Three-Dimensional Representations of Video using Knowledge Based Estimation

by

Henry Neil Holtzman

B.S., Computer Science and Engineering
Massachusetts Institute of Technology, 1986

Submitted to the Media Arts and Sciences Section,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1991

Signature of Author .....
Media Arts and Sciences Section,
School of Architecture and Planning
August 9, 1991

Certified by .....
Andrew Lippman
Associate Director, MIT Media Laboratory
Thesis Supervisor

Accepted by .....
Stephen A. Benton
Chairman, Departmental Committee on Graduate Students

Three-Dimensional Representations of Video using Knowledge Based Estimation

by

Henry Neil Holtzman

Submitted to the Media Arts and Sciences Section,
School of Architecture and Planning
on August 9, 1991, in partial fulfillment of the
requirements for the degree of
Master of Science
at the
Massachusetts Institute of Technology

Abstract

Computer animation tools allow complete control over their virtual worlds. Animators can make changes to their models and re-render their footage over and over, until it looks just right. Traditional filmmakers do not enjoy such freedom. Once film is shot, little can be done to make changes in the contents of the footage. The composition of the frames has been locked in by the camera's two dimensional perspective transformation.

In this thesis, we combine computer animated worlds with real movie footage, building a graphic-photographic model. Given movie footage we'd like to alter, a three dimensional model of the scene is assembled, by hand, using a-priori knowledge about the world. The 3-D structure of the model is refined automatically using the footage, to improve its correspondence with the original set. The model is augmented with the footage, making photo-realistic renditions of the model possible. Next, the model is altered and rendered to create footage which is new and unique. Colorization is used as an in depth example of how the scene may be altered. Other example alterations are demonstrated as well. Problems with the modeling technique are discussed, and workarounds suggested.

Thesis Supervisor: Andrew Lippman
Title: Associate Director, MIT Media Laboratory

The work reported herein was supported by a contract from the Movies of the Future consortium, including Apple Computer, Inc., AT&T, Eastman Kodak Co., Viacom International Inc., and Warner Brothers Inc.

Contents

List of Figures

Chapter 1

Introduction

1.1 Movie Modeling

Modeling tools permit the user to create, view, and alter simulated three dimensional environments. The resulting renditions range from sketch to cartoon like appearance. Such tools are commonly used by engineers, designers, architects, and animators. The ability to alter the contents of computerized models make them an enticing tool for processing movie footage. *Movie modeling* is the process by which film or video footage is converted into a three dimensional representation, or model, which can be used to create altered footage.

One example of altering footage is colorization. The black and white footage is converted into a movie model, the modeler chooses colors for objects within the scene, and new footage is created. Given a sufficiently flexible model, a modeled movie could also be rendered with the camera moved, the lighting changed or using a different lens. Pieces of the model can be moved, removed, or used along with parts of another model to create a set which never really existed.

1.2 Project Overview

Three dimensional models of real sets can be built using cameras which record depth as well as light values [7]. Range cameras are not yet used in practice. Even if they become popular, there is already a vast library of footage on film and videotape. This thesis will address the

problem of creating a model from footage which has no associated depth information.

We will progress from movie footage to a three dimensional model of the contents of the footage, back to footage again. Along the way changes will be made to the contents of the footage. Initially, the model is created by hand using spatial reasoning skills and common sense knowledge. A rough guess at the size, shape and placement of the contents of the scene is made by the modeler. The model is then brought into visual coincidence with a frame from the footage. Next, the model is automatically refined using processed imagery from the movie footage. The model can be used to direct modification of the footage, or combined with the footage, altered, and rendered to new footage.

A simple, yet realistic example, is used throughout this work: a scene filmed in the kitchen set of the "I Love Lucy" show. The test case alteration was to colorized the scene, and we will dwell on optimizations of the work suited to that process. The model will be evaluated, its flaws discussed and improvements suggested. Actors and other moving objects do not fit into our particular model framework, so isolation of the actors will be discussed as well. Examples of other alterations, such as moving the camera, will be provided.

1.3 Related Topics in Joint 2-D/3-D Processing

1.3.1 Model Based Coding

John Watlington used model based coding to create photorealistic animations in real time. His "Synthetic Movies [39]" composited 2-D video sequences in a three dimensional space to give a visual readout of the activity of a computer system. As users logged into the system, they would be shown entering a room. The activity they pursued while in the animated room would depend on what they were doing on the computer. When idle, a user might be shown to lean back in their chair and take a nap. When a user signed off the system, the animation software displayed a sequence in which the user stood up and left the room.

Movie modeling can be used to decrease the bandwidth necessary to transmit a scene. Rather than transmit a large quantity of data which remains essentially the same from frame to frame, the model can be transmitted once. Thereafter, only information about motion of the camera and contents of the model are sent. Patrick Mclean [22] investigated model based

coding using a two dimensional model of the set.

Research in 3-D model based coding has primarily been in the field of videotelephony [1, 41]. A generic model of the human face is constructed for use in the receiver. Some researchers use a fixed model for all faces, others use a model with control points which allow it to conform to contours of individual faces. In either case, a picture of an individual's face is texture mapped onto the model. The texture map is sent only once. As the subject's head moves, the model is moved. Rotations of up to 30 degrees are typically possible without the mapping breaking down. Eye motion is handled by sending a small texture patch for each eye at a regular interval. Mouth movement is handled similarly to eye movement. Using a set of models which has the mouth and cheeks in various expressions, in conjunction with sending texture patches for the mouth, yields a better result.

This sort of model based videotelephony has roots in various projects at the MIT Media Laboratory, and its predecessor, The Architecture Machine Group. Susan Brennan explored constructing facial expressions by altering selected parts of a facial image [8]. Peggy Weil synthesized facial images out of componentry, synthesizing a composite facial image from the user's description [40]. This computergraphic system could be used to construct a face from memory to aid police in their investigations. Clea Waite created a 3-D computergraphic facial model, with controls for varying expression [38]. The *Talking Heads* project [25, 6] increased the speaker's presence in videotelephony by using a physical model of the speaker's head. A translucent mask of the speaker's face was created from a solid model of the head. The mask was placed on a motorized mount which could tilt and swivel to match the motions of the speaker. A video image of the speaker's face was projected into the mask.

1.3.2 Range Cameras and Depth Inferencing

Cameras have been constructed that measure depth and intensity at each point. These experimental cameras produce a three-dimensional representation of the area they photograph. Stereo cameras operate by finding points of correspondence between their two elements [15]. The more offset the points are from each other, the closer they are to the camera. Laser rangefinders operate either by travel time, or by displacement of a stripe of laser light [7]. The laser light is projected into the scene from an angle. The greater depth it achieves at each point, the farther

point, the farther it will reach towards the opposite side of the scene before reflecting towards the camera. The stripe is swept across the entire image. Depth-from-focus rangefinders operate by measuring the blur of each pixel [7, 31]. The blurrier the pixel, the further away from the focal point of the lens.

Depth from focus does not require a special camera. The analysis can be performed on any image, although purposely focusing before the closest object in the scene helps [7]. Motion of a single camera can be used as a substitute for two cameras in the stereo camera analysis. Shape from motion also uses the same basic procedure as depth from stereo, allowing recovery of the shape of a moving object. Shape from shading techniques look at the how the light reflects off a surface to approximate its shape [14, 32]. Textures can be utilized in determining the shape of an object as well. If the texture is uniform then how its 2-D projection bunches up and spreads out can be used to recover the shape of the object. Symmetry-seeking models, which attempt to minimize intrinsic forces when faced with image constraints as extrinsic forces, tend to form into plausibly shaped 3-D objects [36].

Chapter 2

The Movie

2.1 Gathering The Data

The movie footage we are modeling starts out on film, is scanned to video tape, and then digitized as an array of light intensity values. During the film to video conversion the frame rate is changed from 24 fps (frames per second) to 30 frames per second. The frame rate conversion, known as 3-2 pulldown, is accomplished by alternately recording 3 fields of video from one frame of film, followed by 2 fields of video from the next film frame. After digitizing the footage the 3-2 pulldown is undone, yielding 24 progressively scanned data sets of movie frames for every second of video (see figure 2-1). If the original footage is shot using a video camera, skipping over the film step, then the video should be de-interlaced using some other scheme.

Many video digitizers produce pixels which are not arranged in an isotropic lattice. A height to width ratio of 6 to 5 is common. It is rare for a modeling tool to assume anything other than a square array. Since the digitized footage is going to be combined with the output of the modeler, the digitized frames should be resampled so that the pixels fall on a square lattice. The conversion can be achieved by horizontally interpolating each line by the numerator of the height:width ratio, followed by a horizontal decimation by the denominator. To save memory and reduce the computational cost, a position varying two tap averaging filter can be used instead. The result will be slightly blurred, but not noticeably. Figure 2-2 illustrates how to do the resampling in the case of 5 pixels being converted to 6.

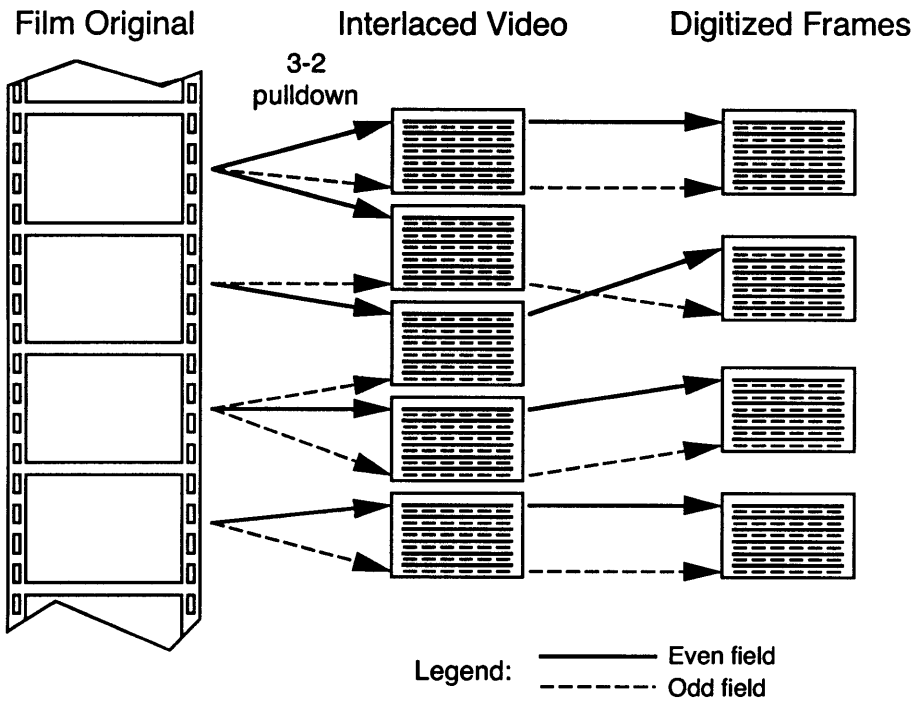


Figure 2-1: Scan conversion processes.

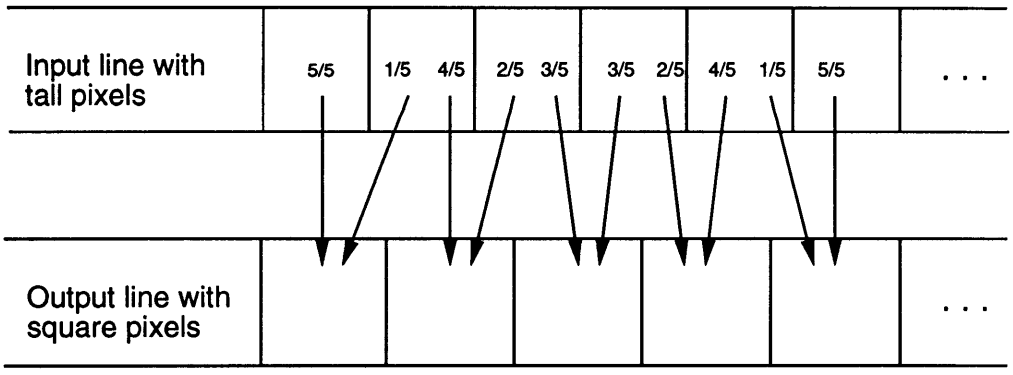


Figure 2-2: Computationally inexpensive method of creating 5 pixels from 6.

2.2 Processing the data

2.2.1 Edge Reduction

The data sets typically contain 640×512 elements. When these elements are displayed in proper order on a computer screen each data set will recreate the original movie frame. That is a large amount of data. Reducing the data in the image in an appropriate fashion will aid reasoning about the contents of the picture. A useful data reduction operator is an edge detector. After a frame has been processed by an edge operator, it is reduced to a bi-valued image, which contains white wherever there is an edge and black elsewhere. An edge image, such as shown in figure 2-4, looks like a sketch of the original, shown in figure 2-3.

2.2.2 The Laplacian Operator

In [21] Marr discusses processing an image to reduce it to edges. An edge is a sharp change in illumination. The first derivative will be a peak, and the second derivative a pair of oppositely signed peaks. The edge is located at the zero crossing between the peaks. Marr's filter for producing the second derivative is $\nabla^2 G$ where ∇^2 is the Laplacian operator $(\partial^2/\partial x^2 + \partial^2/\partial y^2)$ and G is a two dimensional Gaussian distribution:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\pi\sigma^2}} \tag{2.1}$$

σ is the standard deviation of the Gaussian function. The image is blurred by the application of the $\nabla^2 G$ filter. Increasing σ will increase the blur. The scale at which edges are detected can be controlled by the value of σ. Small values of σ will result in sharp texture in the image giving rise to zero crossing, large values of σ will blur the image sufficiently that only the largest image features will give rise to edges.

2.2.3 Non-linear Laplacian filter

Vliet, Young, and Beckers discuss using a nonlinear Laplace Operator in [37]. The advantage of the nonlinear Laplace filter, **NLLF**, is its simplicity. Marr's $\nabla^2 G$ operator requires significantly more computation then a **NLLF**.

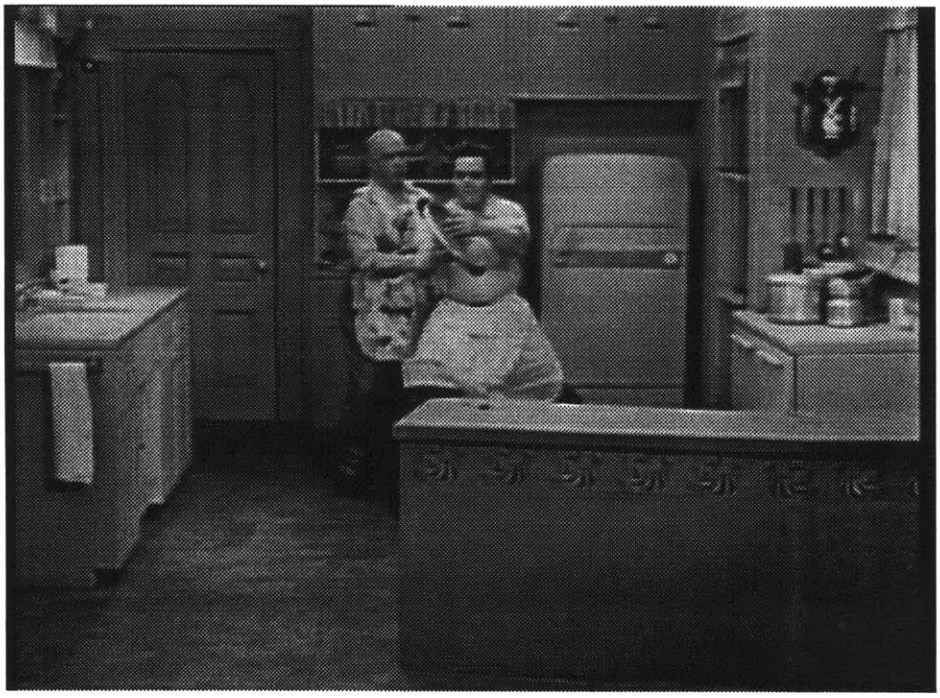


Figure 2-3: A digitized frame from the Ricardo kitchen

Figure 2-4: Edge reduction of the Ricardo kitchen

$$\mathbf{NLLF}(I_{ij}) = max - 2 * I_{ij} + min \tag{2.2}$$

where min and max is the minimum and maximum intensities found in a region around I_{ij}. The size of the search neighborhood affects the operator in an analogous fashion to how σ affected the $\nabla^2 G$ operator. When a small region is searched, minute textures will give rise to edges. As the search region is increased, smaller textures go undetected and the larger image features predominate. Noise spikes might affect the values of max and min, so prefiltering the image is desirable. Either a small box filter or Gaussian filter should be passed over the image before the **NLLF** is applied. The noise peaks, as well as image peaks, will be reduced.

2.2.4 Zero Crossing Detector

The output of either a **NLLF** or $\nabla^2 G$ operator encodes edges as zero crossings. A zero width crossing is easy to detect. Any place a positive pixel is adjacent to a negative pixel is a zero crossing. The output of the filter might contain zeros, however. To properly assign the location of an edge, these zeros must be removed. Wherever a zero is encountered, the distances to the nearest positive and negative pixels are calculated. The zero pixel is reassigned to be positive or negative according to the shorter distance: positive versus negative.

Two masks are formed from the zero removed **NLLF** output. The first is $TRUE$ wherever a pixel is surrounded by eight other positive pixels, and $FALSE$ elsewhere. The second mask is $FALSE$ wherever the four non-diagonal neighbors to a pixel are $FALSE$ and $TRUE$ elsewhere. The two masks are combined with an XOR operator. The result will contain a $TRUE$ contour along the zero crossing.

2.2.5 Edge Strength Measure

Images contain strong and weak edges. Weak edges might result from shadows and other illumination variations, noise, or surface texture. Only strong edges correspond to the boundaries of objects. Gradient operators and grey level contrast operators are suitable for measuring edge strength. The *min*imum and *max*imum values found within a region are also suitable for

making an edge strength measure [37].

$$I_{Edge-Strength}(i,j) = \mathbf{min}\{max, -min\} \tag{2.3}$$

The final edge image is formed by thresholding a point by point multiplication of the edge contour image by the edge strength image. The value of the threshold will control the amount of edge detail. Figure 2-5 illustrates a raw edge image, which was processed with an edge strength measurement to form figure 2-4.

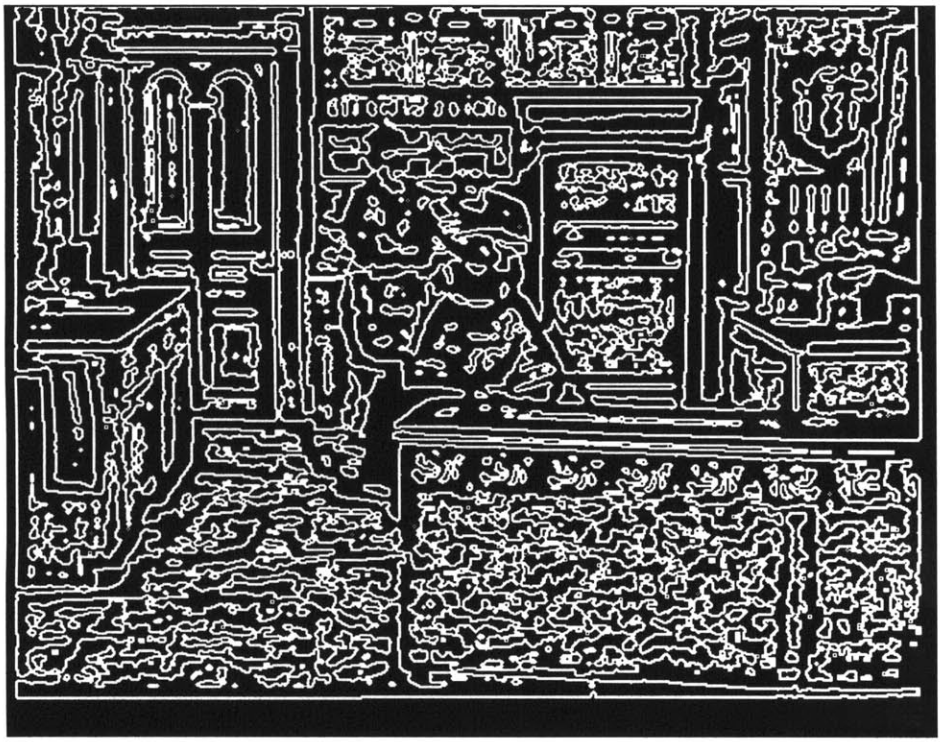


Figure 2-5: Unprocessed edge image

Chapter 3

The Model

3.1 Basics

3.1.1 Model Representation

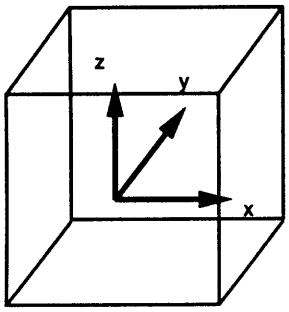
Given some footage shot on a set, we'd like to construct a conceptual model of the set. The purpose of the model is to capture and represent the spatial relationships between the elements of the set. Each object will be modeled by a simple three dimensional shape, such as a cube. The shape can be scaled in X, Y, and Z so that it occupies the appropriate volume; translated to the appropriate position; and rotated to face the correct direction. Surface attributes, such as color, can also be specified for each object.

An English description of the "I Love Lucy" kitchen might contain "Along the back wall there is a green refrigerator. The refrigerator is 5 feet tall, 30 inches wide and 2 feet deep." The model would contain a similar description:

```
object (Refrigerator) {
    instance(cube);              /* create a 1ft x 1ft x 1ft cube */
    scale(2.5, 5.0, 2.0);
    translate(2.0, 0.0, 10.0);   /* Move it to the back wall */
    color(0.0, 1.0, 0.0);      /* Very Green */
}
```

The model will be used to reason about the footage, usually by constructing 2-D images. These projections will direct alteration of the original 2-D footage. 2-D perspective renditions

of the model will be used often. In order to perform those renditions, more than the name of a primitive shape will have to be specified. There is no 3-D information inherent in the name of the shape. Each shape is broken down into faces, which are polygonal. The Polygons are described by their edges, which are line segments. Each line segment can be described by two points in three dimensions. So, a primitive shape is described by a list of points and the connections between the points which form faces. The description for a unit cube at the origin is shown in figure 3-1.



Point	X	Y	Z
1	-0.5	-0.5	-0.5
2	-0.5	-0.5	0.5
3	-0.5	0.5	-0.5
4	0.5	-0.5	-0.5
5	-0.5	0.5	0.5
6	0.5	-0.5	0.5
7	0.5	0.5	-0.5
8	0.5	0.5	0.5

Face	Points			
1	3	5	2	1
2	4	7	3	1
3	2	6	4	1
4	5	3	7	8
5	6	2	5	8
6	7	4	6	8

Figure 3-1: The *unit-cube* model primitive.

3.1.2 The Modeling Tool

If detailed plans and blueprints of a set are available, then making an electronic model of the set is the straightforward task of translating the plans into our model representation. Creating the model is more interesting and difficult if no such information is available. Instead we will use our eyes and spatial reasoning skills to create the model. An interactive graphical modeling tool is used to illustrate how the model looks, provide feedback on how well the model corresponds to the set, and record the model as it is created.

The modeling tool provides visual feedback in the form of viewports. A viewport is a region of the screen which contains a 2-D rendition of the model under construction. There are two basic kinds of viewports: perspective and orthographic. A perspective viewport simulates the real world visual experience of viewing the model. When a person or a camera looks out at the world, the closer an object is to the viewer, the more view area it will occupy. For example, if a coffee mug is viewed from 100 feet away, it will be a small speck. When placed right against the nose, it will block the visibility of all other objects. Perspective viewports allow the modeler

interactive approximations to how renditions will look.

In an orthographic viewport, an object's size does not depend on how far it is from the viewer. Typically there are three orthographic viewports: a front view, a side view, and a top view. These viewports are overlaid with a regular grid. The grid spacing is set to a convenient spatial measurement, such as one foot per division. If the modeler has made any decisions about the size or positions of the objects, the orthographic viewports can be used to ensure these decisions are carried out.

Figure 3-2 illustrates the four common modeler viewports: top, perspective, front and right, in order from left to right, top to bottom. The top, front, and right viewports contain orthographic renditions.

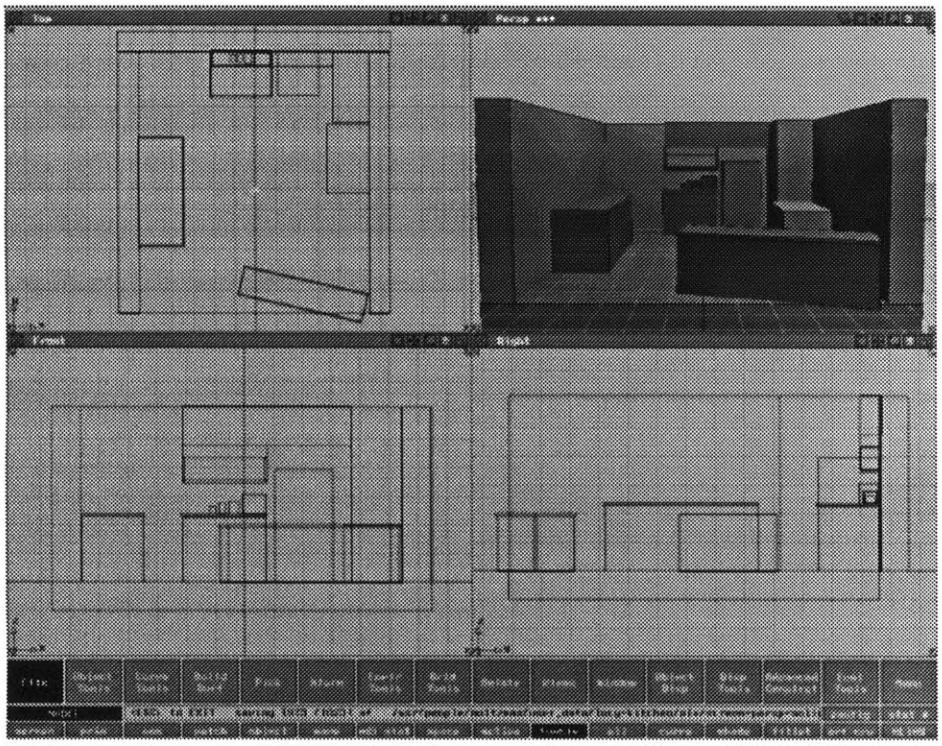


Figure 3-2: Perspective, Front, Top, and Right Viewports

3.2 Creating the Model

3.2.1 Initial Guesswork

A model element is created for each object in the scene. The elements are created and positioned on the grid under cursor control. The size of each side of a model element is changed according to an intuitive guess at the size of the real object. For example, one could measure the height of the counter tops in their own kitchen, and use that height for the counter tops in the model. The relative sizes and positions of all the elements of the model should not be hard to estimate from looking at a frame of the movie footage.

3.2.2 Getting the model right

The perspective viewport is essential for improving the accuracy of the model. By overlapping a frame of the footage with the contents of the perspective viewport, the errors in the model can be judged. To keep the modeling process interactive, the perspective viewport should directly support using a 2-D image as the background for its rendition. Then, objects can be moved and sized to align directly with a photograph of their real-life counterparts.

A multitude of different photographs can be taken of the same scene. The position, orientation, and viewing angle of the camera effect the content of the photograph. Laying real footage into the background of the perspective viewport is only going to be useful if the perspective viewport's virtual camera is set up to match the camera which took the original footage. Most importantly, the viewing angle of the original lens must be determined. The viewing angle is inversely related to the focal length, and determines the degree of perspective foreshortening in the image. A wide angle lens has a lot of perspective foreshortening, while a telephoto lens has little. The position and orientation of the camera are less important, as a change in position or orientation of the virtual camera can also be thought of as a simple translation and rotation of the three basis functions of the virtual coordinate system.

3.2.3 Finding the Vanishing Point

In a perspective rendition, all lines perpendicular to the view plane will converge at a single point in the image. This point corresponds to an infinite distance from the lens, and is called the

vanishing point. Finding the vanishing point of the movie frame and the perspective window will help us deduce the correct camera parameters. The vanishing point of the perspective viewport can be found easily. A light source is placed at a great distance in front of the camera. The light source is given a distinctive color, and a falloff rate of zero (illumination is constant with distance). The vanishing point will be marked by the light source's color. The light source is far enough from the camera when moving it up and down or from side to side has no effect on its location in the view plane.

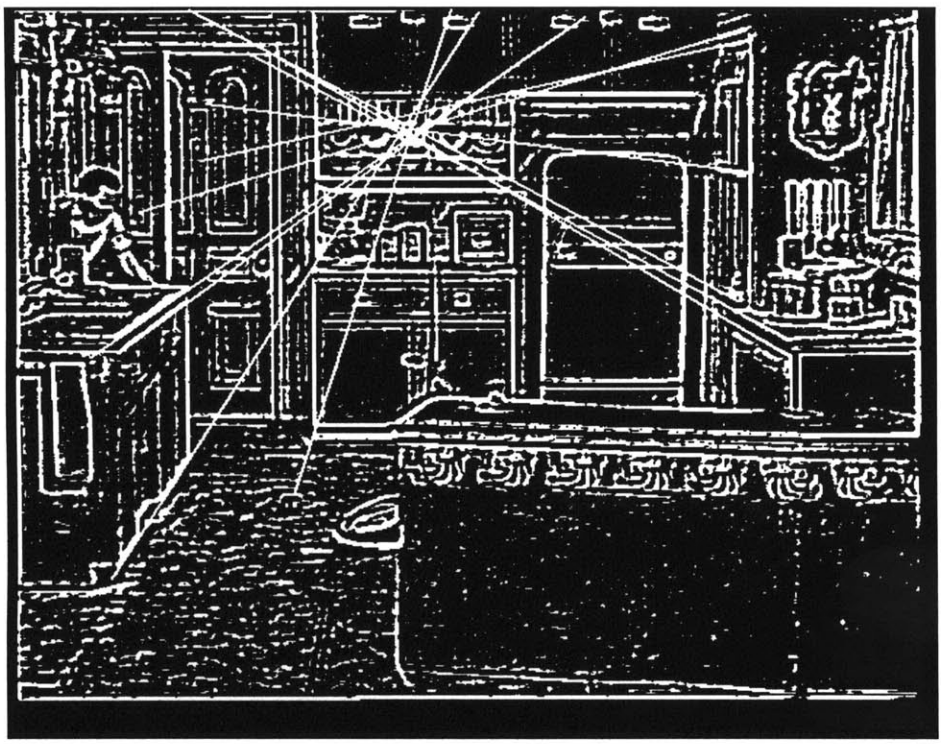


Figure 3-3: Lines away from the camera converge at the *vanishing point.*

The vanishing point of a real image is a harder to determine. An edge operator is applied to the image. Edges with constant slope over reasonable interval are projected in both directions to the borders of the image. Many of those edges should cross in one place. That spot is the vanishing point. An example is shown in figure 3-3 If the image does not contain objects which have straight edges, or the camera is not pointed in the same direction as some of the edges, then the vanishing point will not be found by this method.

3.2.4 Matching View Angles

The view angle can be determined by an iterative process. The vanishing point augmented edge image, developed in section 3.2.3, is placed into the background of the perspective viewport. The viewport's camera position and orientation are set to make the vanishing point and floor of the model align with the vanishing point and floor of the frame. Next, an object is moved and scaled so that the outline of the front surface aligns with the image of the real object. Lastly the object's depth is scaled to try and make the rest of the object's outline align with the image of the real object. This will only be possible if the view angle of the perspective viewport matches the view angle of the camera. The nature of the misalignment will indicate whether the view angle should increase or decrease. When the view angle is changed, the vanishing point will move, and the alignment process will have to started over.

3.2.5 Finishing Up

After the vanishing points have been found and the perspective viewport is set to have the same viewing angle as the movie camera, the modeling task can be completed. Each model element is moved to overlay the object it is modeling. Its size is adjusted to match. Moving an element towards or away from the camera will have the same effect as changing its size. A deduced constraint can usually be used to remove the ambiguity. For example, the object might make contact with the floor, or be the same height as another object. Figure 3-4 shows the kitchen model having just been aligned.

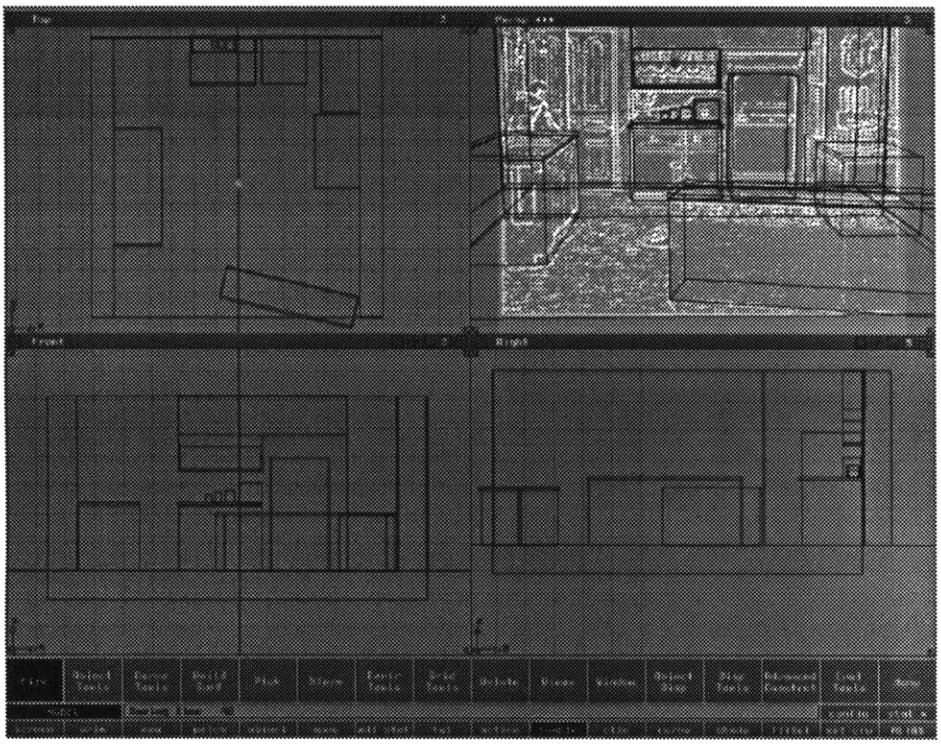


Figure 3-4: The model is aligned to overlay the image objects

Chapter 4

Flaws in the Model

Modeling is a labor intensive process. The data is entered by hand and corrected by trial and error. Sometimes the model will be flawed because the creator did not spend enough time getting it right. For example, an object might be in the wrong place. Other times the flaws are due to the inability of the model to represent what is contained in the scene. It might unreasonable to expect someone to spend the time building a model which accurately represents the contents of a scene. Imagine constructing a polygonal model of a pile of garbage, or plate of spaghetti.

4.1 Inaccurate Descriptions

An object being modeled may not fit the description of any of the primitives available. A surface may kink or bump in an unusual way. The modeler could represent the the object by using more than one primitive, but this could become cumbersome when the object doesn't naturally break apart into smaller, more regular components. A whole new primitive could be constructed which describes the object perfectly. Creating primitives is a time consuming task, however. The modeler might just decide it isn't worth their time, and live with the artifacts. Even if the modeler spends the time to get the object to match its image counterpart, it might not align perfectly when compared to an image filmed from another view.

In our example model, the Ricardo kitchen kitchen was modeled using only cubes. Cubes worked well because the scene is populated mostly by man made objects which are comprised

of elements occupying rectilinear volumes. There are a couple of rounded corners in the image, however. Rather than create more complex primitive elements, these errors were left in the model. We will see how they effect the processed images, and explore how we can reduce the problems caused by inaccurate descriptions.

The model will be used to divide the movie frames into segments according to object, associating the textures of the movie frames with the appropriate objects in the model. Errors in the shape of the model elements will cause the division of the movie frames to be made incorrectly, associating texture information from an object with the wrong model element. When the output frames are rendered, the misplaced texture might get processed incorrectly due to its association with the wrong part of the model.

4.2 Missing Detail

An intrinsic part of the modeling process involves deciding what objects are present in the frame, and if the objects are worth modeling. Most pieces of footage are too complex to model each visible component. It would be overly cumbersome, to model the dust which contributes to the texture of the floor, or all of the clutter on a crowded surface.

In many cases, it is unimportant to model the smallest levels of detail. The pixel data from the original frame will be used to form the output frame, so the detail will still be represented. Artifacts will only occur when the alterations being performed on the frame require different processing be performed on objects which were lumped together. The decision whether or not a small object or detail needs its own model element must be made in the context of how the model will be altered. Also, it may be possible augment the larger element with detail about how to process its components, rather than creating new, smaller elements. An example of this sort of detail is the texture maps used to shade the surfaces. The type of detail needed will be defendant on the type of processing performed. For colorizing, a chroma map might be necessary. When rendering from a new perspective, a bump map would be helpful.

4.3 Dynamic Elements

The discussion thus far has ignored the possibility that the scene evolves from frame to frame. Actors and other objects enter, move about, and leave the scene. Making the problem even worse is the possibility that objects will change shape. As actors move, their various body parts change volume and move in relation to each other, causing their overall shape to change. Our model was created from a single frame and has no mechanism for updating the locations of the objects on a frame by frame basis. Also, all the model elements are assumed to be rigid bodies. For now, dynamic elements will simply be omitted from the model. The resulting artifacts will vary depending on the alterations performed, falling into the category of missing detail. The appearance of the artifacts will vary from frame to frame, however, since their cause is moving. Later, we will explore detecting the actors so they can be modeled separately.

Chapter 5

Colorization: A sample application

5.1 Current Colorization Technique

Television broadcasters are showing an increasing number of movies which were originally filmed in black-and-white, but now have color added. The current colorization technique does not involve structural or 3-D modeling of the scene [20]. The process is much like filling in a color-by-numbers book. Staying within the lines is not a big problem because the eye is less sensitive to chrominance edges than to luminance edges. The frames of the movie are digitized and painted on using an interactive 2-D paint tool. Using a tablet and keyboard, the scene is divided into regions which require separate colors. Each region is assigned a hue. Saturation is derived from the black and white intensity information and assigned on a pixel by pixel basis. Saturation and intensity are usually inversely related, but the exact relationship between the two is decided on a film by film basis to allow the art director control over the overall color scheme of the movie, from pastel to neon. Intensity is also assigned on a pixel by pixel basis, being copied directly from the black-and-white image.

Colorization systems attempt to track the regions while colorizing the rest of the frames in a scene. The black-and-white version of the previous frame is compared pixel by pixel to the current frame. On simple scenes, upwards of 96% of the pixels will have unchanged intensity values from one frame to the next [20]. Where the intensities remain unchanged the current frame receives the same hue as the previous frame. For the remaining parts of the image, where the interframe intensities differ, a combination of techniques are used to figure out the hue

at each of those points. Primarily there is a motion tracking technique which compares the intensity of an unassigned pixel to its neighbors, using their intensities from the previous frame. If the algorithm detects a match, it assumes the pixel belongs to an object which has moved between frames and assigns the current pixel the same color as the neighboring pixel received in the previous frame.

An interframe intensity change may not be a true change, but rather due to the presence of noise. When the 8 closest neighbors to an unassigned pixel have identical hue, the center pixel assumes that hue as well. Any pixel which remains in question is given its hue by the operator. While the number of pixels per frame which require manual intervention may be low, there will be fifty to a hundred *thousand* frames to colorize.

Any time the movie jumps from one camera angle to another, or cuts between scenes, the colorization process must be started again, with the initial frame being hand colorized by the operator. Movie shots last from a few seconds to a couple of minutes. There are $\approx$ 1500 individual shots in a typical feature length film. For each shot, the art director must choose colors and the operator paint on a frame of the footage. The 2-D motion tracking can not handle too large a change. When it becomes confused by a rapidly moving camera, the operator may need to restart the system by hand colorizing frames. Also, the operator must intervene whenever a new actor or object enters the scene. As the contents of the scenes are not modeled, they are difficult to categorize and log in a database. Often colorized movies are plagued with objects which change color from shot to shot.

5.2 Using the Model to Colorize

We use the 3-D model to colorize the frames of the image. Each object in the image is assigned a color. This can be done inside the modeling tool, or later. The model is rendered using a good quality surface shading algorithm, such as Phong shading. Since the original frame already has highlights and shadows, the lighting model used by the renderer should be ambient. Some additional light sources may be desirable to bring out color highlights.

Typically, a renderer produces three color values for each pixel: red, green, and blue. By converting these values to YIQ color space, the chrominance is separated from the luminance:

$$Y_{ij} = 0.299r_{ij} + 0.587g_{ij} + 0.114b_{ij} \tag{5.1}$$

$$I_{ij} = 0.596r_{ij} - 0.274g_{ij} - 0.322b_{ij} \tag{5.2}$$

$$Q_{ij} = 0.211r_{ij} - 0.522g_{ij} + 0.311b_{ij} \tag{5.3}$$

Y_{ij} is the luminance, or black and white value for the pixel at location ij. I_{ij} and Q_{ij} encode the chrominance of the pixel. The colorized image is formed using luminance values from the original black and white image and chrominance values from the rendition of the model. The black and white image is scaled by 0.93 for use as the Y channel of the colorized frame. The scaling is necessary because the maximum possible value of the Y channel is $0.229 + 0.587 + 0.114 = 0.93$ of the maximum value possible in the r, g and b channels, while the black and white image's maximum possible value is the same as that of the r, b, and g channels of the rendition. So:

$$Y_{ij}^{out} = 0.93 Orig_{ij} \tag{5.4}$$

$$I_{ij}^{out} = I_{ij}^{model} \tag{5.5}$$

$$Q_{ij}^{out} = Q_{ij}^{model} \tag{5.6}$$

$$r_{ij}^{out} = Y_{ij}^{out} + 0.9557I_{ij}^{out} + 0.6229Q_{ij}^{out} \tag{5.7}$$

$$g_{ij}^{out} = Y_{ij}^{out} - 0.2721I_{ij}^{out} - 0.6483Q_{ij}^{out} \tag{5.8}$$

$$b_{ij}^{out} = Y_{ij}^{out} - 1.1052I_{ij}^{out} + 1.7046Q_{ij}^{out} \tag{5.9}$$

Figure 5-1 is an original black and white frame from Lucy's kitchen. In figure 5-2, the elements of the model have been assigned colors, and the model rendered. The luminance channel from the original is combined with the chrominance channels from a rendition, resulting in the colorized frame in figure 5-3.

5.3 Review of Problems

5.3.1 Inaccurate Descriptions

The colorized footage will have distinct boundaries between colors. Sometimes these boundaries will not coincide perfectly with the borders of the objects in the scene. The error is caused by imperfections in the way the model represents the scene. For example, the refrigerator in the Ricardo kitchen is modeled by a scaled cube. Cubes have square corners, but the refrigerator has rounded corners. The refrigerator's color appears to bleed onto the wall behind it.

5.3.2 Missing Detail

Objects within the scene which were not modeled, either because they were hard to distinguish, or too small and numerous to seem worthwhile, can not be colored properly. They will take on the color of whatever is behind them.

Figure 5-1: Black and white original from "I Love Lucy"

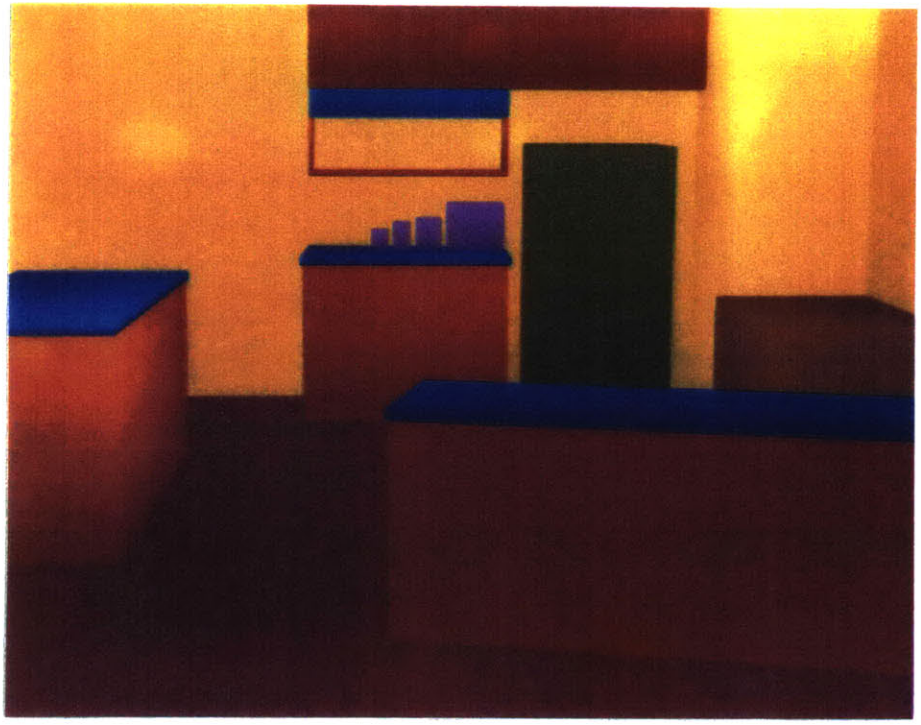


Figure 5-2: Solid rendition of model with assigned colors

Figure 5-3: Colorized frame from "I Love Lucy"

5.3.3 Dynamic Elements

Since dynamic elements were not modeled, they come under the same heading as missing detail. The actors will be colored to match whatever is behind them. This can be particularly disturbing, because they may obstruct more than one object, causing the actors to have multiple areas of miscoloration. As the actors move, they pass in front of different objects. This means that their miscoloration will change with time.

5.4 Advantages of Model Based Colorization

Model based colorization has a few advantages over the tradition method. The color assignments are made to the model, rather than the image directly. There won't be the shifting of color assignments that tends to accompany colorization. As new shots are introduced, the model maintains its validity. In the example of a sitcom, the same model can be used not only from shot to shot and scene to scene, but from episode to episode. Model elements do not have to describe their real world counterparts perfectly because the human visual system has low spatial resolution when it comes to detecting color. Detail which was left unmodeled, as well as finely colored textures can be handled by introducing chroma maps to the faces of objects. Chroma maps are the color equivalent to texture maps.

Chapter 6

Evolving the Model

Creating an accurate model of the set by hand is a difficult task. Some objects on the set will have complex contours. Even simple contours can be time consuming to model. In early experiments, all objects were modeled with scaled cubes, because cubes worked well for most of the objects in our scene. There was noticeable error, however, in the top of the refrigerator in the Ricardo kitchen. The corners of the refrigerator are rounded. We could have replaced the model primitive for the refrigerator with one that had rounded corners, but a more automatic approach was desired. The footage was used to refine the final shape of the model.

An edge operator is applied to a frame of the footage. The result will contain the actual contours of the objects. The model, at the onset, contains only cubes. The model entry for the object can specify that a particular object requires its top, bottom, left or right edges conformed to the contours in the footage. The front face of the object is conformed, any changes made to the face is propagated along the sides to the rear of the model. The end result is an extruded volume whose front and back resemble the outline found in the footage. The depth of the extrusion is identical to the original depth of the model element.

The front face is adjusted by repositioning the points which define the edges of the polygones that make up the faces. The original *unit-cube* model element only contained the eight points which represented the corners. A detailed *unit-cube* model is needed, subdividing the faces so that there will be control points along the edges for the alignment software to adjust. The face subdivisions could be made in a regular grid, for simplicity, or added only where needed to provide another control point.

The control points are projected back and forth between the model's 3-D virtual world coordinate system, and the 2-D image plane. The matrix transformation which maps points from world space to image space, called the camera transformation matrix or viewing transform, actually maps the image space as a 3-D box. A planar projection is is made on all the points in image space to form the final image plane. Since the image plane is perpendicular to the X and Y axes in image space, the planar projection is created by dropping the Z coordinate. When the alignment software maps points to the image plane, it will keep track of the Z coordinate value of the point so that the inverse mapping can be performed.

First, the points in the model element are searched for maximum and minimum x, y, and z values. The search will allow us to determine which points lie along the edge being adjusted. Each point along the edge is projected from 3-D model space to 2-D image coordinates. The viewing matrix used to project the points is designed to match the camera which shot the original footage, so comparision can be made to the edge image. The edge image is examined at the location of the projection. If there is no edge at the location of the projection, a search is performed. When conforming the top or bottom of an object, the search proceeds vertically. When conforming the left or right sides, the search is done horizontally.

Two edges are found, one on each side of the projected point. The closer edge is assumed to correspond to the edge of the modeled object. The point along the edge is projected back into world space using the inverse of the viewing matrix and the z value of the original transformation. The appropriate coordinate value in the model is changed. To preserve the spacing of points along the edge, and keep the front face planar, the model point is only perturbed in one dimension. The corresponding control point on the backface and the points along the line joining the front point to the back point are found and perturbed by the same amount.

Edge images contain gaps in contours. The contour found may not be the correct one because the search may have proceeded through a gap. To prevent that from happening, the search aborts rather than proceeding a significant distance from the projected control point. When the search fails, the varying coordinate from the previous search is used instead. How far the search proceeds before failing will determine the maximum perturbation allowed. Ten pixels was arbitrarily chosen during our test work, as objects which did not share an adjoining edge were usually spaced more than ten pixels apart, and ten pixels was a sufficient distance to move

all the points into alignment. Another approach would be to ignore the gaps in the contours and find all the perturbation along the edge. A smoothing constraint would then be applied, where each point would not be allowed to deviate from its neighbors by more than a small amount. When a point's deviation exceeds the threshold, which would define the maximum curvature of the edge, it is interpolated from its neighbors.

Figure 6-1 shows the refrigerator in the original frame. The edge reduced frame, shown in figure 6-2, contains the cues used to alter the model. The original model of the refrigerator, shown figure 6-3 is replaced with a detailed cube model element. The front face of the model takes on the shape of the outline and is extruded back to the refrigerator model's original depth, forming the new refrigerator model, figure 6-4. The new model is rendered in figure 6-5, and the rendition used to colorize the frame with less artifacts. The colorization is show in figure 6-6. A rendition of the original model, and the corresponding colorization can be found in Chapter 5, figures 5-2 and 5-3.

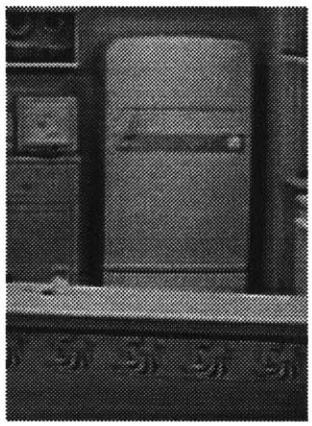


Figure 6-1: Refrigerator from the set of "I Love Lucy."

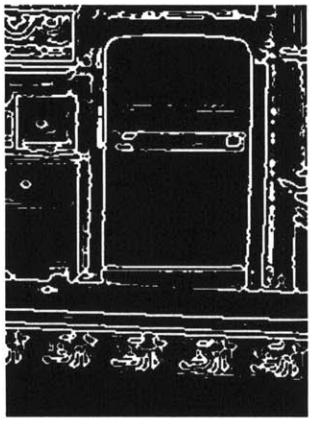


Figure 6-2: Edge reduced frame reveals outline of refrigerator

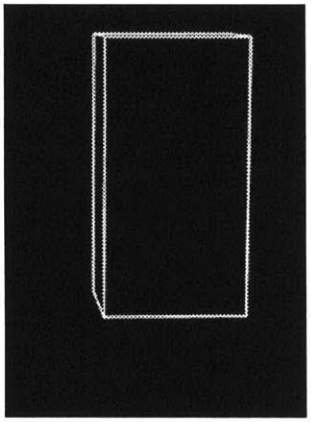


Figure 6-3: The initial model of the refrigerator

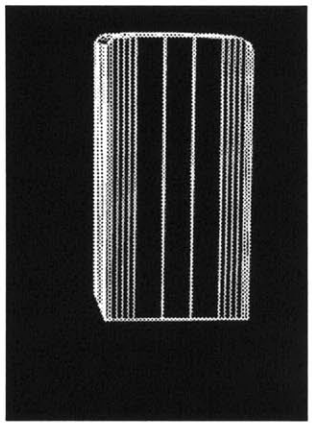


Figure 6-4: Refrigerator model is altered to match outline

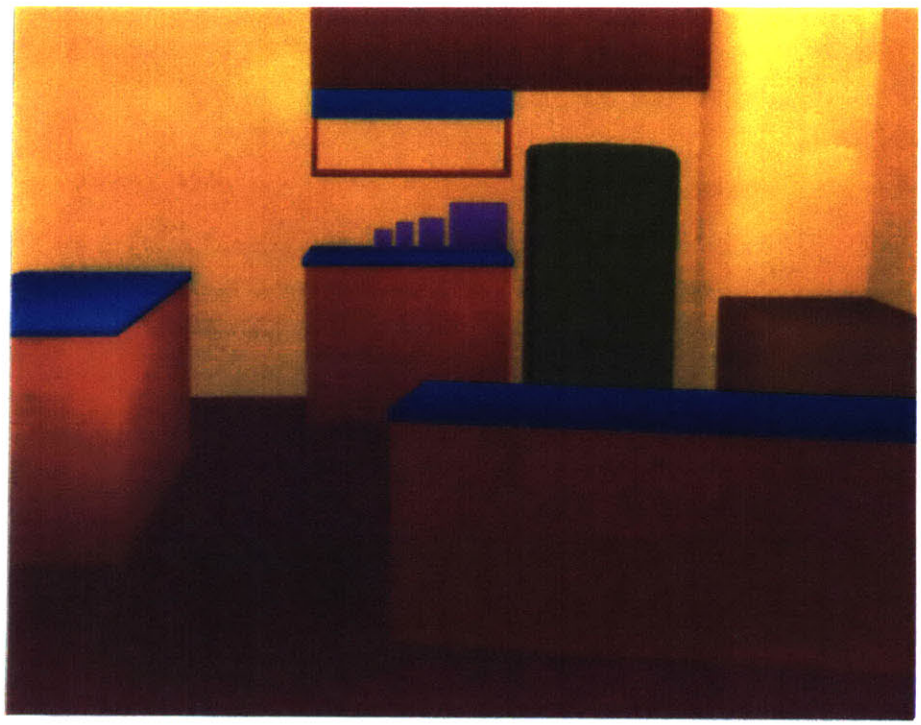


Figure 6-5: Rendition of the automatically corrected model

Figure 6-6: Colorized kitchen made from adapted model

Chapter 7

Finding the Actors

To limit the scope of this project it was decided to omit the actors and other moving elements of the scene from the 3-D database. Instead, they are detected, isolated and saved as 2-D patches. These patches might be discarded, composited unchanged into the final footage, or processed differently from the rest of the footage.

Motion prediction algorithms break up an image into small blocks and track how each block moves from frame to frame [28]. Forming a histogram of the motion vectors can isolate camera motion [22]. The blocks which move at a different velocity from the camera contain actors in motion. Motion prediction is not sufficient for finding the actors because they might hold still over a long period of time. A conservation requirement could be imposed, requiring a block which is flagged to contain a part of an actor to maintain that flag even when its motion vector is zero. The block would pass the flag along with its contents when its motion vector is nonzero, even if the motion matches the global pan. If all actors were rigid bodies, such a constraint would be sufficient, but then they would also be easy to model. Actors will deform from frame to frame, changing the number of blocks necessary to represent each actor.

Even from a single still frame, people can easily pick out the actors. That's not surprising; people manage to decipher the contents of the frame. The database is also a representation of what is contained in the frame. Finding the actors is a matter of figuring out what is in the frame that is not in the database.

7.1 Preparing the data

While the database and a movie frame are representations of the same scene, they are composed of different kinds of data. In a 640×512 image there will be 327680 samples. The database is a considerably sparser set of data. There might be 16 objects, each with 10 parameters: object type (*i.e. cube*), x, y, and z location, width, height, and depth scale factors, and color. The database will also contain descriptions of the light sources, camera parameters, and more geometric information about the object primitives. All in all, for 16 objects composed of one primitive, 2 light sources, and the camera, there would be $\approx$ 250 data points. In order to compare the database with the frame image, they each need to be transformed onto common ground.

For comparison to the movie frame, the database is rendered into an image. Two kinds of renditions are readily available: solid and wire frame. Many sorts of solid renditions are possible, but they all attempt to create a realistic looking image from the database. Even so, the image produced is inappropriate for direct comparison to the movie frame because the rendition lacks all of the surface detail of the real image. The detail cannot be rendered because it is not present in the model.

Rather than increasing the amount of data in the rendition, the movie frame can be processed to remove detail. One method of reducing the data in the move frame is to run an edge operator over it. The result is a binary image which contains white where there are edges, and black elsewhere. For comparison to the edge image, the database is rendered as a wire-frame. The wire-frame rendition is a line drawing, representing only the edges of the objects in the model. No surface shading is performed.

The wire-frame contains only objects which are in the database. The edge image contains all the objects. The database objects are removed from the edge image by removing edges which correspond to wire frame edges. What is left should be the edges of things which are not modeled. In practice, this is more difficult. The edge image is not a clean representation of the edges. Detail within objects sometimes create edges. Sometimes objects look a whole lot like what they are in front of, and edges are lost. In practice, this technique did not work well.

7.2 The background frame

Rather than reduce movie frames to edge images, the database could be augmented with textures from footage. The texture mapped rendition of the model would be compared to the movie frame directly. For this to be useful, we must gather textures which are not polluted by the presence of the actors. This can be done by taking the textures from a multitude of frames. The reason we did not model the actors was because they were moving around, so they should be obscuring different textures as a function of time. It is possible that the actors will keep some object's faces concealed over the entire movie. When that occurs, no texture should be assigned to that area. Otherwise, the texture would contain a part of the actor, and the actor would be present in the rendition.

If the camera does never moves, a simpler 2-D process can be used to create a single actorless reference image, called the *background frame.* The background frame has the same contents as a frame of the movie, except it is missing the actors. A background frame is constructed by compositing sections from two or more frames of the movie. In figure 7-1 Ricky and Fred are in the left half of the frame, so we grab the right side. In figure 7-2 they are on the right, leaving the left side empty. Combine the two halves, and we get figure 7-3, the background frame.

7.3 Comparison to the Background Frame

7.3.1 Frame Differences

The background frame is missing the actors. The desired result is a frame with only the actors, which we'll call the *foreground frame.* Actually, there should be a foreground frame for each movie frame. All together, they will be the *foreground footage.* While it was acceptable to create the background frame manually, the foreground frame must be created automatically. It would take too much effort to create minutes worth of foreground footage by hand.

Given a movie frame and the background frame, the foreground frame is the difference. A first pass at creating the foreground frame is to subtract, on a pixel by pixel basis, the background frame from the movie frame. The difference image is passed through an absolute value filter, and then thresholded. The result is a segmentation mask which is black anywhere the luminosity of the movie frame differs from the background frame by less than the threshold,

Figure 7-1: This frame contributes to the right half of the background frame

Figure 7-2: This frame contributes to the left half of the background frame

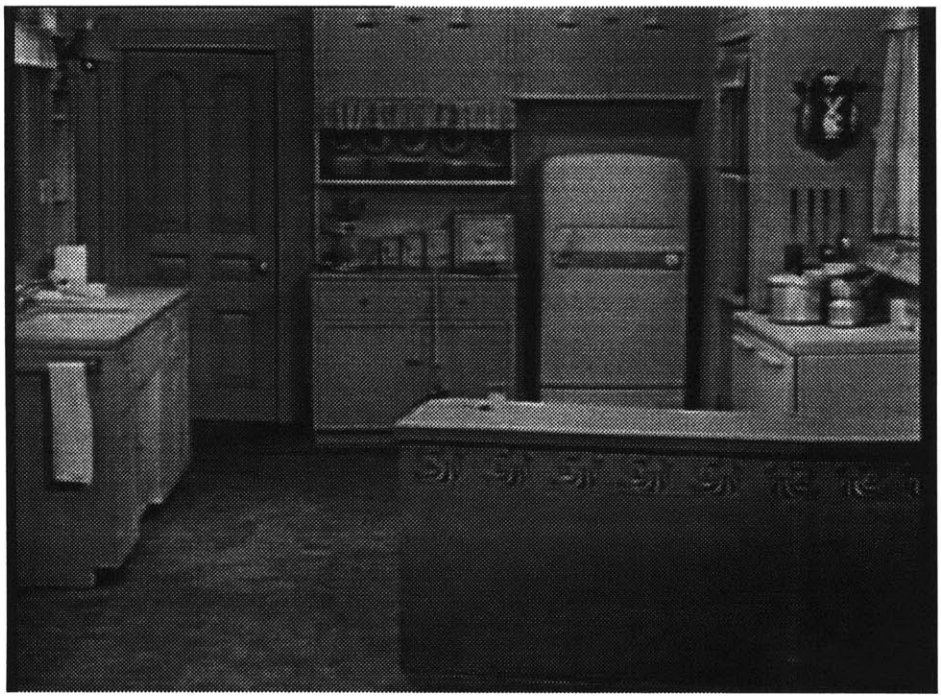


Figure 7-3: The background frame

and white elsewhere:

$$Diff_{ij} = Image_{ij} - Background_{ij} \tag{7.1}$$

$$Mask_{ij} = \begin{cases} 255 & \text{if } |Diff_{ij}| \geq Thresh \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

The mask is used to select which pixels from the movie frame will be used in the foreground frame:

$$Foreground_{ij} = \begin{cases} Background_{ij} & \text{if } Mask_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

Noise in the background or the movie frame will come through in the foreground image. If the noise is above *Thresh*, the mask will contain a white pixel at the noise location. Also, there will be holes in an actor where the actor and the background are similar in luminance value. One might argue that such holes are acceptable; if the foreground and background don't differ,

it doesn't matter which receives the pixel. The reason for saving aside the actors, however, is that they must be processed differently from the rest of the frame. Therefore the pixel which might be in either the foreground or background should be in *both* the foreground and the background.

7.3.2 Correlating with the Background

Block comparisons between the background and movie frames give better results than pixel by pixel comparison. Noise spikes do not have as great an effect on the mask because they are averaged out over the block. The foreground and background would have to be similar over an entire block for the block to drop out of the mask. A more complex operator must be used to compare the blocks. The linear correlation coefficient, known as *Pearson's r* is used [33]. The correlation for an $M \times M$ block located (x, y) is defined as:

$$r = \frac{\sum_i \sum_j (f_{ij} - \bar{f})(b_{ij} - \bar{b})}{\sqrt{\sum_i \sum_j (f_{ij} - \bar{f})^2} \sqrt{\sum_i \sum_j (b_{ij} - \bar{b})^2}} \tag{7.4}$$

where $i = x - M/2, \ldots, x + M/2$ and $j = y - M/2, \ldots, y + M/2$. b_{ij} is the pixel at location (i, j) in the background frame, B, and f_{ij} is the pixel at location (i, j) in movie frame F^t. $\bar{b}$ and $\bar{f}$ are the mean values of the blocks within B and F^t, respectively:

$$\bar{b} = \frac{\sum_i \sum_j b_{ij}}{M^2} \tag{7.5}$$

$$\bar{f} = \frac{\sum_i \sum_j f_{ij}}{M^2} \tag{7.6}$$

The correlation coefficient, r, is scaled by the denominator in 7.4 to range from -1 to 1. When r is 0, the blocks are completely unrelated. The closer $|r|$ is to 1, the better the match between the two blocks. Negative values of r indicate a correlation of one block with the negative of the other block. We are not interested in negative matches, so we will preserve the polarity of r. As with difference images, a mask image, *Mask* is formed using a thresholding

operation:

$$Mask_{xy} = \begin{cases} 255 & \text{if } r \leq Thresh, \ (0 \leq Thresh \leq 1) \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

The foreground frame is formed from the mask using equation 7.3.

The block size, M of the correlation is an important parameter. If the block size is small, a slight misalignment between the background frame and the movie frame, as would occur if the camera jitters, would cause the correlation to perform very badly. Small changes in illumination will also adversely affect correlation when the block size is small. When the mask is formed with a small block size, actors will contain holes where the background matches the actor.

Larger block sizes will bring out the importance of the larger image features. An actor is an example of a large image feature. Unfortunately, at large block sizes the borders of the mask will not hug the borders of the actors. A combination, or *multiscale* approach is mandated. The correlation coefficients are calculated at both large and small block sizes. The large blocksizes are used to find the objects. At the borders of the objects, the mask formed with the small block size correlation coefficients is searched, refining the edges of the mask. An edge operator can also be applied to the movie frame and the edge image searched for the location of the borders.

The mask may still need to be cleaned up. Actors should not contain any holes and spurious noise needs to be removed. The holes are removed by flood filling each "blob" in the mask. Noise is removed by removing any blobs which span a small number of pixels. The minimum pixel size is set by creating a histogram of blob sizes. If there are three actors in the scene, then the minimum blob size is set to keep the largest three blobs and filter out the rest. Alternatively, the minimum blob size can be fixed to a moderately small number of pixels, which will remove noise but leave actors.

Chapter 8

Combined 2-D/3-D Techniques: Colorization by Flood fill

Our basic 3-D colorization technique is to take the intensity for each picture element from the original footage and use the color channels from a rendition of the model. Where the model does not match the footage, there will be errors. A slight shape error in the model will yield small, perhaps unnoticeable error. Large errors will occur when there are objects which have not been modeled, or which were modeled and moved. One example is an actor. As the actor strolls across the set, he or she will change color to match the objects occluded by the actor.

A new approach was developed to isolate the moving actors. An edge filter is applied to the movie footage. The resulting binary image contains white pixels where there are edges and black pixels elsewhere. Seedpoints are chosen by hand, and the insides of objects are flood filled. The outlines of the actors stop the flood fill operation. The process is iterated until all the objects are filled. The result is a mask which is applied to the color rendition of the model. Areas in the mask which are black correspond to actors and other detail which was missing from the model. Those areas are blackened out on the rendition. The chrominance channels of the rendition are then combined with the intensity information from the footage, as outlined in Chapter 3. Thus, the actors have been removed from the model based colorization process. The compliment of the mask is available to cut out the actors so they can be colorized using a different method.

Unfortunately, the edge image does not cleanly represent the outlines of the objects. In setting the parameters for the edge detector, there is a tradeoff between getting a surplus of edges, including outlines of surface detail, and losing portions of edges where the light gradient was weak. With the parameters tweaked to give a usable edge image there will still be a little of both effects. Some excess edges are not terribly harmful to the flood fill process. Spurious edges could cause a surface to be broken up so that a single flood will not fill it, but multiple seedpoints can be specified to circumvent the problem.

The model is used to ensure there is a continuous edge at the border of each surface. A binary wire frame rendition of the model is created. The edges from the wire frame are added to the edge reduction using a boolean *OR* operator. A combined edge and wireframe is shown in figure 8-1. Wire frame rendition usually show lines which pass behind surfaces. Those lines are a nuisance for the flood fill algorithm because they break up the surfaces into multiple regions. To prevent those lines, the model was first rendered using a surface renderer with and the depth value of each output pixel was saved. Then the model was rendered using a wireframe renderer, also calculating the depth of each pixel. Any time the wire pixel was behind a surface pixel, it was not drawn into the wireframe output.

The edges of the actors may have gaps in them as well. When there is a break in the outline of an actor, the flood of the surrounding area will invade the actor, causing a miscoloration. Figure 8-2 shows flood fills which have run into trouble. Gaps in the actors' outlines need to be fixed, even if by hand. In figure 8-3 the outlines have been repaired manually. Small discontinuities in the actor's outlines can be found automatically by following the edges, and where they end, searching the neighborhood for the end of another edge. If one is found, the edges are joined. This algorithm meets with some success, but often fails due to large discontinuities, and edges which spiral in on themselves. To prevent all the little edges in the picture from getting joined together into a jumbled mess, small edges are not allowed to join other small edges, but may join with long edges. Once a small edge is part of a long edge, it is considered a long edge and may be connected to a nearby small edge at its other end.

The flood-fill based colorization process is flowcharted in figure 8-4. Note that the model still determines the borders of the color areas. The mask resulting from the flood fill process gives a yea or nay vote as to whether to colorize a pixel or leave it in black and white. It has

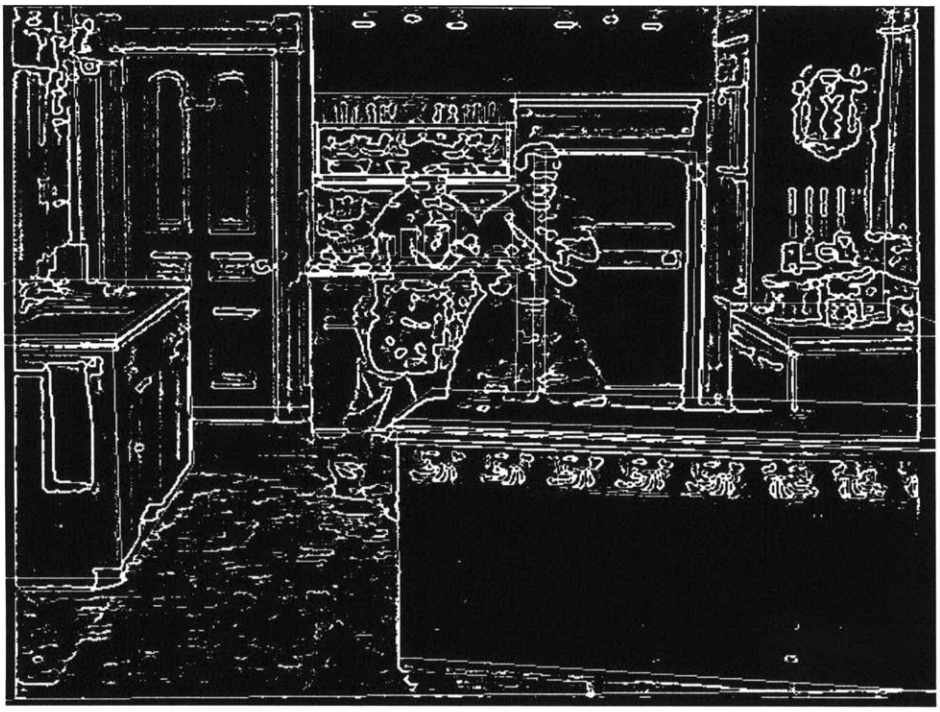


Figure 8-1: Combination edge and wireframe rendition

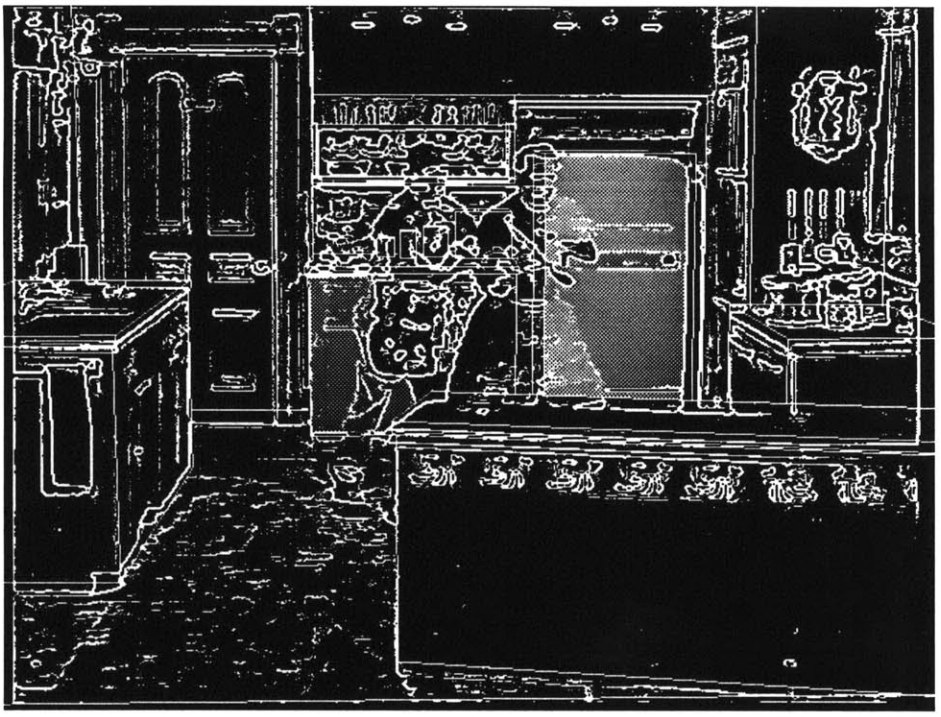


Figure 8-2: Gaps in the actors' outlines cause leaky floods

Figure 8-3: Edge-and-wireframe corrected by hand

already been noted that sometimes the model does not correspond well to the model. Since the flood fills use edge detail to determine their shape, this is a perfect opportunity to fix the errors caused by mis-shapen model elements. The flood mask is replaced by a full color flood image. When floods are made, the model is examined to determine the color of the surface selected by the seedpoint. The flood is made using that color. The operator will have to override the choice of color from time to time. The strips which lie between the wireframe's object boundaries and the edge detected edges need to be colored differently than what the model will dictate, as these are the regions where the model contains error. The correct color will be found on the opposite side of the wireframe boundary. Once all the regions have been flooded, the chrominance for the flood mask is combined with the luminance of the footage to construct a colorized frame. This improved flood procedure is flowcharted in figure 8-5.

An interactive graphical tool was built for controlling the flood fills. Initially it displays the combined wireframe and edge image. The operator can sketch on the image to repair holes in outlines. A flood can be performed by specifying the seedpoint with a mouse. The operator can toggle between three views: the flood mask under construction, the colorization in progress,

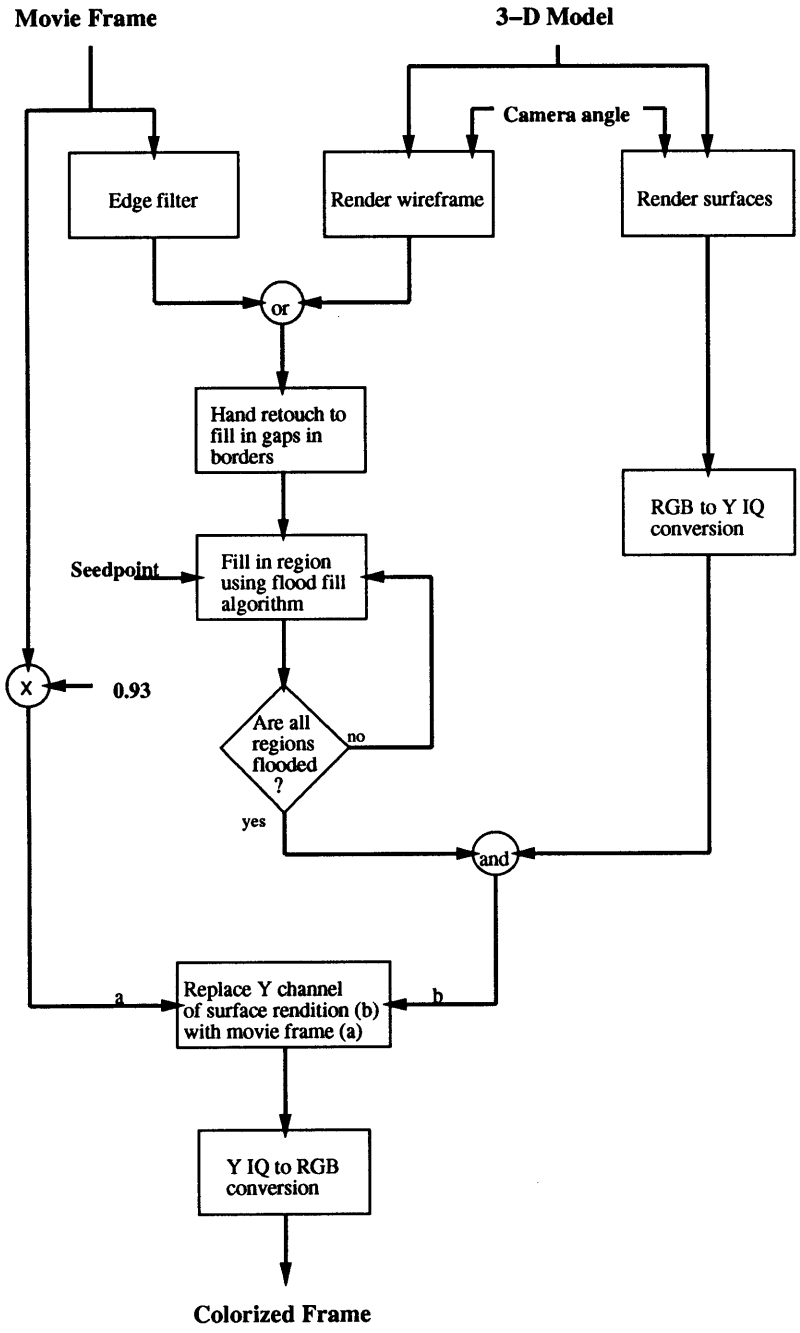


Figure 8-4: Colorizing using model and flood fills

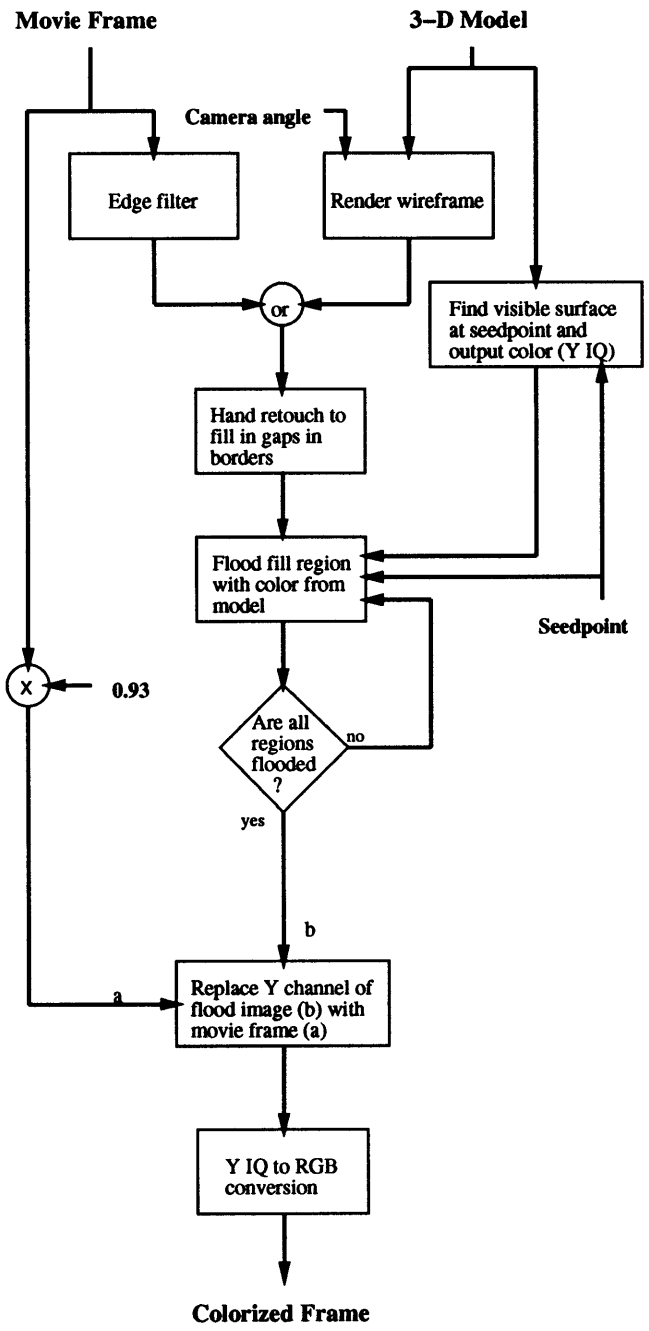


Figure 8-5: Flood fill algorithm modified to improve chrominance positioning

and a mix of the two which shows less and less of the edge image as regions are flooded and replaced with the final result. Figure 8-6 shows a black and white version of a partially flooded image in the combined view, figure 8-7 shows the output view of same in progress work. In figure 8-8, the frame has been completed.

The colorization of subsequent frames of the scene can be partly automated. The seedpoints were saved when the initial frame was colorized. A seedpoint may wind up being inside of an actor, because the actor moved. The seedpoint will have to be moved, or the actor will be flooded with color from the objects behind it. The 3-D database can be used to move the seedpoints as the camera moves. Each seedpoint is projected into the 3-D model by noting the Z value of the rendered pixel at the same (x, y) location and applying the inverse of the camera transformation matrix [34, 11]. To use the seedpoint in future frames, it is transformed back into 2-D space using the new camera transformation matrix corresponding to the updated camera position. The algorithm for calculating each successive frame, $(n+1)$, given a colorized n, is as follows:

1. Back project the seed points into the 3D model.
2. Update the camera position
3. Render a color image of the model surfaces from the new camera perspective.
4. Render a wire frame binary image from the new camera perspective.
5. Create an edge image from the current $(n+1)$ frame.
6. Combine the edge image and the wireframe with a boolean OR operator.
7. Project the seed points from the new camera position.
8. Allow the operator to shift the seed points to allow for the new positions of the shifting scene elements.
9. Allow the operator to specify additional seed points to correct for new object subdivisions.
10. Allow the operator to fix any problematic outlines.
11. Flood the edge-and-wireframe image to create a chrominance mask.

12. Scale the current frame $(n+1)$ by 0.93

13. Convert the surface rendition to Y IQ format.

14. Use the scaled frame to replace the Y channel.

15. Set the I and Q channels to 0 wherever chrominance mask is 0.

16. Convert the Y IQ image to RGB, yielding the colorized frame.

Figure 8-6: Combined view of flood in progress

Figure 8-7: Output view of a flood in progress

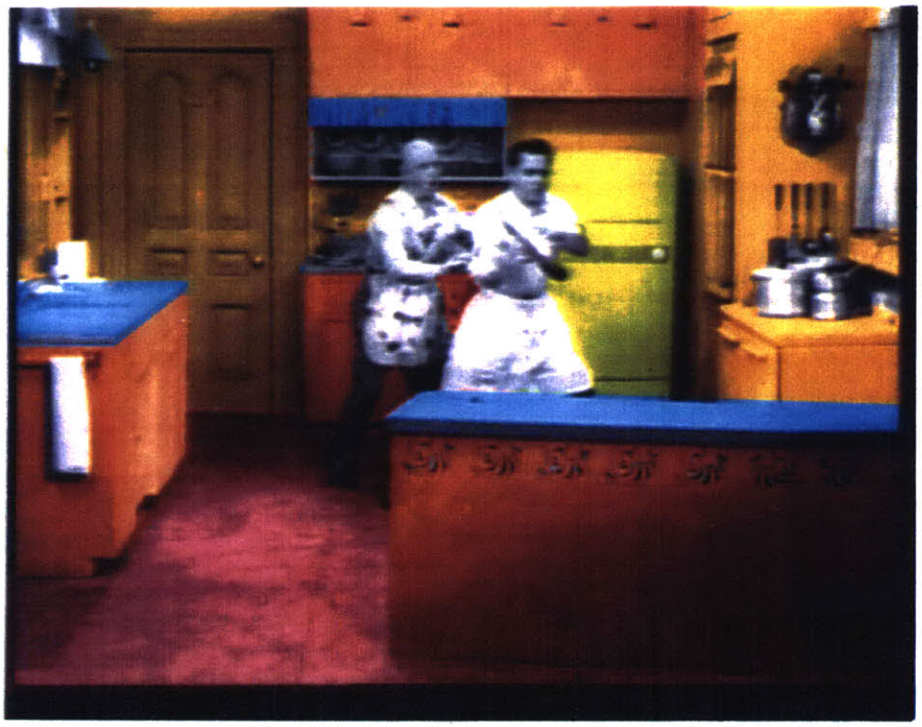


Figure 8-8: Movie frame colorized by model and flood fill

Chapter 9

3-D Model based processing techniques

Thus far I have focused on using the 3-D model to colorize the original footage. Colorization is just one of many ways the footage can be processed once a 3-D representation of the scene is available. Other possible processing include: moving the camera, changing the focal length of the lens, reducing the depth of field, changing the illumination, compositing in new elements, and low bandwidth coding.

9.1 Reprojecting the Image Using Model Depth

The three dimensional representation of the scene can be rendered from a different camera location than was used when making the footage. The model is rendered, using the original footage's camera parameters. The depth of each rendered pixel, also called the *Z buffer*, is saved. The Z buffer is used to assign a depth to the pixels from a frame of the original footage, creating a voxel representation of the frame. The voxel database can be rendered from a new camera perspective [7]. Figures 9-1 and 9-2 illustrate changes the camera's viewing position.

The processed frame has holes where surfaces were not exposed in the original. If the camera moved while filming the original footage, the hidden surfaces may have been exposed. The entire sequence of footage should be integrated into one voxel database to increase the data available for rendering the scene at different angles [18].

Figure 9-1: A frame from "I Love Lucy" is rendered from a new camera angle

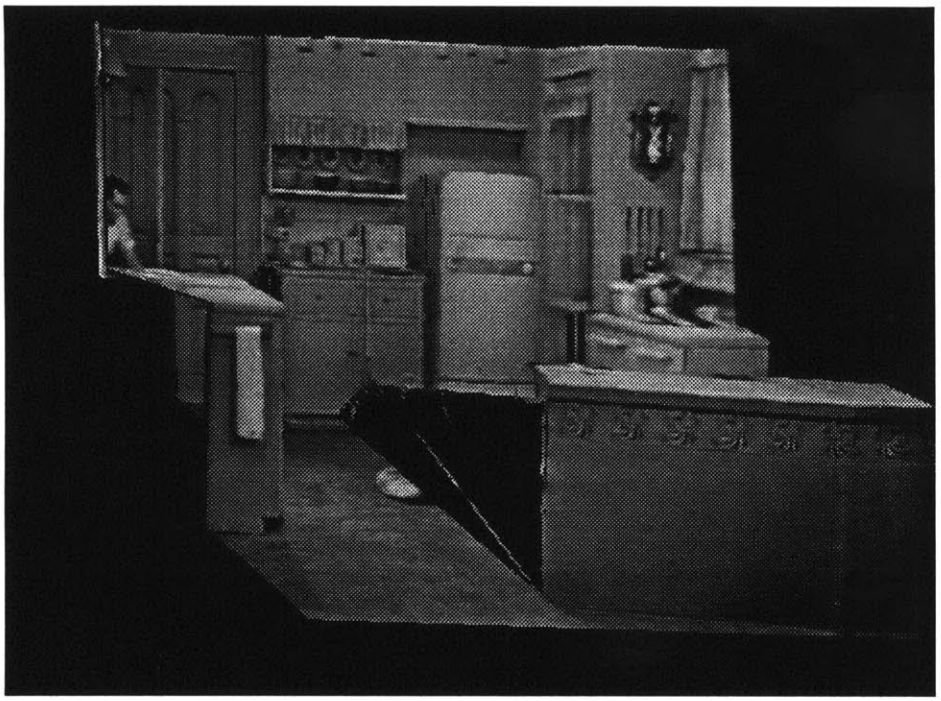


Figure 9-2: A second example of changing the camera angle

Where parts of surfaces are never exposed, texture information must be invented. This information could come from exposed sections of the same surface. Going back to the polygonal model, the hole can be associated with a particular surface. The surface is then scanned to find out if any portion of it was exposed in the original frame. If so, the hole can be tiled with a texture patch from the original frame. When no part of the surface was ever exposed, the hole is filled using surface information from the model and a surface shading algorithm. The modeler could provide a library texture to apply on top of the shading.

9.2 Changing the Lighting

The voxel representation also allows a new rendition to be made with different light sources than were present in the original. A surface normal is required for each voxel to calculate how the light bounces off, but this information is available from the structure of the model. Even if the surface normals were not available, the voxels can be differentiated to find their surface orientation [7]. Our example scene only contains objects around its perimeter, causing any but the most extreme lighting changes to be unnoticeable. Examples of lighting changes can be found in Bove's PhD thesis [7].

9.3 Changing the Focus

The focus of the lens can be altered as well. Sharpening the scene is difficult, because a blurred original does not contain the information necessary to create a crisp rendition at a high signal to noise ratio. Blurring the picture, to simulate a lens with a particular focus and small depth of field is easier. By changing the focus as a function of time, a focus pull could be added after the image was filmed. Figures 9-3 and 9-4 show the focus shift from the back wall to front counter.

9.4 Adding New Elements

Compositing objects into the scene is a straightforward task. Thanks to the model, the depth of each pixel of the original is known. The object being inserted could be a two dimensional

Figure 9-3: Lens focused on back wall with narrow depth of field

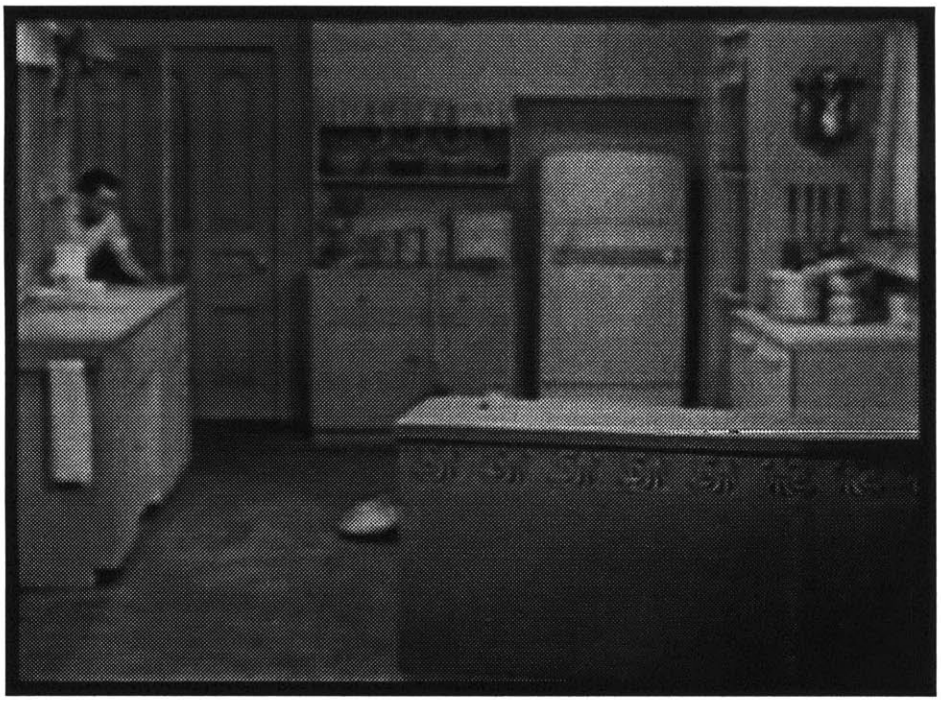


Figure 9-4: Lens focused on front counter with narrow depth of field

Figure 9-5: A Media Lab graduate student naps in the Ricardo kitchen

plane, to be inserted at a particular depth, or a three dimensional object. Those pixels from the new object which are closer to the camera then the corresponding pixels in the original are displayed in their stead.

9.5 Low Bandwidth Coding

In "Structured Video Coding" [22] a two dimensional database is created from movie footage. The database contains the background frame. The background is transmitted to the receiver once. The actors are transmitted as they move, and composited into the background at the receiver. The camera is allowed to pan during the footage. Camera pan results in a wider background as more of the set is exposed. The camera's pan vectors are sent to the receiver at a regular interval so that it is always displaying the appropriate slice of the background.

Given a 3-D database, arbitrary camera moves could be supported. The model would be transmitted once to the receiver, and thereafter only the camera parameters and actors would have to be transmitted on a frame by frame basis. The model may contain significantly more

information than the 2-D background, because of having surface textures for viewing an object from more than one camera location. If the database can not be transferred to the receiver in advance, then it could be interleaved with the foreground information. The basic model would be sent, along with surface information to form the first frame. After that, as the camera moves, the incremental surface information necessary to form a good quality rendition from the new view would be sent. If the camera moves slowly, or repeatedly traverses the same views, there will be still be a large reduction in information transmitted.

In the case of a weekly television series, there is the potential for sending the model out through an alternate distribution method. For example, the model could be published on CD at the beginning of the season. The weekly transmitted data stream would contain information as to what pieces of the model are needed, where the actors are situated, how the camera is moving, and any object information not contained on the CD, such as special props used only in the current scene. Besides cutting down on the quantity of data transmitted, there are side benefits to the viewer and broadcaster. The viewer can employ all the techniques listed above to alter the viewing experience to fit their mood and taste. The broadcaster could use distribution of the model to control access and increase revenue.

Chapter 10

Conclusion

If the model is kept simple, creating a three dimensional model by examination of two dimensional footage is not a difficult task. We could have experimented on our own contrived footage, but instead, the work on this thesis was applied to a real example: a scene from the "I Love Lucy" show. Still, the scene chosen has undoubtedly influenced the modeling procedure, as well as the direction taken in the basic research. The set was indoors, the objects familiar and for the most part, rectilinear. There was not much moving in each frame, and the camera never moved. Remove any of those constraints and much of the modeling process could break down.

A number of basic flaws in the modeling process were exposed and solutions to the flaws explored. Foremost is the problem of moving actors. A much more sophisticated modeling approach will be necessary to actually handle these actors. Actors generally require a much more detailed representation to describe their shape, they move with time, and they change shape with time. Only the first step of coping with these actors was undertaken in this thesis, that being detection.

The initial model should only be used as a hint for forming the final model. The actual shapes of objects needs to come from the image itself. Making such an adaptation was explored in Chapter 6 with good results. The method used to search the edge image to adapt the front face was primitive and should be replaced with a more robust approach, such as using snakes [16]. The simple extrusion used to recover the third dimension worked well with the type of objects modeled in our examples, but a more complex method exploiting the model's symmetry and image's surface gradient [36] should yield better results over a wider class of objects.

A great deal of the motivation for introducing a three dimensional model was to handle camera motion. A two dimensional model is not useful when the camera moves because of perspective shortening. The experimental footage did not have any camera motion, so there was no opportunity to judge the difficulty of creating a model which would be appropriate for multiple camera views. To process a moving sequence, camera tracking code needs to be developed, as well as code to integrate the texture information from multiple viewpoints.

We examined two ways of preventing incorrect colorization of actors. The first was by correlation with a background frame and the second was to colorize using flood fills on edge reduced frames. The correlation method was lacking because it failed to segment the image tightly along the actors' boundaries. It would at times lose portions of their extremities, and lump a fair amount of background along with the actors. The flood fill method does much better at finding the actors edges, but requires manual intervention on a frame by frame basis, both to fix the holes in the actors' borders and to move seedpoints that become obscured as the actors move.

A hybrid system for dealing with the actors should be explored. This system be based on the flood fill algorithm, but would use correlation with the background image to make a rough estimate as to where the actors are located. The seedpoints would be shifted automatically, shying away from areas of low correlation. The edge detection software would use the correlation coefficients to decide how much edge detail is appropriate on a block by block basis. Where the background correlates well with the movie frame, less edge detail is requested. Where the correlation is extremely poor, more edge detail is requested. At boundaries between high and low correlation coefficients, which will also be the boundaries of the actors, the most edge detail is requested.

The hybrid system could employ a motion estimator as well. First the moving objects are located using the motion estimator. The motion vectors are used to segment the image into the background and a couple of moving regions. Seed points which project within the moving regions are considered suspect and moved outside the regions. The edge image detail knob is varied, requesting increased detail within the areas zone for each moving region. Using a motion estimator has the advantage of not requiring a background frame, but has the problem that if an actor, or portion thereof, holds still, the actor will become part of the background.

Bibliography

[1] K. Aizawa, Y.Yamada, H.Harashima, and T. Saito. Model-based synthesis image coding system: Modeling a person's face and synthesis of facial expressions. *IEEE Globecom*, 1989.

[2] Ruzena Bajcsy and Franc Solina. Three dimensional object representation revisited. *IEEE Reprint CH2465-3/870000/0231*, 1987.

[3] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[4] Stephen T. Barnard. Interpreting perspective images. Technical Report 271, SRI International, November 1982.

[5] Bir Bhanu. Representation and shape matching of 3-d objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6–3:340–351, May 1984.

[6] Richard Bolt. *The Human Interface: where people and computers meet.* Lifetime Learning Publications, Belmont, CA, 1984.

[7] V. Michael Bove, Jr. *Synthetic Movies Derived from Multi-Dimensional Image Sensors.* PhD thesis, Massachusetts Institute of Technology, 1989.

[8] Susan Elise Brennan. Caricature generator. Master's thesis, Massachusetts Institute of Technology, 1982.

[9] Rodney A. Brooks. Model-based three-dimensional interpretations of two-dimensional images. In Martin A. Fischler and Oscar Firschein, editors, *Readings in Computer Vision*, pages 360–370. Morgan Kaufman Publishers Inc., Los Altos, CA, 1987.

[10] Brian M. Croll. Rendermatic: An implementation of the rendering pipeline. Master's thesis, Massachusetts Institute of Technology, 1986.

[11] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1990.

[12] W. Eric L. Grimson and Tomás Lozano-Pérez. Model-bases recongnition and localization from sparse range or tactile data. *The International Journal of Robotics Research*, 3(3), Fall 1984.

[13] H. Harasaki, M. Yano, and T. Nishitani. Background separation/filtering for videophone applications. *ICASSP*, 1990.

[14] Berhold K. P. Horn and Michael J. Brooks, editors. *Shape from Shading.* The MIT Press, Cambridge, MA, 1989.

[15] Berthold K. P. Horn. *Robot Vision.* The MIT Press, Cambridge, MA, 1986.

[16] Michael Kass, Andrew Witkin, and Dimitri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1987.

[17] Michael Leyton. A process-grammar for shape. *Artificial Intelligence*, 34, 1988.

[18] Paul Linhardt. Integration of range images from multiple viewpoints into a particle database. Master's thesis, Massachusetts Institute of Technology, 1989.

[19] Andrew Lippman. Semantic bandwidth compression. *PCS*, 1981.

[20] Wilson Markle. The development and application of colorization. *SMPTE Journal*, 93(7), July 1984.

[21] David Marr. *Vision.* W. H. Freeman and Company, New York, 1982.

[22] Patrick Campbell Mclean. Structured video coding. Master's thesis, Massachusetts Institute of Technology, 1991.

[23] P. Mertz and F. Gray. A theory of scanning and its relation to the characteristics of the transmitted signal in telephotography and television. *The Bell System Technical Journal*, 13, July 1934.

[24] T. Mizuno. Knowledge-based coding of facial images based on the analysis of feature points, motions and isodensity lines. *SPIE1199, Visual Communications and Image Processing IV*, 1989.

[25] Nicholas Negroponte, Andrew Lippman, and Richard Bolt. Transmission of presence. MIT Architecture Machine Group proposal to the Cybernetics Technology Division of the Defense Advanced Research Projects Agency, 1980.

[26] Arun Netravali and Barry Haskell. *Digital Pictures, Representation and Compression.* Plenum Press, New York, 1988.

[27] William M. Newman and Robert F. Sproull. *Principles of Interactive Computer Graphics.* McGraw-Hill Book Company, New York, 1979.

[28] Michael Orchard. Predictive motion field segmentation for image sequence coding. *ICASSP*, 1990.

[29] Don Pearson. Model-based image coding. *IEEE Globecom*, 1989.

[30] Don Pearson. Towards a theory of model-based image coding. *IEE Colloquium on low bit rate video*, 1990.

[31] Alex Pentland. A new sense for depth of field. *IEEE Transactions on Pattern Analysis and Machne Intelligence*, PAMI-9(4), July 1987.

[32] Alex Pentland. A possible neural mechanism for computing shape from shading. *Neural Computation*, 1, 1989.

[33] William H. Press, Brain P. Flannery, Saul A. Teukolsky, and William T. Vettering. *Numerical Recipies: The Art of Scientific Computing.* Cambridge University Press, Cambridge, 1986.

[34] Alvy Ray Smith. The viewing transformation. Technical Memo 84, Lucasfilm Ltd., June 1983. Revised May 4, 1984.

[35] Thomas M. Strat. Recovering the camera parameters from a transformation matrix. In Martin A. Fischler and Oscar Firschein, editors, *Readings in Computer Vision*, pages 93–100. Morgan Kaufman Publishers Inc., Los Altos, CA, 1987.

[36] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. *Artificial Intelligence*, 36, 1988.

[37] Lucas J. Van Vliet, Ian T. Young, and Guus L. Beckers. A nonlinear laplace operator as edge detector in noisy images. *Computer Vision, Graphics and Image Processing*, 45, 1989.

[38] Clea Theresa Waite. The facial action control editor. Master's thesis, Massachusetts Institute of Technology, 1989.

[39] John A. Watlington. Synthetic movies. Master's thesis, Massachusetts Institute of Technology, 1989.

[40] Peggy Weil. Computergraphic synthesis and manipulation of facial imagery. Master's thesis, Massachusetts Institute of Technology, 1982.

[41] W. J. Welsh, S. Searby, and E. Brigant. Model based coding of videophone images. *IEE Colloquium on low bit rate video*, 1990.

Acknowledgments

First, my thanks go to a few people who contributed to this thesis: Andrew Lippman, my advisor, for his ideas and direction; Walter Bender, for his proofreading, suggestions, and uncompromising commitment to excellence; Mike Bove, for his diligence as a reader, help with the pictures in Chapter 9, and above all, his encouraging words.

Thanks to my fellow students, especially the folks in the garden, for making the lab a home. Special thanks to Paddy, for the encouragement, advise, and example which kept me going to the end; Joe for being a good source of image processing equations; Mike Halle, Dave Chen, Steve Pieper, Mike Klug, and John Underkoffler, for answers and ideas which got me past many intellectual roadblocks; and Dave Small, for his friendship and last minute help.

Also, thanks to the jocks at WMBR, for picking up the slack and drinking my share while I was too busy; and to Nancy, for keeping me company, even when I had little time and no humor.