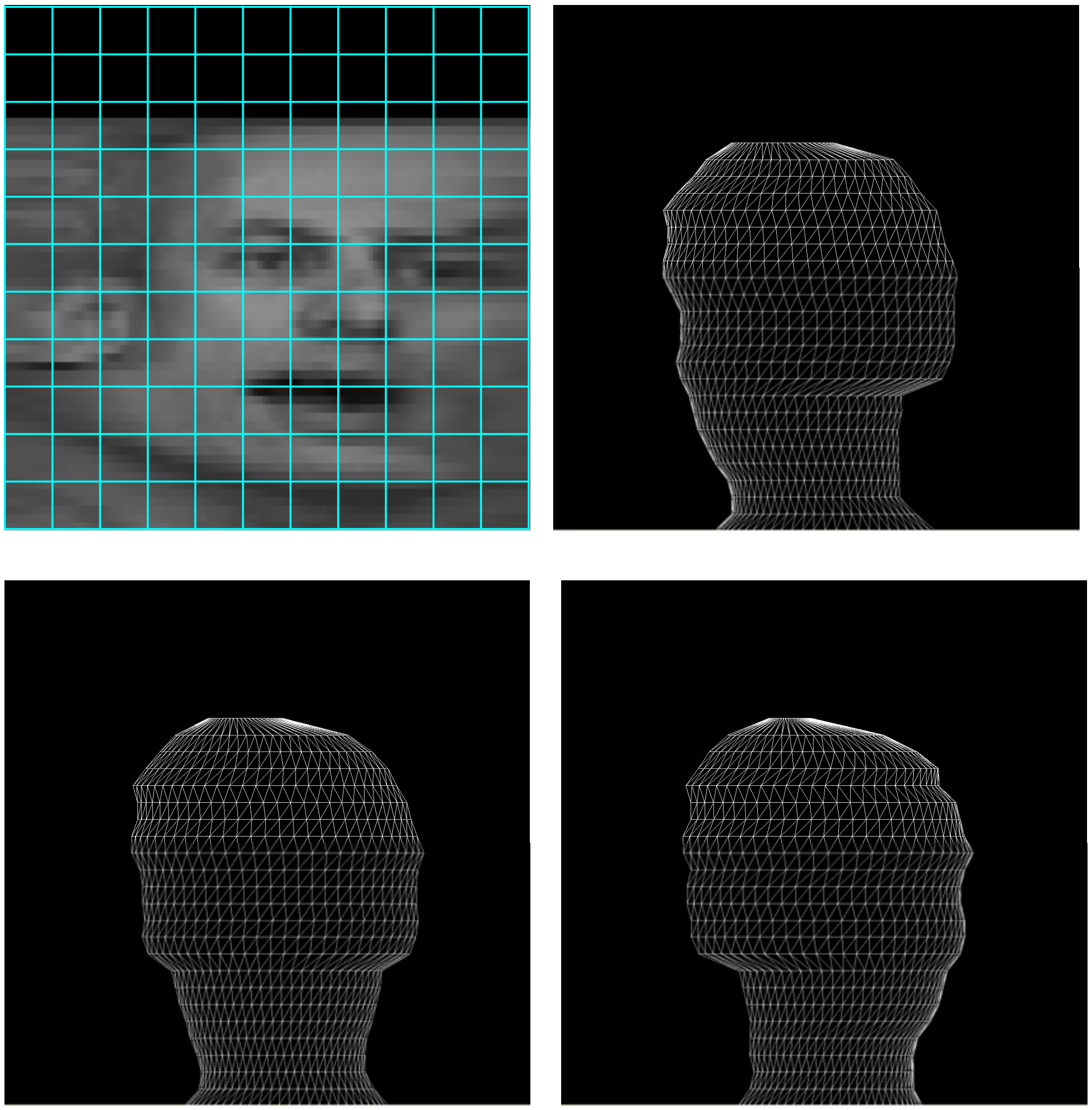


Figure 38 Distribution of the warping mesh in both NCP and image space.

4.7 <u>Minimization of Resampling</u>

One goal of the system is to minimize the resampling of the data as much as possible. The Seitz implementation (Seitz and Dyer, 1996) sampled the data three times: by pre-warping the images into a rectified image plane, interpolating these images to render an image in the rectified plane, and then producing the final image by warping this view morphed image. By taking advantage of OpenGL's geometric vertices, texture coordinates, and its set of transformation operations, the output system generates the final image by directly referencing the source images.

The key to single sampling the source images when generating the output images is that the output of the matching process is vertices only, not image data. During the matching process, only texture coordinates are manipulated, not the texels themselves. When the matching process is complete, the vertices in NCP space are inversely mapped to the original source images for use in the final image pipeline.

4.8 Occlusion

Occlusion is an issue for all stereo camera based systems. It deals with the inability to see a region in one that is visible in another view. If a region is occluded, then there is no solution to what is the correlation between these in points in the image where they are not visible. Therefore, some simple operations to deal with the problem that generally produce reasonable results are performed. Two types of occlusion are distinguished, macro and micro. Macro-occlusion deals with the lack of image information because the region is rotated too far around the main body of the subject. In this application, it is the region on the other side of the head. For example, the left edge of the subject in the left view will not be visible in the right view because the rotation of the subject is too extreme. The other kind of occlusion will be referred to as micro-occlusion. This deals with sub-regions of the subject that are not viewable in the other view because of the extreme protrusion/indentation of geometric entities. An example of this is the left side of the nose being visible in the left view but hidden in the right. The two types of occlusion are differentiated because the solution to them is different.

4.8.1 Macro-occlusion

As stated above, macro-occlusion deals with the image content on the extreme edges of the subject. In the general case, let X be assigned the value of either left or right and Y be assigned the value not assigned to X. Then macro-occlusion is defined as:

> The region of the subject visible in the X view but not visible in the Y view corresponding to the extreme X section of the X view that has been rotated beyond the line of sight of the extreme X section of the Y view.

Some observations about macro-occlusion are made. The assumption is made that it is always present (unless one is dealing with something that has a perfectly flat surface). It is known where it is located in the image. A method of estimating its extent and what to fill the region with is known.

The mechanism for dealing with macro-occlusion is as follows. The concept of macro-occlusion is built into the NCP transform. It states that if the cameras are separated by an angle that is percentage **x** of a semi-circle, then in NCP space the images should be shifted by percentage **x**. The regions of non-overlap are the regions of macro-occlusion. Figure 39 below highlights the location of the areas of macro-occlusion. From viewing the diagram, the solution for handing macro-occlusion can be immediately grasped. For the extreme left/right regions that are only visible from one camera, only use the image information from that one camera.

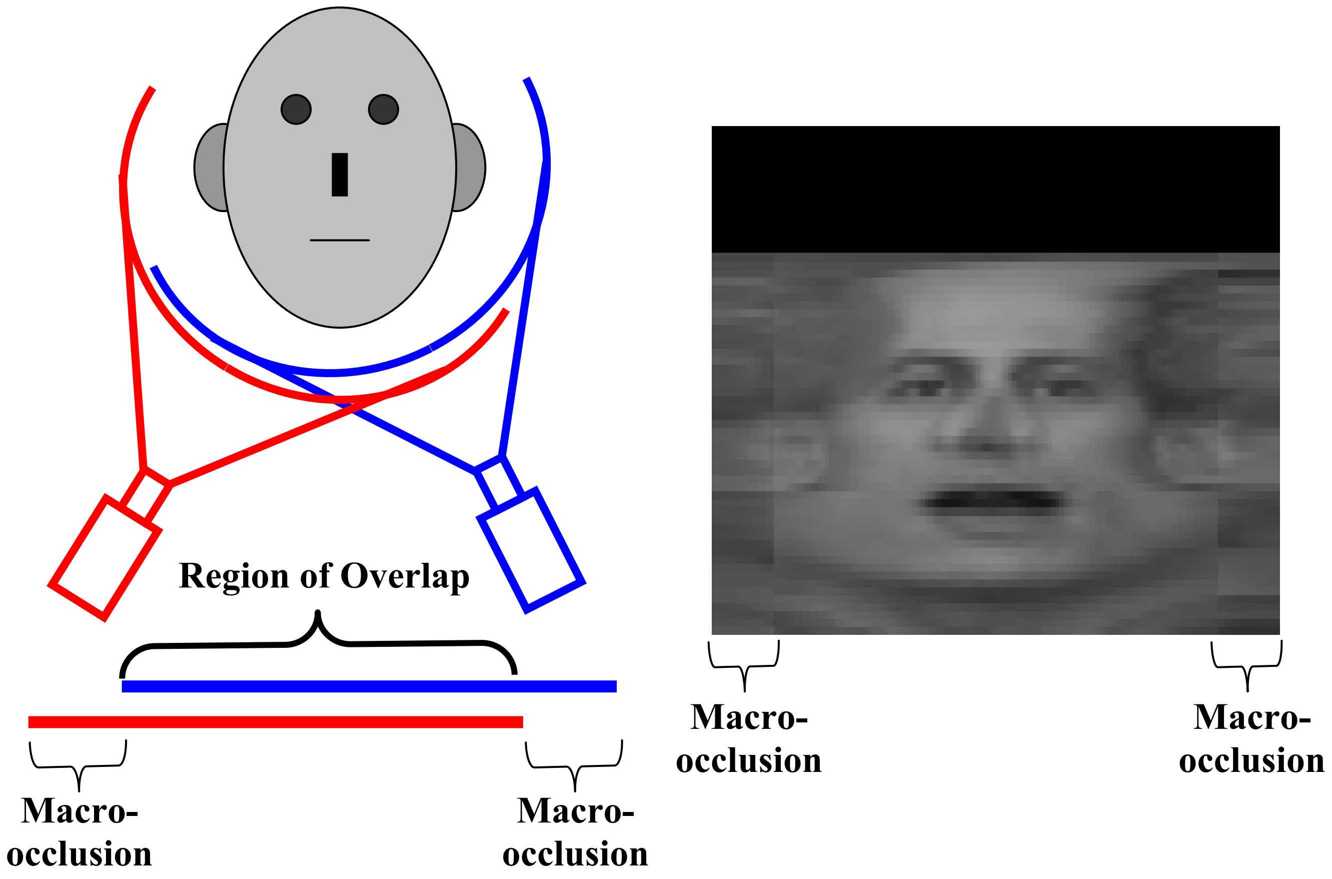


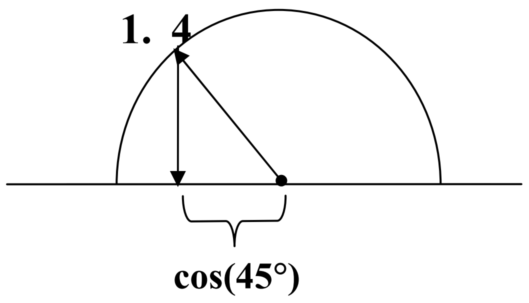
Figure 39 Diagram showing the region of "macro-occlusion".

By having information from one camera, one is not able to perform the view morph operation. Instead a simple linear scaling of the region is preformed. When the point of view is straight on for the camera that has the view of the region, then the scaling of the region will be 100%. When the point of view is the other extreme and that of the other video camera, then the scaling is 0%. For the in between views, the scaling is directly proportional to the angle of rotation. As a practical implementation concern, the border between view morphed and the linearly scaled image will appear as a discontinuity. This edge can be attenuated by performing some alpha blending in this region to produce a more graceful transition.

While the NCP transformation gives an initial guess on what the region of macro-occlusion will be, this is only an approximation. The outside edge of the camera that has the single view is fixed, for example the left edge of the left camera. The inside edge of one camera will be located with respect to the other based on the angle between the cameras, for example the location of the left edge of the right image when placed on the left image. The NCP assumption of projecting onto a semi-circle and knowledge of the angle between the cameras to estimate this location is used. That initial location is then adjusted as the pattern matching is performed.

It might appear to be a concern that there are regions where the view morphing does not apply. While the concern is valid, it is mitigated by a couple facts. First, the region where the view morph is missing is generally not of the most importance in the image. In the scenario where the subject is facing towards the center of the two cameras, the area of macro-occlusion would lie on the side of the subject's head. The face would remain in the region of view morphing.

There is a second mitigating factor on the effect of macro-occlusion. Given the nature of the range of the viewable surface from various angles, the region of macro-occlusion is only a small percentage of the total image. This percentage is smaller that the percentage of the angle between the cameras. For example, when the angle between the cameras is 45°, or 25% of a semi-circle, it might be assumed that the region of macro-occlusion would also be 25%. Using the NCP assumption, it is seen that the region's size is related to the cosine of the angle. In this example, the percentage macro-occlusion is only 15% as seen by the calculation in Figure 40. It is kept in mind that this estimate of macro-occlusion, while insightful, is just a crude estimate. It varies from reality in so far as people's head are not circles and it lacks symmetry due to the arbitrary gaze of the individual. In section 5.4 there is a discussion on how to deal with macro-occlusion.



$$\textbf{\% macro-occlusion} = \frac{1-\cos(45°)}{2} * 100\%$$
$$= 15\%$$

Figure 40 Calculation of region of macro-occlusion using the NCP assumption and where the cameras are separated by 45°.

4.8.2 Micro-Occlusion

The second kind of occlusion is refered to as micro-occlusion. This deals with sub-regions of the subject that are not viewable in the other view because of the extreme protrusion/indentation of geometric entities (e.g. one side of the nose being visible in the one view but hidden in the other). Micro-occlusion must be handled differently than macro-occlusion because it lies in the middle of the view morphing range. The correspondence mechanism that is at the heart of view morphing fails completely because there is a lack of information. Fortunately, it has been observed (Seitz and Dyer, 1995), that local failures of the view morphing assumption of monotonicity do not affect the global performance of the algorithm. Missing information of localized regions of the data on input show up as localized distortions in final image. The kinds of error that are seen in these regions are blurring and ghosting. Ghosting is the situation where two disparate regions are placed on top of each other on the output.

A simple solution to the micro-occlusion problem is performed that produces generally good results. The approach is to assume a continuous surface that does not exhibit micro-occlusion. This has the effect of glossing over and the blurring of regions that are missing. The matching algorithm is such that it tries to find the best distortion of the source image to match the destination image. It works with the best information that it has. It distorts the source image in such a way to minimize differences in the destination image. Given the generally constant flesh tone of the human face, small discontinuities in the matching process do not standout in the final composited image. Alternately, one can think of the approach as that of a simple hole filling algorithm where holes of missing image information is simply filled with the pixels from the nearby data.

4.9 OpenGL vs. DirectX

The two main 3D imaging libraries, OpenGL and DirectX, each have their advantages. Microsoft's DirectX set of interfaces, DirectDraw, Direct3D and DirectShow, have nice advantages over OpenGL in that it fully integrates management of video with computer graphics. Ultimately, OpenGL was chosen as the basis of its ability to run on multiple platforms. The application has been ported to SGI's IRIX and Sun's SPARC Solaris operating systems along with Windows based PCs. In addition to supporting OpenGL, all these platforms had implementations of the GLUT library that provided a common windowing and mouse and keyboard interface in these heterogeneous environments.

After exploring the video texture implementation of video textures in Microsoft's DirectX imaging library, it was decided to implement a proprietary one using OpenGL primitives. The implementation consisted of simply a front and back buffer for each image texture. The front image texture buffer was the one given to the back buffer of the OpenGL pipeline for rendering. While the image's front texture buffer was in use by OpenGL, the texture's back buffer was loaded from the movie file and then processed by the matching subsystem.

4.10 Fixed-Point vs. Floating-Point

The decision to use fixed-point or floating point was a critical one early in the design of the program. One advantage of floating point representation is the general flexibility of being able to work with a large range of numbers. Another is to not have to concern oneself with the scaling factor adjustments when performing multiplies for example. While division in general is an expensive operation, division by a constant known ahead of time can be performed by a multiply. For example, division by 3 can be performed by a multiply by 0.333333. Floating point multiplies can usually be considered to be faster than integer multiplies with the exception of using the SIMD (Single Instruction Multiple Data) instructions now available processor such as the MMX instructions on a PC.

There are several advantages of using fixed-point arithmetic. One advantage is that multiples and division by powers of two can be implemented by shifting and hence can be performed very fast. Another minor advantage is a smaller data format. When dealing with texture coordinates for example, 16 bits can suffice instead of 32. This size advantage is not of much concern on today's computers.

Originally, a fixed-point format was chosen for manipulating texture coordinates. The main motivation was an idiosyncrasy dealing with the PC floating-point architecture.

To render intermediate resolution images, a MIP-map texture mapping operation must be performed. To determine which MIP-map to use, a logarithmic operation must be performed to determine which MIP-map level to reference. An actual logarithmic operation is too expensive to perform so a table look up is performed instead. The size of the lookup table is quite reasonable in size since the texture sizes being used are small. If one is using a floating-point texture coordinate is used, it must be first converted to an integer. This seems like it should be a very cheap operation. Indeed, there is an assembly language instruction that will convert a floating-point value to an integer. There is a big caveat however. That instruction uses whatever the current round off mode is of the processor. When performing floating-point calculations, one generally wants the mode to be round rather than truncate. However, the ANSI C standard states that the casting of floats to ints should be a truncating operation. To guarantee a truncation, compilers go through a very expensive sequence of operations.

First the current state of the floating-point processor must be saved. Then the floating-point state of truncate is loaded into the floating-point flag register of the processor. This operation is expensive because before the state can be set, the floating-point pipeline must be drained. The wait is the maximum pipeline depth. Then the new flag is set. The single machine instruction conversion operation is then performed. Now the state of the floating-point processor must be restored. Again, one must wait for the floating-point pipeline to drain. To perform all these operations a subroutine is called. Timings of the whole process show that the whole operation takes about 100 clocks. Because there are several table lookups for each texture point rendered, a floating-point representation was not used.

With time, two things occurred to change the opinion on the use of floating-point. First, to take the advantage of fixed-point's fast division, many data structures and operations were limited to powers of two. More flexibility was needed for creating scan-line variations. A scan-line variation rendering algorithm was devised that did not use conventional MIP-maps.

The conversion from fixed to floating-point was a major under taking. The program was permeated with the fixed-point calculations and changing any one part of the program had cascading affects. With much effect the program was debugged and a pure floating-point version was implemented.

The use of floating-point was extended to pixel data information also. All matching is performed on gray-scale versions of the images. In an early version of the program, this gray scale data was represented with a byte. With the conversion from fixed to floating-point texture coordinate calculations, the gray-scale information was converted to floating-point also. This insured no significant loss of precision when generating scan-line variations and also when calculating SSD comparisons.

In general, the use of floating-point format for both the gray-scale and the coordinate information, simplified the code needed to perform tasks, opened up flexibility by allowing for non power of two variations of data, and likely increased the performance of the program given the high efficiencies of the floating-point operations on today's processors.

4.11 SSD Variations

Recall the SSD, Sum of Squared Differences, operation. Consider two scan-lines whose values are represented as N-tuples S0 = (p01, p02, p03, … , p0N) and S1 = (p11, p12, p13, … , p1N), then the SSD operator can be defined as:

SSD = sqrt((p01-p11)^2 + (p01-p11)^2 + … + (p0N-p1N)^2)

Taking the square root will gives the actual measure of the distance of two scan-lines in the N-dimensional problem space. For the purpose of efficiency when deciding which scan-line is the better match, the square root term can be dropped.

In the actual implementation, a common variation on the standard SSD operation is made called the weighted SSD. A heuristic assumption is made that pixels in the center of the image are more important. This is done this for several reasons. First, the center regions of the region undergo a smaller distortion from the NCP operation. Second, the relative distortion between corresponding regions of NCP aligned images is smaller in the central regions of the images. Third, in the usual case where the subject is facing the camera, the information in the central region of the images is more important. For example, getting the eyes, mouth, and nose aligned is more important that aligning the hair on the side of one's head.

The weighting function used is simply that of a semi-circle that has been scaled vertically. In Figure 41 the distribution of weights can be seen visually.

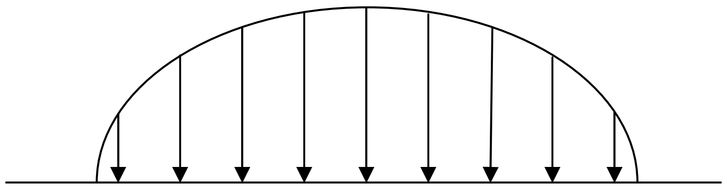


Figure 41 Diagram of the SSD weighting factor. Pixels in the center are considered more important.

4.12 Matching Issues Due to Image Quality

The accuracy of the matching depends fundamentally on the quality of the images entering the system. Following is some discussion on the factors that affect the quality of matching.

4.12.1 Video Camera Differences

Although, some initial investigations demonstrated success with unmatched cameras, using identical cameras leads to better results. Working with different kinds of video cameras involves trying to resolve variation in the gamma values of various computer types. It is a non-trivial task that needs its own set of expertise. The video multiplexer used did allow for per-stream color correction. This feature was taken advantage of with an early set up that had different cameras. With a lot of tweaking, better color matching of the streams was attainable but the results were limited. The use of identical cameras eliminates all these issues.

4.12.2 <u>Noise</u>

All video devices possess some noise in the images that needs to be dealt with. In general, the cheaper the camera, the more noise there will be. High quality cameras would have been desirable but the experimental set up was greatly limited by available funds. The cameras used in the research were the cheapest conventional camcorders available, in the $200 range. The noise in the images can be mitigated by post processing the images. Various techniques exist for its removal (Trucco and Verri, 1998). The manner to deal with the noise is dependent on the kind of noise in the image.

One type of noise is referred to as impulsive or “salt and pepper” noise. This kind of noise is characterized by random values, and generally, very different from their true values and very different from the neighboring pixels. It appears as a sprinkling of light and dark spots in the image. It can be caused by faulty CCD elements, transmission errors, and noise in the analogue to digital conversion process. A good way to deal with impulsive noise is the median filter. This filter replaces a pixel with the median of the pixels in it neighborhood, e.g. a 3x3 grid. This filter is effective for impulsive noise because its resulting value is not affected by the severity of the erroneous pixel values.

Another model is to assume that the noise is Gaussian. Having made the claim that the noise behaves in a Gaussian manner, there is a whole body of theory for how to deal with it. By convolving the image with a Gaussian filter, the noise can be smoothed away (Trucco and Verri, 1998). A more simplistic approach to dealing with noise in an image, somewhat related to Gaussian noise, is to use a mean filter.

The mechanism used in this dissertation to deal with noise was sub-sampling the image after a simple box filter was passed over the image. This is a crude low-pass filter implemented by averaging neighboring pixels together. This has the effect of minimizing the noise away. It should be noted that noise reduction was not the intent of the box filtering. A reduced size image was used for performance reasons. The noise reduction effect of this reduction is a bonus.

5. FURTHER ANALYSIS

This section deals with topics that are not necessary to get initial results from the view morphing system but are useful enhancements. Topics include understanding the relationship between a linearly interpolated view morph versus a virtual camera traveling in an arc, how to deal with macro-occlusion, and several ideas about performance enhancement.

5.1 Chord to Arc Adjustment

The goal of the view morphing is to provide the illusion of a rotation around a 3D object. The effect of a view morph is transversal of the straight line rather than an arc. Looking at Figure 42, the differences between these two values can be seen. It is desired that the camera to be positioned on **arc AC** passing through point **B'**. Instead, view morph camera moves along the **chord AC** passing through point **B**.

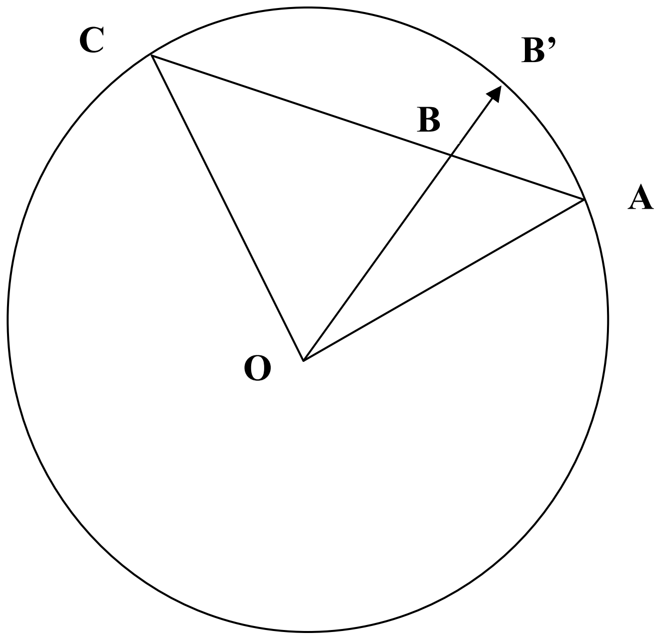


Figure 42 Chord - Arc Comparison

5.1.1 <u>Chord to Arc Angle Adjustment</u>

It is preferred that the virtual camera to travel along the arc **AC** but the view morph travels along the chord **AC**. In general, the ratio of the chords **AB /AC** is not the same as the ratio of the arcs **AB' /AC**. What is the difference between the percentage of the chord subtended versus the percentage of the arc subtended? That difference will be determined in the following calculations. Without loss of generality, a coordinate system is chosen such that the origin of the circle **O** is at (0,0) and point **A** is at (0,1).

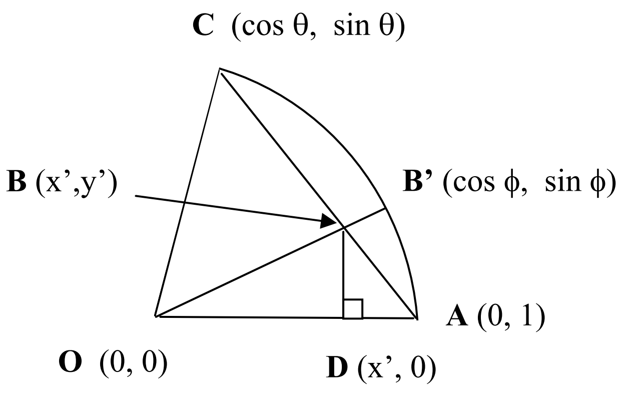


Figure 43 Chord – Arc angle subtended.

The ratio of the lengths of the sub-chord **AB** to arc **AB'** can be found by finding the location of point B. It is realized that point **B** is at the intersection of the lines **OB** and **CA**. Let the angle between the stationary cameras be θ and the angle of the virtual camera be ϕ. Let subscript 1 be used for line **OB** and subscript 2 be used for line **CA**. For line **OB** it is noted that this line passes through the origin and that its slope is determined by the ratio of sin ϕ to cos ϕ. Therefore:

$$y_1 = m_1 x + c_1 \text{ where } m_1 = \sin\phi / \cos\phi \text{ and } c_1 = 0$$

For line **CA** the slope is determined by noting that **C** is located at (sin θ, cos θ) and that **A** is located at (1, 0). The constant term is calculated by observing that y = 0 when x = 1, hence the constant term is just the opposite of the slope.

$$y_2 = m_2 x + c_2 \text{ where } m_2 = \sin\theta / (1 - \cos\theta) \text{ and } c_2 = - m_2$$

The location of point **B** can be solved for by finding the intersection of these two lines. Solving for x first gives:

$$m_1 x = m_2 x + c_2$$

$$x' = c_2 / (m_1 - m_2)$$

Substituting x into the equation for line **OB** gives:

$$y' = m_1 (c_2 / (m_1 - m_2))$$

The length of the arc subtend is π r ϕ. Because r is one, this gives:

$$\text{arc } \mathbf{AB'} = \pi \phi$$

To calculate the length of length of the sub-chord **AB**, the values x' and y' calculated in the previous section are used. By Pythagorean's theorem:

$$\mathbf{AB} = \text{sqrt}(\mathbf{DA}^2 + \mathbf{DB}^2), \text{ where}$$

$$\mathbf{DA} = 1 - x' \text{ and}$$

$$\mathbf{DB} = y'$$

The total length of the chord is:

$$\mathbf{AC} = \text{sqrt}((1 - \cos\theta)^2 + \sin\theta^2)$$

Hence, given the angle ϕ, the amount that the virtual camera should be moved is:

$$\alpha = \mathbf{AB} / \mathbf{AC}$$

In the table below, the term "View Morph Alpha" refers to how much of the view morph chord must be traversed for the given angle percentage. The general observation is made that the larger the camera separation angle the more significant the difference between the chord and alpha values. Even without looking at the table one observation that can be made is that, in addition to the end points, at the halfway point the angle and alpha are in synch. That is, at the midpoint of the angle the chord is also at the midpoint of the arc.

Looking at the table the following observations are made. At the angle 45° and below the difference of the chord traversed and the angle is not significant. At the angle of 90° the angle subtended is significant between the range of 10° - 30° and, by symmetry, the range 70° - 90°.

TABLE I RELATIONSHIP BETWEEN ANGLE AND VIEW MORPH ALPHA

Percent Angle	View Morph Alpha		
	90°	45°	22.5°
0%	0.00	0.00	0.00
10%	0.14	0.11	0.10
20%	0.25	0.21	0.20
30%	0.34	0.31	0.30
40%	0.42	0.40	0.40
50%	0.50	0.50	0.50
60%	0.58	0.60	0.60
70%	0.66	0.69	0.70
80%	0.75	0.79	0.80
90%	0.86	0.89	0.90
100%	1.00	1.00	1.00

5.2 Multi-Threading Performance Enhancement

The natural structure of the algorithm is such that its performance could be enhanced by multiple processors. Performance can be enhanced by exploiting both parallelism and pipelining.

5.2.1 Parallelism

There are various kinds of parallelism in the system that can be taken advantage of if needed. At the lowest level are data computations that can be speed up with SIMD (Single Instruction Multiple Data) processor systems. Because they operate on data organized in regular grids, some of the image processing aspects of the system are candidates for SIMD operations. For example the generation of the extended MIP-maps.

On a multi-processor computer, the system can make use of two kinds of parallelism. In a divide and conquer approach, multiple processors can be working in parallel on the same stage of the process but on different data. For example, the scan-line orientation of the system makes the scan-lines generally independent of each other. When performing the image correlation, two processors could be put to use by dedicating one to the top half and the other to the bottom half of the image. **N** processors could be used by allocating each one 1 / **N** of the scan-lines to work on. In addition, on a two camera system, most stages of the entire process can be thought of consisting of two parallel tasks, e.g. two image capture, two extended MIP-map generations, two image correlations, etc.

5.2.2 Pipelining

A second kind of parallelism to be taken advantage of is pipelining. The different stages of the sequential processing are independent. Different processors or threads of control can be allocated to the various sequential stages in the processing below.

- Image capture
- Extended MIP-map generation

- Base level correlation of this image with the other
- Multiple passes of refinement of this image with the other

5.3 Multi-resolution sub-system resource allocation

A major architecture feature that supports the efficient use of computation resources is how the system is laid out in a multi-resolution manner in both space and time. For example, with respect to space, the resolution of the texture may be 256 x 256 but the geometric vertices might be ¼ of that. Given how the brain's perceives information, different parts of the system can be performed at lower spatial resolutions and lower frame rates without affecting the general perception. This multi-resolution processing allows a more resources to be allocated to selected areas thereby providing higher overall frame rate of the system.

Below are the areas of multi-resolution broken out by time and space from the highest to the lowest resolutions/rates.

- Time
  - Rendering frame rate to the screen of morph including changes in viewing angle
  - Rate at which the video textures are updated
  - Rate at which matching is performed
- Space
  - Texture resolution
  - Geometric mesh resolution
  - Coarse to fine matching resolutions

5.3.1 Multi-resolution Time

Current commodity graphics hardware can render frame rates at monitor refresh rates. The last stage of processing, the geometric transformations and texture mapping can be performed at these rates. By updating the angle of viewpoint at these rates, smooth rotations can be performed even when the video rate is slower.

Finding the correlation correspondence between images is the most computationally intensive activity of the system. It is not likely to be performed at monitor refresh rates. Fortunately, this rate matching is not generally needed. Geometry changes slower than texture data. Consider the view of someone speaking. The lip are moving rapidly, occasionally there is a blink or a wrinkling of a brow but the general geometry of the head, due to reorientation generally does not change as quickly. Updating the texture data at video frame rates can create perceptibly accurate images even if the geometry is updated at slower rates. Reasonable frame rates per second are 70 for the screen rendering, 30 for the video texture updating, and 10 to 15 for the geometry updates from the matching algorithm.

5.3.2 Multi-resolution Space

The multi-resolution of space is generally exploited in computer graphics as typically witnessed by using polygons that are larger than individual pixels. A sphere is implemented with a relatively small number of vertices with individual pixels being lit and shaded to create an illusion of roundness. In the case of this project, if the texture data has a resolution of N x N, the rectangular mesh that is used to distort the image data has a resolution of ½ or ¼ of that in each dimension.

The system has another manner in which multi-resolution space is exploited. Even by exploiting scan-line algorithms, the number permutations involved in the matching process can grow exponentially causing even moderate resolutions to produce computationally intractable obstacles. For example, if each data point on the scan-line was tested in three positions, left, right, and center, then a scan-line of only 32 data points would result in 3^32 combinations of modified images or 1.85e15 set of modified scan-lines to compare. By using a smaller set of control points, a real-time computationally viable set of scan-line comparisons can be generated.

5.4 Removal of Macro-occlusion

In section 4.8.1 macro-occlusion was defined as the inability to perform view morphing on a region of the subject due to part of the subject not being seen in the image of one camera as the result of the region being rotated out of view. In this section what can be done to deal with macro-occlusion will be discussed briefly.

In the version of the system implemented for this dissertation, the complexity and computational cost was simplified by only using the input from two video cameras at any one time. Therefore only a single region of overlap was correlated. For an ideal system that generates correlations on all regions of overlapping images, the system will, in general, have three separate overlap regions. Each of these three regions will need to use the images from two cameras to create a view morph. Because adjacent regions share a camera, four cameras are needed to create the three regions of double coverage.

Consider instead a system that provides 360 degree viewing in a single plane by placing four cameras, each 90 degrees apart and assume a subject that generally behaves well under the NCP transform. From Figure 44, one can see that to create the full virtual camera image, four real cameras are needed.

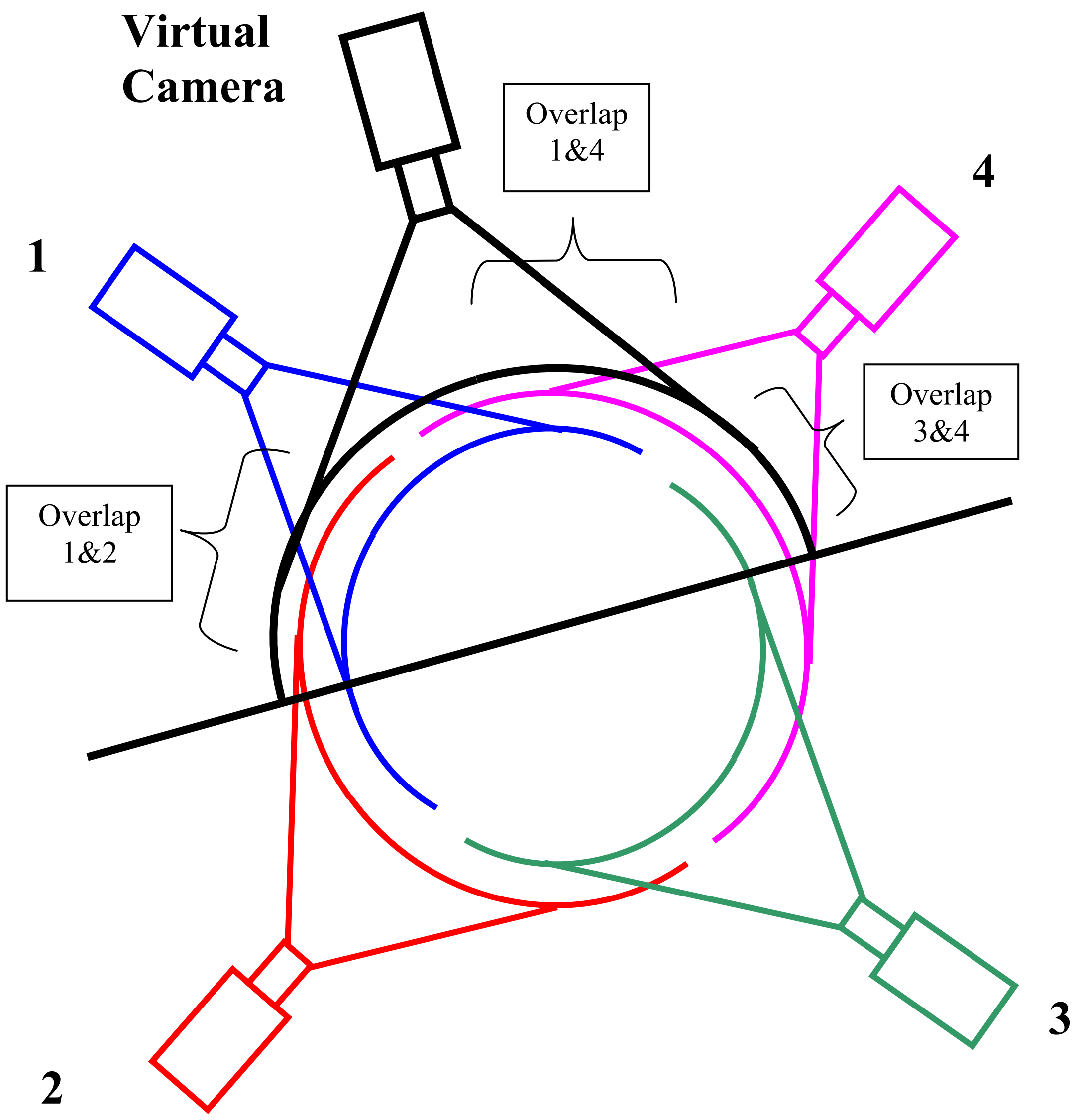


Figure 44 Diagram of overlapping cameras for full view morph coverage.

5.5 Working In 2D Is Better Than 3D

Something that was initially just a design consideration chosen for performance reasons that later evolved into an intellectual challenge and a sub-goal of the project is the following.

- To implement the entire system by performing only 2D operations.

While the images generated give correct 2D projections of the 3D scenes, a **z** value is never manipulated to create those projections. When a **z** value is necessary such as passing vertices to the OpenGL graphics library, it is always set to zero. It might seen non-intuitive, but there are cases in which, given two 2D images of a 3D scene, one can construct a novel 2D projection of that scene even though the 3D reconstruction of the scene would be ambiguous.

To understand this phenomenon, consider the "polar bear in a snowstorm" example. Given two images from a scene that is all white, one would not be able to reconstruct the 3D scene information due to ambiguity in the data. For example, one could not locate a polar bear in this scene. But one could easily construct a 2D projection of that scene from a new vantage point. The new image would simply also be all white. While this is an extreme example, similar behavior in localized regions of everyday images are easy to imagine, e.g. a mono-chromatic wall of a room or a cloudless blue sky.

5.6 <u>Computer Vision and Computer Graphics</u>

Computer graphics is a synthesis operation that has traditionally taken geometric models and generated images. Computer vision is an analysis operation that has traditionally taken images and tried to generate geometric models. Recently, there has been a trend towards the synergistic blending of these two fields (Lengyel, 1998). Many of the innovative advances in computer graphics in recent years have been the integration of sampled representations (Debevec, 1998), (Rademacher and Bishop, 1998), and

(Shade et al., 1998). Computer graphics researchers have taken great interest in using real world images as a new primitive for the creation of computer generated images and have created the field of Image-Based Modeling and Rendering. Computer graphics benefits from the vision techniques that have been developed to analyze, segment, and manipulate these sampled representations.

Whereas at first appearance, the topic of this dissertation is that of computer graphics, its core research area is that of computer vision. Once having analyzed the correspondences between the two video streams, the actual rendering to the screen is rather straightforward.

6. SOFTWARE ARCHITECTURE

6.1 Data Structures

An object-oriented design was used to create the software architecture. The program was organized into the following major classes.

- **MoviePlayer** – provides user interface to the whole system by taking the data from a **MovieSeq** and manipulating that data with the **View** class
- **MovieSeq** – manages a time stamped sequence of images
- **View** – virtual camera which contains a **Match**, a **Morph**, and two **Imgs**
- **Morph** – knows how to blend two images into a view morph
- **Match** – takes two **Imgs** and finds the correspondences
- **PathSearch** – performs the low-level pattern matching operations
- **Img** – holds all the pixel and other data for a single image
- **Xmip** – the Extend MIP-map data structure
- **NCP** – provides the Normalized Cylindrical Projection transform
- **XmipNCP** – hybrid class composed of the **Xmip** and **NCP** classes
- **Calibration** – used to perform camera calibration
- **ChromaKey** – used for performing chroma-keying

The sections below have been divided the classes into higher-level classes that perform coordinating tasks in the system and the lower-level classes that implement specific data structures and perform well-contained operations.

6.2 Higher Level Classes

The higher-level classes in the system are **Match**, **Morph**, and **View** for manipulating individual movie frames of image pairs and **MovieSeq** and **MoviePlayer** for storing and displaying time stamped sequences of images.

6.2.1 MoviePlayer

This data structure is the highest-level class of the system and is used to coordinate all the other classes and it controls the user interface. It takes the data from a **MovieSeq** and the input from the user from either the mouse or keyboard and sends the information to the screen. It interacts directly with the operating system and the window manager. A goal in the implementation was to make the software portable by making use of the **GLUT** OpenGL library (Woo et al., 1997). The **GLUT** library has versions that run under a variety of window managers and operating systems. By limiting all references to window management and the user input operations to those provided by **GLUT**, the program was able to be run on a Windows PC, SGI IRIX, and Sun Solaris systems by just recompiling the software.

6.2.2 MovieSeq

The **MovieSeq** class handles managing the time stamped sequences of multiple images. A conventional movie format file would contain a single image at each interval in time. For a view morphing system multiple images are needed. A single time stamp is referred to as a frame. A frame can be composed of two or more panes. In the basic system implemented, a frame consists of a left and right pane but a more elaborate system with more cameras would contain more panes.

In the system implemented, the image data was stored to disk and the **MovieSeq** class would read this data from a file. The class provides basic functions that return the next frame based on a time stamp. The **MoviePlayer** class does not know where the **MovieSeq** class gets the image data. To provide a system that used live video data, instead of prerecorded images, the **MovieSeq** class could be modified or a sub-class created to retrieve live time stamped video data.

6.2.3 View

The View class is what ultimately generates the view morphed image on the screen. It has pointers to two images referred to as the left and right images. Not containing the images better supports the ability to share them on multi-view system using more than two cameras, see Figure 45. Critical to **Img** objects being able to be shared between **View** objects is that the **Img** class not contain any knowledge about its neighbors, i.e. it does not contain any match information. The **View** class also has **Match** and **Morph** classes. These classes do virtually all the work of performing the match and view morphing with the **View** class mainly just coordinating operations between them.

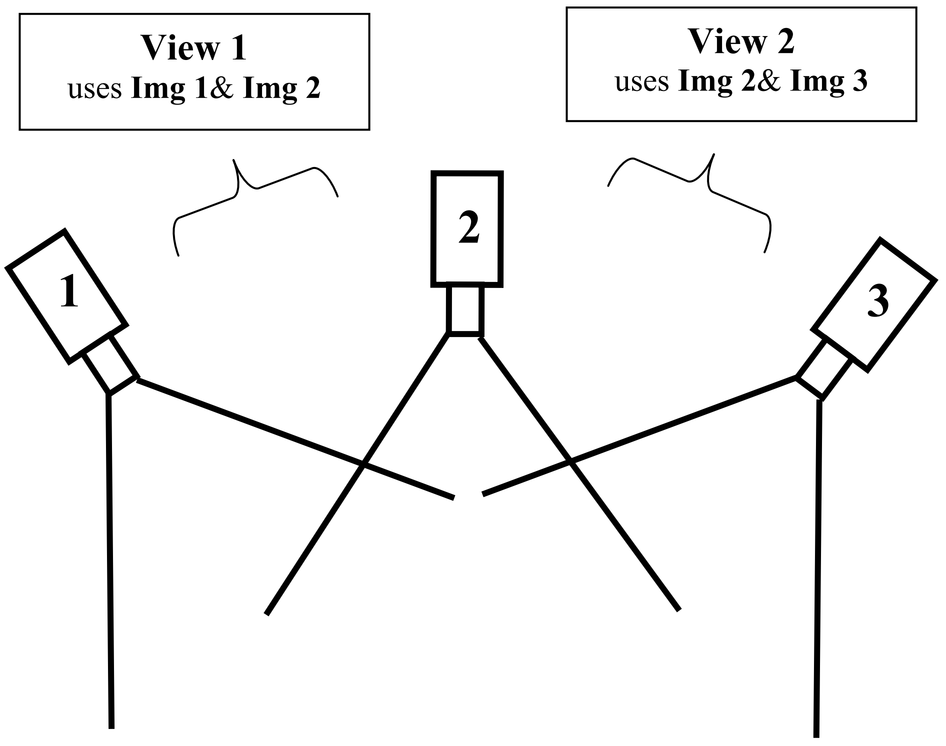


Figure 45 The *View* class supports the sharing of *Img* objects.

6.2.4 Match

The **Match** class manages the operation of matching in the system. While it does perform some matching itself, the most computationally intensive searching for the best scan-line variation for a match is performed in the lower level class **PathSearch**. The **Match** class handles the storage of vertices and texture coordinates as matching progresses through its various stages. The **Match** class also handles the multi-resolution nature of matching providing routines for doubling rows and columns of match points.

6.2.5 Morph

The **Morph** class takes the data from the **Match** class and generates view morphed images. Using the viewing angle as input, it performs interpolation operations on the match data. It is the class that actually draws the image on the screen. For that reason it must have specific information about the 3D graphics library. While much of the computation is done in the **Morph** base class, the base class is an abstract class. To actually draw the information, the sub-class **Morph_ogl** is created. From the name one can discern that it is an OpenGL implementation but other graphics libraries could be used. To support another graphics library such as DirectX, only about half a dozen routines need to be implemented in the subclass.

6.3 Lower Level Classes

The lower level classes in the system would be **Img**, **Xmip**, **NCP**, **XmipNCP**, **PathSearch**, **Calibration**, and **ChromaKey**.

6.3.1 Img

The **Img** class stores information and provides utilities for manipulating a single image. It is where the actual pixel data is stored. It contains this pixel information in various formats: a color image, a gray scale version of the image, and a rectified gray scale image. It manages the storage of the color image by placing it into the texture memory of the graphics system and retains a pointer to it. The **Img** data structure has a **XmipNCP**. The **XmipNCP** has a copy of the rectified gray scale data that has first been transformed by the NCP operator and then expanded by the **Xmip** class.

In addition to the pixel data of the image in various formats, the **Img** class also contains structural information. It contains the start and stop location of the segmented image on a row basis. It contains an initial set of texture coordinates and vertices prior to any matching operations. The last values are created by a regular rectangular sampling in rectified NCP space and then mapping them back to image coordinates.

6.3.2 Xmip

The **Xmip** class creates and contains the data for the extended MIP-map data structure that was created as part of this dissertation. It is very well contained and possesses routines to create the data structure. The bulk of its interface is a variety of access routines to reference the various image and row levels in those images.

6.3.3 NCP

The **NCP** class implements the Normalized Cylindrical Projection transform. This class is quite small and provides routines for generating the NCP and for performing an inverse NCP mapping.

6.3.4 XmipNCP

The **XmipNCP** is a hybrid class that combines both the **Xmip** and **NCP** classes. It is implemented as a subclass of the **Xmip** and has a **NCP**. It main challenge is to efficiently create the **XmipNCP** image data. It provides various access functions to get at the data.

6.3.5 PathSearch

The **PathSearch** class is where the sophisticated pattern matching of the system is performed. It is the most computationally intensive of all the classes. Given two scan-line rows of pixels, it tries to find the best distortion to map one of the rows to the other. The details of how it performs its matching operations are described in section 3.5.3.

6.3.6 Calibration

The **Calibration** class is used to implement the operations needed to perform camera calibration. After the calibration is performed, the class is referenced throughout the running of the application to perform various image rectification and un-rectification operations. It makes extensive use of matrices and references the **Matrix** class.

6.3.7 ChromaKey

The **ChromaKey** class is used to segment the subject from the background. To perform this task it has the ability to perform color space conversions. In addition to performing chroma-keying to segment the image, it performs shape smoothing operations to provide a clean, well shaped segmentation to the program. See section 4.5.2 for more details about these operations.

6.4 Utility Programs

A variety of programs are used to implement the system. The main program that implements the ideas in this dissertation is called **xform**. It takes sequences of images and in real-time performs the matching on them and then displays the morphed images to the screen. Two additional programs used are utilities that help with generating the proprietary movie format files that can be used instead of live video.

6.4.1 Avi2mseq

Working on a Windows based PC, the standard format for video capture programs is that of the **.avi** format. Using this format directly was not desirable. First it does not support a stereo image format (as far as the author is aware of). Theoretically one could take the images from the two streams and after the fact put them together in a stacked format that is twice as tall. But, it is not clear if standard AVI movie players would play this unconventional format. More importantly is that portability is paramount as one of the goals of the movie format. While the AVI format is generally supported on Windows, it is not readily supported on UNIX platforms. A viewer was created to display the proprietary format. Having written the code for the viewer, one can recompile it on any platform that they are interested in. In addition to the viewer program, which the main **xform** program is an example of, a program was needed to convert the AVI files to the "movie sequence" format files. The utility created was called **avi2mseq**. At is core is the **MovieSeq** class described in section 6.2.2. More details about the file format is described in section 4.3.

6.4.2 ChromaKey

Using commodity video capture hardware, built in chroma-keying capabilities were not provided. While high-end video capture/processing hardware can perform chroma-keying in real-time, the equipment and hardware used only had the ability to save the raw video to disk. Identification of the chroma-key regions was only done after the video had been captured. To determine the range of color values that cut out the background completely but do not cut into the foreground, a custom utility was written. This program was based on the **Movieseq** and **Chromakey** classes. Determining the colors can be a tricky business. Instead of the user specifying the colors abstractly by numeric values, the background colors are chosen from the images displayed using a mouse. More details on this process is explained in section 4.5.1.

7. RESULTS

The results of the system can be evaluated on different criteria. The following characteristics were looked at: performance, image quality, and comparison with other systems.

7.1 Performance

Many factors influence performance. The work of this dissertation concentrated on the matching algorithm. Another important factor is the size or resolution of the video data. This data needs to be copied through the system both to and from main memory but also to and from the graphics card. While a newer state of the art PC may have been able to handle the data rates that used, the computer used for development was stressed by the data throughput: a 1.2 GHz AMD Athlon processor with RAM speed of 133MHz and an AGP graphics card. Some additional testing was performed on 2.4 GHz Pentium 4.

One of the processing steps that must be performed before matching can take place is image rectification. This operation was performed by mapping the original texture data to a keystone shaped quadrilateral to perspectively distort the image. This is an operation that the graphics card can perform extremely fast. Using OpenGL, the program was able to send the texture to the card, distort it, and read it back with good image quality. While performing analysis on the performance of the system, it was discovered that is operation consumed 27% of the total processing time. Although it was never tried, it is speculated that performing this operation with a scan-line algorithm by the main CPU would have been faster (Wolberg, 1990). Commodity graphics cards, such as the one used in this system, are very good at sending data from the CPU, to the card, and then to the display. They are less efficient at reading the information from the graphics card back into main memory after it has been transformed.

Another factor in the performance of the system is the size of the video data that was being matched and displayed. The low-level matching algorithm was generally immune to the size of the video data because the bottom level match was performed on the same size reduced image. There was additional overhead with the larger texture size when shipping the data around the system and performing the rectification process mentioned in the previous paragraph. Below is a table showing the performance of the system performing just the NCP operation versus the system performing the full match. The frame rates are shown for both 128 x 128 and 256 x 256 size textures. The video rate was 30 frames a second, the refresh rate of the monitor was 70 Hz, and image data was 24 bit.

TABLE II PERFORMANCE OF THE VIEW MORPHING SYSTEM

Texture Size	**Processor Speed**	**NCP Only fps**	**Full Match fps**
128	1.2 GHz	70	23
256	1.2GHz	23	17
256	2.4Ghz	70	46

From the table, it is seen that the texture size of 256 is bandwidth limited even before the computationally intensive pattern matching was performed. With the smaller texture, the NCP can be performed at refresh rates (the bare NCP operation includes image rectification). When performing the full match on the 128 pixel textures, the system slows down greatly. Given the bandwidth and image rectification overhead of the 256 pixel texture, the full pattern match has less of an impact.

It might seem like a coincidence that the value of 23 frames per second (fps) appears twice in the table above. This is due to the locking of the image display to the refresh cycle of the monitor. The value of 23 fps, with a monitor refresh rate of 70 Hz, is equal to 3 monitor refreshes and the value of 17 is 4 monitor refreshes.

7.2 Image Quality

The quality of the system on individual frames can be quite good. Most frames show good registration with little ghosting. One is even often able to discern the cracks between the subject’s teeth, see Figure 46. Where the system is weakest is in its stability. When viewing a video stream, sections of the image can become unregistered. This is especially true in the region of the nose, see Figure 47. This can be very distracting and is the major failure of the system. It is not clear that this instability negates the general applicability of the system. This dissertation proposes a general architecture for a view morphing system. There are aspects of the low level matching routines that are subject to variations and are, in general, open ended. A more sophisticated matching algorithm should solve the match instability problem.

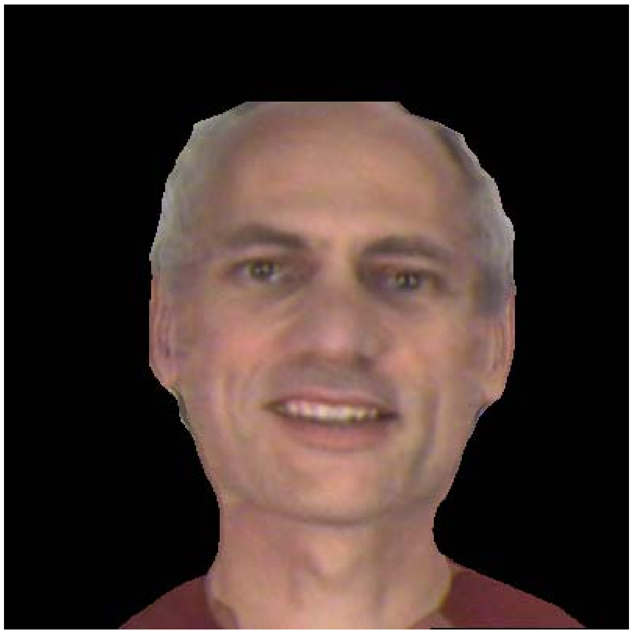


Figure 46 View morph with good registration. Note, cracks between the teeth are discernible.

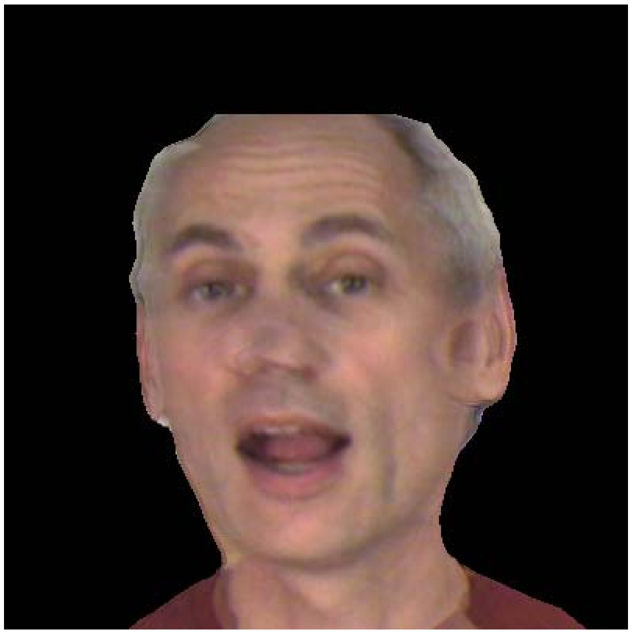


Figure 47 View morph with bad registration. Note the mangled nose and the ghosting of the ear.

7.3 Comparison with Other Systems

To understand the usefulness of the approach of this dissertation, a variety of systems were compared using various criteria. The systems considered were:

- **Single Video**: Single video stream no viewpoint manipulation (Sugawara et al., 1996)
- **Multiple Still**: Still images taken from various points of view (Insley, 1997)
- **Multiple Video**: Multiple video streams from unique viewpoints but no viewpoint manipulation (Sakamura et al., 1999)
- **Tracker Generic**: Video projection onto generic model with head tracker (Wang, 1998)
- **Tracker 3D Model**: Video projection onto previously constructed 3D model with head tracker (Rajan et al., 2002)
- **Model Spots**: Model based systems dynamically generating novel expressions from novel viewpoints using computer vision with tracking spots (Guenter et al., 1998)
- **Model No-Spots**: Model based systems dynamically generating novel expressions from novel viewpoints using computer vision without tracking spots (Pighin et al., 1998)
- **Depth from Stereo**: Depth from stereo, (Sandin et al., 2000), (Suh et al., 2002)
- **View Morph**: View morphing multiple video streams (Timm, 2003)

The following criteria were used to evaluate the systems: cost, accuracy, visual appearance, hardware needed, latency, subject independence, ease to integrate into a software system, 3D realism, equipment setup, pre-rendering subject involvement, range of viewpoint, and the ability to handle spontaneous novel input. A simple three value scale is used to rate the different systems: '+' is better than average, '-' is less than average, and '=' is about average. The term average is subjective and some mechanism is needed to anchor it to something. The *depth from stereo* approach was selected to define average. Anything that performs better than it gets a plus and anything that performs worse gets a minus. This still leaves a lot to subjectivity but is a starting point for evaluating the systems. The *depth from stereo* technique was chosen because it has a lot of similarities to the view morphing approach. It uses multiple cameras, does not require position-tracking devices, and relies heavily on computer vision.

TABLE III COMPARISON OF VIDEO AVATAR SYSTEMS

System	Cost	Fidelity	Hardware	Latency	Subject Indep.	Ease Setup	Ease Use	3D Real-ism	Pre-Requisites	Novel View-Point	Novel Input
Single Video	+	+	+	+	+	+	+	-	+	-	=
Multiple Still	=	+	=	+	-	-	+	=	-	=	-
Multiple Video	-	+	-	=	=	-	-	+	+	=	=
Tracker Generic	-	=	-	-	+	-	=	+	=	+	=
Tracker 3D Model	-	+	=	-	-	-	=	+	-	+	-
Model With Spots	=	+	=	=	-	-	-	+	-	+	-
Model No Spots	=	-	+	=	+	+	+	+	+	=	-
Depth from Stereo	=	=	=	=	=	=	=	=	=	=	=
View Morph	=	+	-	=	=	-	-	+	=	-	=

Viewing the table of the results of the evaluations, what can be said about which system is best? One observation is that there are many pros and cons to all the systems. The system that appears to perform the best at first glance is, ironically, “Single Video”, simply inserting a single video stream into a 3D scene. Single Video has the most “+” signs. It does not, however, produce good “3D Realism” or “Novel View Points”. For a system where 3D realism is the goal, it fails completely. To truly evaluate the systems, a scale factor was needed to rate the importance of the criteria. For example for one system, a cost of $10,000 might be a prohibitive factor; in another it might be of little consequence.

Image coding systems such as those described above have advantages over the system described in this dissertation in that they are able to transmit animated facial images using very low bandwidth (Eisert and Girod, 1998). They can be used for text to speech applications and for creating synthetic actors. Although systems exist for phoneme accurate lip-synching (Ezzat et al., 2002), expressing a range of emotion (Pighin et al., 1998), and the automatic imitation of pose and expression (Eisert and Girod, 1998), none of these systems fully possesses all these traits, especially in real-time.

Some of these systems automate some of the steps, but there is lot of time and effort to build the models. Once a model is built, it is associated with a specific individual. In addition, systems such as these will naturally only express a subset of the full range of human facial expression. The strengths of the view morphed based system presented in this dissertation are: full range of expression, subject independence, moderate cost of hardware, and real-time performance.

8. WHAT COULD BE DONE BETTER

While every effort was made to create a system that produces high quality results, that nature of this problem, and computer vision in general, is open-ended. It would be naive to think that this implementation was the final story on real-time view morphing. Along the way, sacrifices had to be made to software complexity to achieve real-time performance. Other limitations were due to the feasibility of a single person creating a complex software task in a reasonable amount of time. This section is to expose some of the areas where quality could be improved with either more software development or better hardware. It is presented now as a demonstration that the author was aware of these shortcomings and to serve as a starting point from which improvements could be made.

8.1 Account for lens distortion

All camera lenses produce distortion. Because the technique in this dissertation is based on matching similar regions of the scene, this distortion can impede the algorithm. Research in the domain of computer vision addresses this issue. There a variety of aspects of lens distortions that can be addressed such as chroma aberration in which different colors of light are mapped to slightly different areas of the image due to the refractive nature of the media. The most important issue that should be dealt with first is radial distortion. All lenses produce radial distortion to varying degrees. A fisheye lens is an extreme example producing a very curved appearance in the final image. Techniques exist to first measure the distortion and then to correct for it.

8.2 Perform gamma correction for matching and blending

A much often overlooked aspect of image data that is collected from video sources is that it is not linear in its intensity and the numbers used to represent those values. Instead of being a straight line, the intensity curve is typically bowed to the right. The existence of this curve is well known and is referred to as the gamma of the system. These gamma curves are used to characterize both the image input of the system and also how the intensity levels are displayed on output. An incorrect gamma level on output will cause the image to appear too dark or too bright and not be a good representation of the original image.

The linearity of the color information on input is the main concern. During the matching process, the color/intensity information from adjacent pixels is often combined to produce new pixel intensities. Implicit in these linear blending operations is that the sample intensities are represented linearly. In so far that the intensity is not represented linearly, image distortions are introduced. Jim Blinn has written excellent introductory articles on this topic (Blinn, 1989; 1998). Ideally, one would take the original image data, linearize it, and then perform all the image manipulation operations such as performing the NCP transform and creating the XMIP data structure.

The linearization of the image data would consist of two steps. First would be measure what the gamma curve of the system is. This rather difficult process could be skipped and a standard gamma curve assumed. The second step would consist of mapping the gamma sample values to linear values before any image manipulation takes place. Because the number of values per color channel is limited to 256, this conversion could be done with the use of the lookup table. The mapping of 8 bit gamma color to 8 bit linear color causes some loss of the original color information. Jim Blinn (Blinn, 1998) says that 14 bits of information is needed to represent the color information without loss. In the matching process, 32 bit floating-point numbers were used to represent the gray scale image. Therefore, the per channel color lookup tables would be used to map the 8 bit colors into 32 bit floating point linear color values before computing the linear gray scale values.

After the matching process has completed, the colors of the two original images are blended together to create the virtual camera image. In terms of program flow, the image manipulation operations performed during the matching process are separate from those of the display process. This independence is due to the output of the matching process being only texture coordinates and vertices and not any pixel information. In the same manner as the image sample data is falsely assumed to be linear during the generation of intermediate images during the matching process, the intensities of the image samples that are blended on output are encoded in a gamma curved manner. This false assumption causes distortions in the final image and needs to be accounted for in the same manner as the input images.

8.3 Synchronize The Camera Image Frames

The first pair of cameras used was just conventional camcorders. Commodity video equipment like this does not have any mechanism to synchronize the timing of the image capture. It was naively thought initially that given the sampling rate, not enough motion could occur between frames to be of significance. Viewing individual frames, it was seen that this was not the case. The difference of motion was especially seen with the lips while speaking.

These original cameras were later replaced with cameras that could be synchronized. While these new cameras did not have an explicit master-slave connection to synch one with the other, they did have ability to individually be synchronized with the same line/AC source. The result of the synchronization was hard to characterize as there are other important improvements as a result of replacing the camera. With the new cameras, the fidelity of the CCD’s were higher, better and brighter lenses were used, and at the same time, a different video capture system was put into place. Previously the video streams were combined into a low quality multiplexer and then the combined video was sent to the computer video capture card. With the new cameras, the video output of each camera was attached directly to its own video capture port on the computer. The most important benefit of using the new synchronized cameras is that it ruled out the problem of synchronization as a factor in the matching of images.

8.4 Use Digital Image Input

The idea of using digital image input was toyed with to remove some of the distortion due to the analogue capture process. The output of conventional digital video sources is the mini-DV format. This is a compressed format that produces about 4 mega-bytes of data per second instead of the usual 30 mega-bytes that the raw data does. The color separation is excellent with the DV format compared to analogue video due to the way the three color signals are encoded into one with the composite connection. The inadequacies of a composite connection can be reduced with a S-video connection that transmits the video signal in two parts, a luminance component and a chroma component containing two sub-signals. A component system transmits the three red-green-blue signals on separates wires is the best for chroma separation.

The digital signal option was rejected for a variety of reasons. The composite inputs provided enough color detail to satisfy the needs of the system given the excessive image distortion that the view morphing process produces. The compressed DV format would need significant computational power to decode although special video card could perform this operation. The resulting decoded images would be of higher resolution than would be needed by the system and hence would need to be sub-sampled. Finally, the cost of analogue equipment is enough cheaper than digital so as to sway the decision.

8.5 User Hardware Real-Time Chroma-Keying

The implemented system did not perform chroma-keying in real-time. All images were first captured and then processed. To better demonstrate the real-time capabilities of the system, a live video stream should be processed. The chroma-keying process is very computationally intensive. Hardware exists that will perform high quality chroma-keying in real-time without loading the main CPU(s) of the system. While this might be a nice addition to the system, it was not chosen for two reasons. First it would have added extra cost to the system. Second it would have tied us to a specific hardware platform.

8.6 Use a Different Image Segmentation System

A variety of image segmentation schemes exist. A chroma-keying scheme was used but even within chroma-keying there are a variety of approaches. First there is the choice of which color space to use. The YCrCb color space was chosen but the HSV color space may have worked better.

The chroma-key system depends completely on having a mono-chromatic background. A backdrop like this would not be practical in many environments such as a business office for example. What would be more discrete and not demand special equipment would be use digital differencing of the background. If the background is such that it does not change, then it can be captured and the foreground could be quickly segmented out, perhaps with less computation and more effectiveness than the chroma-keying.

8.7 Render the Image Using Lighting

To give the subject more realism, lighting might be added. On some level, this is an ill-posed request. This enhancement does not make sense because lighting is based off of 3D normals. The system that was implemented did not make use of nor generate any 3D information. However, it might be possible to provide an implied 3D model based off of the NCP assumption. The value of this problem was not high; therefore addressing it was not attempted.

8.8 Dealing With Specular Highlights

A problem that is the opposite of that posed in section 8.7 is that of how to remove lighting effects. One problem that hinders the performance of many vision systems is that of specular highlights. The issue of specular lighting effect is hard to deal with because the location of the highlights depends on the viewpoint of the observer. With the use of cameras pointing at the subject from two different angles, the highlights will be located at different locations on the subject. With the simple SSD matching algorithm used in this dissertation, the intensity of the highlights generate large but incorrect values that cause misalignments.

It is beyond the scope of this dissertation to try making any attempt to remove the specular highlights after the fact. Effort can be fruitfully deployed at image capture time to reduce specular lighting effects. One technique is to provide diffuse lighting of the scene. Florescent lights provide nice soft lighting. Another approach is to make the subject less shiny. A traditional method to do this is to powder the face of the subject.

9. FUTURE WORK

Several important areas to be further investigated are:

- More effective matching algorithms.
- Ability to support dynamic backgrounds.
- Due to the redundancy of image information from the multiple video streams, a compression method for sending the data to a remote site could be devised.
- Because the algorithm does not depend on human heads as the object of interest, the robustness with other subject such as animals or the rotation of other objects may be of interest.
- View morphing with two degrees of freedom using three or more non-collinear cameras.

A couple of implementation areas to be investigated are:

- The effect of different lighting on the effectiveness of the matching algorithms.
- The use of higher resolution images such as from two separate digital cameras.
- Explorations of various mesh sizes and aspect ratios.

9.1 Compression

Video conferencing is very bandwidth intensive. Remote viewing of a single video stream across a network stresses most systems. The work in this dissertation augments the single video stream of a typical video conferencing system with two video streams and two geometric streams (at lower resolution). It is quickly observed that the video streams have redundant information that can be compressed out. In the region of overlap between the two images, only subtle differences due to different points of view and minor occlusion should occur. It would be more efficient to encode these differences than transmit the second overlapping region.

An alternative approach would be to render the final image locally and only transmit a single video stream. For this approach to work, knowledge of what is the required view would need to be transmitted from the remote site.

9.2 Intelligent Hole Filling

This dissertation suggests a simple method for filling the holes in an image due to occlusion (see section 4.8). When one has at least one unoccluded view by a camera, that single camera's image should be used in that region. This would require first an enhancement to the vision stage of the process to accurately detect the occlusions. This detection is not an easily solvable task due to ambiguities when correlating the images. The actual hole filling should be a straightforward enhancement to the rendering stage.

9.3 Image Morphing with Two Degrees of Freedom

The system presented in this dissertation describes a virtual camera system that is able to move along a one-dimensional line between two video images. By using a system with three or four cameras arranged into a triangle or square, a system that would allow for the movement of the virtual camera in two dimensions could be made. With a four-camera system, one could first create two view morphs, one for each pair of video images and then view morph together these generated images. This would allow for the efficiency of scan-line algorithms.

9.4 Pattern Matching in Time

It has been observed, that selecting a single image and rotating it produces view morphs that look better than the streaming video. When streaming the video, the inaccuracies of the simple pattern matching system implemented are exposed when sections of image jump drastically between frames. Using an inter-frame matching process could provide a smoothing control mechanism to limit the transitions between frames. Another big advantage of performing inter-frame matching is that it could make use of information from the previous frame to predict the match of the current frame. This would make better use of resources by limiting the region of search.

10. CONCLUSION

A system for creating a video avatar based on view morphing was implemented. Running on commodity PC hardware, using input from a pair of video cameras, and video capture card, a system was produced that performs in real-time (section 7). To implement the system, novel data structures and algorithms were devised. The Normalized Cylindrical Project (NCP) operation was fundamental to transforming the image correlation problem from one that was extremely difficult to one that was solvable. The RIP-map data structure provided a means, which was efficient in both space and time, to render the distorted images while managing the issues of aliasing and blurring. Finally, the fast scan-line rendering routines are at the heart of what makes this system realizable in real-time on commodity hardware. The current system exhibits some instability of the image correlation. This issue is independent of the overall architecture solution presented and more research into the matching algorithms and more computational power would alleviate these deficiencies. In conclusion, this dissertation demonstrates the feasibility of using view morphing as the basis of a video avatar system.

CITED LITERATURE

Baker, H., Tanguay, D., Sobel, I., Gelb, D., Goss, M., Culbertson, W., and Malzbender, T.: The Coliseum Immersive Teleconferencing System, International Workshop on Immersive Telepresence, Juan Les Pins, France, December 6, 2002.

Beier, T. and Neely, S.: Feature-Based Image Metamorphosis, Computer Graphics (Proc. SIGGRAPH 92), Vol. 26, pp. 35-42, July 1992.

Beymer, D., Shashua, A., and Paggio, T.: Example based image analysis and synthesis. Technical Report 1431, MIT AI Lab, 1993.

Blinn, J.: A Ghost in a Snowstorm. IEEE Computer Graphics, January/February 1998.

Blinn, J.: Dirty Pixels. IEEE Computer Graphics, July 1989.

Blinn, J. and Newell, M.: Texture and Reflection in Computer Generated Images. Communications of the ACM, vol. 19, no. 10, pp. 542-547, October 1976.

Bregler, C., Covell, M., and Slaney, M.: Video Rewrite: Driving Visual Speech with Audio. Computer Graphics Proceedings, SIGGRAPH 1997, pp. 353-360, 1997.

Catmull, E.: A Subdivision Algorithm for Computer Display of Curved Surfaces. Ph. D. thesis, Department of Computer Science, University of Utah, Tech. Report UTEC-CSc-74-133, December 1974.

Chen, S., and Williams, L.: View Interpolation For Image Synthesis. Computer Graphics (SIGGRAPH Proceedings), pp 279-288, 1993.

Chen, S.: QuickTime VR – An Image-Based Approach to Virtual Environment Navigation", Computer Graphics Proceedings, SIGGRAPH, pp. 29-38, 1995.

Crow, F.: Summed-Area Tables for Texture Mapping. Computer Graphics (Proc. SIGGRAPH 84), Vol. 18, No. 3, pp. 207-212, July 1984.

Cruz-Neira, C., Sandin, D., and DeFanti, T.: Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. Proceedings of SIGGRAPH'93, pp.135-142, 1993.

Cruz-Neira, C. Sandin, D., DeFanti, T., Kenyon, R. and Hart, J.: The CAVE: Audio Visual Experience Automatic Virtual Environment", Communications of the ACM, Vol. 35, No. 6, pp. 65-72, June 1992.

Debevec, P.: Rendering Synthetic Objects Into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. Computer Graphics Proc., SIGGRAPH, pp. 189-198, 1998.

Eisert, P. and Girod, B.: Analyzing Facial Expressions for Virtual Conferencing. IEEE Computer Graphics and Applications, pp. 70-78, September/October 1998.

Ekman, P. and W. Frisen, W.: Facial action coding system. Palo Alto, CA: Consulting Psychologists, 1977.

Ezzat, T., Geiger, G., and Poggio, T.: Trainable Videorealistic Speech Animation. Computer Graphics Proceedings, SIGGRAPH 2002, pp. 388-398, 2002.

Ezzat, T. Poggio, T.: Facial Analysis and Synthesis Using Image-Based Models. Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, Killington, Vermont, October 1996.

Faugeras, O.: Three-Dimensional Computer Vision: A Geometric Viewpoint. MIT Press, Cambridge, Massachusetts, 1993.

Foley, J., Van Dam, A., Feiner, S., and Hughes, J.: Computer Graphics Principles and Practice. Addison-Wesley Publishing Company, Reading, Massachusetts, 1991.

Gortler, S., Grzeszczuk, R., Szeliski, R., and Cohen, M.: The Lumigraph. Computer Graphics Proceedings, SIGGRAPH, pp. 43-54, 1996.

Gose, E. and Johnsonbaugh, R.: Pattern Recognition and Image Processing, to be published, circa 1995.

Guenter, B., Grimm, C., Wood, D., Malvar, H., and Pighin, F.: Making Faces. Computer Graphics Proceedings, SIGGRAPH, pp 55-66, 1998.

Havaldar, P., Lee, M., and Medioni, G.: View Synthesis from Unregistered 2-D Images. Proc. Graphics' Interface 96, pp. 61–69, 1996.

Heckbert, P. S.: Survey of Texture Mapping. IEEE Computer Graphics & Applications, 6, 11, pp 56-67, November 1986.

Hirose, M., Ogi, T., Yamada, T., and Tamagawa, K.: Development of Stereo Video Avatar in Networked Immersive Projection Environment. ICIP1999 21-25, 1999.

Hodges, M. and Sasnett, R.: Multimedia Computing – Case Studies from MIT Project Athena. Addison-Wesley, 1993, pp. 89-102.

Insley, J. "Using Video to Create Avatars in Virtual Reality" Visual Proceedings of the 1997 SIGGRAPH Conference, Los Angeles, CA, pp. 128, 1997.

Ishii, H. Kohayashi,M., and Arita, K.: Iterative Design of Seamless Collaboration Media. Communications of the ACM, August 1994.

Larson, R. D., and Shah, M.S.: Method for Generating Addresses to Textured Graphics Primitives Stored in RIP Maps. US Patent 05222205, 1993.

Laveau , S., Faugeras, O.: "RR-2205 - 3-D Scene Representation as a Collection of Images and Fundamental Matrices, Rapport de Recherché INRIA, No. 2205, February, 1994.

Lee, S., Badler, J,. and Badler, N.: Eyes Alive. Computer Graphics Proceedings, SIGGRAPH 2002, pp. 637-644, 2002.

Lee, S., Wolberg, G., and Shin, S. Y.: Polymorph: Morphing Among Multiple Images. IEEE Computer Graphics & Applications, pp. 58-71, January 1998.

Lengyel, J.: The Convergence of Graphics and Vision. IEEE Computer, pp. 46-53, July 1998.

Leung, W.H., Tseng, B.L., Shae, Z., Hendriks, F., and Chen, T.: Realistic Video Avatar. IEEE Intl. Conf. on Multimedia and Expo., New York, July 2000.

Li, H., Roivainen, P., and Forchheimer, R.: 3-D Motion Estimation in Model-Based Facial Image Coding. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 6, pp. 545-555, June 1993.

Lippman, A.: Movie-Maps: An Application of the Optical Videodisc to Computer Graphics. Proc. SIGGRAPH '80 pp. 32-43, 1980.

Mallat, S.: Wavelets for a Vision", Proceedings of the IEEE, Vol. 84, No. 4, April 1996.

Matusik, W., Buehler, C., Raskar, R., Gortler, S., and McMillan, L.: Image-Based Visual Hulls. Computer Graphics Proceedings, SIGGRAPH 2000, pp. 369-374, 2000.

Mark, W., McMillan, L., and Bishop, G.: Post-Rendering 3D Warping. Symposium on Interactive 3D Graphics, pp. 7-16, 1997.

Mccormack, J., Farkas, K., and Jouppi, N.: Feline: Fast Elliptical Lines for Anisotropic Texture Mapping. Computer Graphics Proceedings, SIGGRAPH 1999, pp. 243-250, 1999.

McMillan, L. and Bishop, G.: Plenoptic Modeling: An Image-Based Rendering System", Computer Graphics Proceedings, SIGGRAPH, pp. 39-46, 1995.

Meijering, E., Zuiderveld, K., Niessen, W., and Viergever, M.: A Fast Image Registration Technique for Motion Artifact Reduction in DSA. IEEE International Conference on Image Processing - ICIP pp. 435-439 1999.

Miller, G. et al.: The Virtual Museum: Interactive 3D Navigation of a Multi-Media Database. The Journal of Visualization and Computer Animation, Vol. 3, No. 3, pp. 183-198, 1992.

Moezzi, S., Katkere, A., Kuramura, D., and Jain, R.: Reality Modeling and Visualization from Multiple Video Sequences. IEEE Computer Graphics and Applications, pp. 58-63, November 1996.

Ogi, T., Yamada, T., Tamagawa, K., and Hirose, M.: Video Avatar Communication in a Networked Virtual Environment. INET 2000, The 10th Annual Internet Society Conference, Yokohama, Japan 18-21 July 2000.

Ogi, T. Yamada, T., Tamagawa, K., Kano, M., Hirose, M.: Immersive Telecommunication Using Stereo Video Avatar. Virtual Reality Conference Proceedings (VR'01), p 45, Yokohama, Japan, March 13 - 17, 2001.

Ohta, Y. and Kanade, T.: Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 2, pp. 139-154, March 1985.

Oppenheim, A.: Discrete-Time Signal Processing. Prentice Hall, Englewood Cliffs, New Jersey, 1989.

Pentland, A., and Choudhury, T.: Face Recognition for Smart Environments. Computer , Vol. 33 Issue 2 , pp. 50-55, Feb. 2000.

Pighin, F., Hecker, J., Lischinski, D., Szeliski, R., and Salesin, D.: Synthesizing Realistic Facial Expressions From Photographs. Computer Graphics Proceedings, SIGGRAPH 1998, pp. 67-74, 1998.

Rademacher, P. and Bishop, G.: Multiple-Center-of Projection Images. Computer Graphics Proc., SIGGRAPH, pp. 199-106, 1998.

Rajan, V.: View-Dependent Texture Mapping of Video for Realistic Avatars in Collaborative Virtual Environments. Sketches and Applications, SIGGRAPH 2001.

Rajan, V., Subramanian, S., Keenan, D., Johnson, A., Sandin, D., DeFanti, T.: A Realistic Video Avatar System for Networked Virtual Environments. 7th annual Immersive Projection Technology (IPT) Symposium, Orlando, Florida, 2002.

Rydfalk, M.: CANDIDE: A Parameterized face. Department of Electrical Engineering Rep. LiTH-ISY-I-0866, Linkoping University, Sweden, October 1987.

Sandin, D., Hill, A., Plepys, D, Rajan, V., and Subramanian, S.: 3D Avatars Using Depth from Stereo. Electronic Visualization Laboratory (EVL), Director Defanti, T., University of Illinois at Chicago. 2000.

Seitz, S. and Dyer, C.: Physically-valid view synthesis by image interpolation. Proceedings IEEE Workshop on the Representation of Visual Scenes, pp 18-25, Cambridge, MA, June 1995.

Seitz, S. and Dyer, C.: View Morphing. Proceedings of SIGGRAPH, 21-30, 1996.

Shade, J., Gortler, S., He, L., and Szeliski, R.: Layered Depth Images. Computer Graphics Proc., SIGGRAPH, pp. 231-242, 1998.

SNHC: "SNHC Systems Verification Model 4.0", MPEG4, ISO/IEC JTC1/SC29/WG11 N1666, Bristol, Great Britain, April, 1997.

Subramanian, S., Rajan, V., Keenan, D., Sandin, D., DeFanti, T., Johnson, A: A Realistic Video Avatar System for Networked Virtual Environments. VR2002 – Immersive Projection Technology 2002 Symposium March 25, 2002.

Sugawara, S., Matsurra, N., Kato, Y., Sasaki, K., Imai, M., Yamana, T., Kiyosue, Y., and Shimamura, K.: Inter Space Project - Cyber Campus. CSCW96, Video Program, No. 3, Nov. 1996.

Suh, Y., Hong, D., and Woo, W.: 2.5D Video Avatar Augmentation for VRPhoto. ICAT, Tokyo, Japan, December 4-6, 2002.

Szeliski, R.: Video Mosaics for Virtual Environments. IEEE Computer Graphics and Applications, 22-30, March, 1996.

Szeliski, R.: Image Mosaicing for Tele-Reality Applications. 0-8186-6410-X/94 $4.00 1994 IEEE.

Takeuchi, A. and Nagao, K.: Communicative Facial Displays as a New Conversational Modality. Proc. INTERCHI '93, 187-193, April 1993.

Tokuda, J., Morikawa, S., Dohi, T., and Hata, N.: Ultra-fast Image Registration Embedded in Intraoperative MR Imaging. 16th International Congress and Exhibition of Computer Assisted Radiology and Surgery, 2002.

Torborg, J. and Kajiya, J.: Taliman: Commodity Realtime 3D for the PC. Computer Graphics Proceedings, SIGGRAPH, pp 353-364, 1996.

Terzopoulos, D. and Waters, K.: Analysis and Synthesis of Facial Image Sequences Using Physical and Anatomical Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 6, 569-577, June 1993.

Timm, K.: Real-Time View Morphing of Video Streams. Sketches & Applications 2003 SIGGRAPH Conference, San Diego, CA, 2003.

Trucco, E. and Verri, A.,: Intoductory Techniques for 3-D Computer Vision. Prentice Hall, Upper Saddle River, New Jersey, 1998.

Ullman, S. and Basri, R.: Recognition by Linear Combinations of Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 10, pp. 992-1006, October 1991.

Wang, F.: Video Avatars in Collabrative Virtual Environment. http://www.evl.uic.edu/fwang/video.html, 1999.

Werner, T., Hersch, R., and Hlavac, V.: Rendering real-world objects using view imterpolation. Proc. of International Conference On Computer Vision, pp 957-962, June 1995.

Williams, L.: Pyramidal Parametrics. Computer Graphics (Proc. SIGGRAPH 83), Vol. 17, No. 3, pp. 1-11, July 1983.

Wolberg, G.: Digital Image Warping. IEEE Computer Society Press, Los Alamitos, California, 1990.

Woo, M, Neider, J., and Davis, T.: OpenGL Programming Guide. Addison Wesley Developers Press. Reading, Massachusetts, 1997.

Xiao, J., Rao, C., and Shah, M.: View Interpolation of Dynamic Scenes. Eurographics 2002.

Yura, S., Usaka, T. and Sakamura, K.: Video Avatar: Embedded Video for Collaborative Virtual Environment**.** IEEE International Conference on Multimedia Computing and Systems Volume II-Volume 2, p. 433-438, Florence, Italy**,** June 07 - 11, 1999**.**

Yura, S., Usaka, T., and Sakamura, K.: Design and Implementation of the Browser for the Multimedia Multi-User Dungeon of the Digital Museum. Transactions of Information Processing Society of Japan, Vol. 40, No. 2, IPSJ 1999.

VITA

NAME:	Karl Walter Timm
EDUCATION:	B.A., Physics, Saint Olaf College, Northfield, Minnesota, 1978.
	M. S., Computer Science, University of Wisconsin Milwaukee, Milwaukee, Wisconsin, 1987.
	Ph.D., Computer Science, University of Illinois at Chicago, Chicago, Illinois, 2003.
TEACHING:	Department of Computer Science, University of Wisconsin Milwaukee; Assembly Language, 1985.
	Department of Computer Science, University of Illinois at Chicago; Introduction to Computing, 1994.
HONORS:	Sigma Pi Sigma – Honorary Physics Student Society, Saint Olaf College.
	Phi Kappa Phi – Honorary Society, University of Illinois at Chicago.
PROFESSIONSAL MEMBERSHIP:	Computer Society - ACM Computer Society – IEEE
ABSTRACTS:	Timm, K.: Real-Time View Morphing of Video Streams. <u>Sketches & Applications 2003 SIGGRAPH Conference</u>, San Diego, CA, 2003.
SERVICE:	United Status Peace Corps, Malaysia, 1978 – 1981.