Using Video as Textural Input to a Computer Graphic Database

by

Christopher P. Thorman

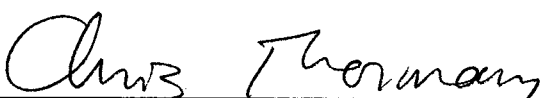
Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements
for the degree of

Bachelor of Science in Computer Science and Engineering

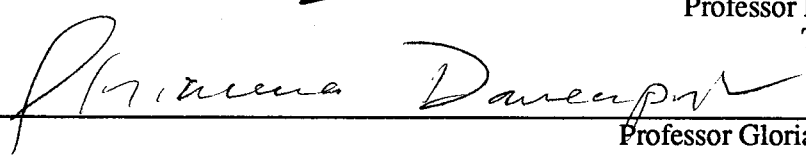
at the

Massachusetts Institute of Technology

December 1988

Author 
Department of Electrical Engineering and Computer Science
December 23, 1988

Certified by 
Professor Patrick A. Purcell
Thesis Supervisor

Certified by 
Professor Glorianna I. Davenport
Thesis Reader

Accepted by _____
Leonard A. Gould
Chairman, Department Committee on Undergraduate Theses

Using Video as Textural Input to a Computer Graphic Database

by

Christopher P. Thorman

Submitted to the
Department of Electrical Engineering and Computer Science

December 23, 1988

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Abstract

A method for the spatial tagging of images is proposed. With each image, the location, direction vector, aspect ratio, and view angle of the camera are recorded. These variables correspond to those used in the process of traditional three-dimensional rendering, and are sufficient to characterize the viewing transform performed by the camera. Because of this correspondence, spatially tagged photographs or frames of video may be treated as two-dimensional perspective renderings of their subjects. To demonstrate an application which uses spatial information, several frames of video of a building on the MIT campus were tagged with spatial data as they were generated, and then put through an inverse rendering process to create a three-dimensional database of video textures. The database was used to generate textured renderings of the object from arbitrary viewpoints.

Thesis Supervisor: Patrick A. Purcell
Title: Visiting Associate Professor of Computer Graphics, The MIT Media Laboratory

Thesis Reader: Glorianna I. Davenport
Title: Assistant Professor of Media Technology, The MIT Media Laboratory

Using Video as Textural Input to a Computer Graphic Database

by

Christopher P. Thorman

Submitted to the
Department of Electrical Engineering and Computer Science

December 23, 1988

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Abstract

A method for the spatial tagging of images is proposed. With each image, the location, direction vector, aspect ratio, and view angle of the camera are recorded. These variables correspond to those used in the process of traditional three-dimensional rendering, and are sufficient to characterize the viewing transform performed by the camera. Because of this correspondence, spatially tagged photographs or frames of video may be treated as two-dimensional perspective renderings of their subjects. To demonstrate an application which uses spatial information, several frames of video of a building on the MIT campus were tagged with spatial data as they were generated, and then put through an inverse rendering process to create a three-dimensional database of video textures. The database was used to generate textured renderings of the object from arbitrary viewpoints.

Thesis Supervisor: Patrick A. Purcell
Title: Visiting Associate Professor of Computer Graphics, The MIT Media Laboratory

Thesis Reader: Glorianna I. Davenport
Title: Assistant Professor of Media Technology, The MIT Media Laboratory

Table of Contents

Introduction 1
Recovering depth information for an image 1
Three-dimensional models and spatial correspondence 2
Describing the Spatial Attributes of Images 3
The Experiment 5
Choosing a subject 5
Choosing a coordinate system 6
Making a polygonal model of the subject 6
Obtaining spatially tagged images 8
The inverse rendering process 9
The actual inverse rendering process 10
Viewing the texture database 12
Storing the database 13
Results and Evaluation 14
A finished image 14
Database generation time and rendering time 14
Quality of interpolated video textures 15
Accuracy and sources of error 16
Suggested improvements and enhancements 18
Conclusions 18
References 19

Acknowledgements and Dedication

First, I owe a great debt to Professor Patrick Purcell for his four and one half years of support, advice, and encouragement that have made my years at MIT bearable, and have allowed me to grow academically in ways that the regular curriculum could not have allowed. This thesis is dedicated to Patrick. I also owe many thanks to Professor Glorianna Davenport of the Media Laboratory's Film/Video section. Glorianna's ideas about spatial tagging for scene description were my original inspiration to investigate this topic. Additionally, I am grateful for the time and effort Glorianna has spent as the reader of this thesis. Alan Lasky, also of the Film/Video section, donated days of his time to helping me take the photographs and gather the spatial data for the images used in the thesis project. His talents, interest and exceptionally friendly personality have made life at Film/Video fun.

Mike Liebhold of Apple Computer encouraged me to develop ideas for applications for spatially tagged video, among which is the inverse rendering application explored in this paper. Barry Haynes, also of Apple Computer, provided some useful insights into the technical issues of spatial tagging for video. Bob Sabiston, of the Media Laboratory's Visible Language Workshop, thought of the idea to use the hardware Gouraud shading capabilities of the Hewlett-Packard Renaissance graphics board to interpolate shades of color between pixels in the three-dimensional images. Dave Small, Brian Anderson, Dan Applebaum, Hal Birkeland, Sylvain Morgaine, and Laura Robin helped me to work with the computer systems at the VLW. Mary Lou Jepsen provided much moral support when I needed it the most, and also graciously loaned me a sleeping bag on a few otherwise sleepless nights.

Ben Lowengard of the Media Laboratory supplied me with some of the plans of the building used in the experiment. Mike Bove's work provided an extremely interesting example of methods for getting three-dimensional imaging information from a two-dimensional context. I owe him thanks for the time he spent describing his work to me. Ben Davis of MIT's Project Athena allowed me to use the Athena video production studio to digitize the images for this project. Brian Press, of the MIT Architecture Department, helped me write the software that allowed me to create an animation of my model.

Thanks also to Merrill W. Smith of the MIT Rotch Visual collections, for teaching me to write research papers; to Vernon Rosario, for much moral support and academic inspiration; and to Bill Coderre for being a model of wisdom, passion for excellence, and lust for life.

My wonderful parents, Mary and Paul Thorman, made it possible for me to be alive, and for me to have an MIT education. This thesis is also dedicated to them.

Introduction

This paper describes an experiment that uses spatially tagged images to add textured surfaces to a polygonal computer graphic model of a real object. Images are tagged by recording the location, direction vector, aspect ratio, and view angle of the camera. This same information is used to characterize the viewing model in the computer graphics rendering process, and allows the scanned images to be mapped onto the surfaces of the model.

There are two fields of related work that bear mention here: methods for recovery of depth information from two-dimensional images, and methods for defining spatial correspondences between two-dimensional images and three-dimensional spaces.

Recovering depth information for an image

When a two dimensional image is created by traditional photographic methods, depth information about the scene is lost. Various methods have been developed for recovering this information. Some methods involve recording the information at the time the image is created, and some attempt to infer the depth information from the content of the image by using various vision techniques.

Bove has developed a camera which records depth information for video images by using a depth-sensing camera [Bove 1988]. Both Bove and Pentland [Pentland 1987] have demonstrated the use of focusing information to determine depth. Pentland [Pentland 1988] and Brooks and Horn [Brooks & Horn 1985] have explored the use of lighting and shading cues to produce depth models. Stereographic methods have also been developed to recover depth information from two spatially similar images [Richards 1983].

The approach described in this paper is not primarily concerned with recovering depth information from flat images. Instead, its related goal is to explore the integration of those images into a three-dimensional computer model. It presumes the existence of a model which represents a real scene, and uses two-dimensional photographs of the scene to generate textured surfaces for the model.

Using this approach, the camera's imaging process is reversed, and the texture values appearing in

the photographs are projected backwards from the camera's point of view onto the surfaces of the scene's computer model. From the point of view of recovering depth information, it could be said that the model provided the depth information necessary to transform the two-dimensional images into three-dimensional ones. In another sense, the polygonal model is being "textured" or "painted" by using the spatially tagged images as sources for textural data.

Three-dimensional models and spatial correspondence

The question of spatial corresondence between the images of objects and models of the objects was addressed by Naimark in his work with developing a system for projecting images so that the location of the reconstructed image is determined by the precise location of the camera when the image was originally made [Naimark 1979]. The system allowed Naimark to create the illusion of objects whose images remained stationary in the user's environment while the projector that was producing them rotated.

Lippman's development of the Aspen Movie-Map resulted in a system in which varying sorts of data about an entire town were "accessed *spatially*, where the particular organization corresponds to the physical layout of real space." [Lippman 1980, p. 32] In the Aspen project, photographic images of the fronts of various buildings in Aspen, Colorado were mapped onto the corresponding faces in a polygonal database of the city. As one viewed the database, the two-dimensional images of the buildings appeared in their proper locations with respect to the model, providing photographic cues within the visual context of a solid polygonal model.

Eihachiro Nakame has explored the spatial correspondence between computer models of proposed buildings and the sites onto which those buildings will be placed [Nakame 1986]. Computer montage methods were used to create realistic renderings of buildings in new environments by the combination of photographic and computer-generated components into the same scene.

To demonstrate an application which uses spatial information, several frames of video of a building on the MIT campus were tagged with spatial data as they were generated. These images were put through an inverse rendering process to create a three-dimensional database of video textures. The database was used to generate textured renderings of the object from arbitrary viewpoints.

Describing the Spatial Attributes of Images

What is needed to completely describe the spatial attributes of an image? To answer this question, we will look at three-dimensional rendering methods used in traditional computer graphics applications. For a thorough discussion of this topic, the reader is referred to [Foley & Van Dam 1982].

Computer graphic rendering systems often use the metaphor of the rendering program as a camera creating an image of its model world. We imagine the renderer is a camera that can be arbitrarily positioned in space. Not only can it move to any location, it can also point in any direction, and use any angle of view. This process is known as a *view transformation.*

The goal of computer rendering, then, is to place the camera, and render that part of the world which the camera "sees".

Thus, an analogy can be drawn between the view transformation performed in traditional three-dimensional computer rendering systems and the view transformation performed by a physical camera. From this comparison a set of variables sufficient to describe the spatial attributes of a photographic image can be determined. These variables describe the location of the camera, its orientation, its angle of view, and the shape of the image.

Location. To specify the location of an object in three-space, three independent values are required. In a computer environment, the Cartesian values x, y, and z are often used, and are taken with respect to some origin. In some cases, other coordinate systems may be more useful, for instance Earth based coordinates of *latitude*, *longitude*, and *elevation* could be used.

Either an absolute or relative coordinate system may be chosen. In the experiment presented here, a local coordinate system based on Cartesian coordinates was used, but the origin of that system was determined with respect to the Earth's Latitude-Longitude system. This way, all data recorded for the project could be obtained using a project-based system, simplifying the process greatly, and if needed, the appropriate mathematical transformations could be applied to relate the model to a universally-accepted reference point.

Orientation. There are three more degrees of freedom that specify a camera's orientation.

Specifying a direction vector for the camera constrains two of these degrees, and specifying the camera's rotation about this vector constrains the third. Traditionally, *pitch* and *yaw* are used to specify the direction vector, and *roll* is used to specify the camera's rotation about this vector.

For this experiment, determining pitch and yaw without special equipment was difficult. However, it was comparatively easy to determine the location of any point in the environment by using architectural plans. So, instead of recording pitch, roll, and yaw for each image, the precise point at which the camera was aimed was recorded instead. This point was termed the *lookat* point. The vector between the camera and the lookat point defined the pitch and the yaw. Roll was constrained to 0. Pitch and yaw were determined by the following equations:

$$pitch = \sin^{-1}((z_{lookat} - z_{camera}) / dist)$$

$$yaw = \tan^{-1}(\cos(yaw), \sin(yaw))$$

where

$$\cos(yaw) = (y_{lookat} - y_{camera}) / (dist * \cos(pitch))$$

$$\sin(yaw) = (x_{lookat} - x_{camera}) / (dist * \cos(pitch))$$

and

$dist$ = the distance from the camera to the lookat point.

Aspect ratio. Most imaging devices produce rectangular images, so only one variable is needed to describe the image shape: the *aspect ratio*. This is defined as the ratio of an image's width to its height. A square frame would have an aspect ratio of 1; 35mm film has an aspect ratio of 1.5; NTSC video has an aspect ratio of 1.33; and so on. Generally, this number is a constant with respect to any given camera.

The images in this experiment were generated with a 35mm camera, so their aspect ratio was 1.5.

View angle. The angle of view of a camera is a function of the length of the lens and the camera's

format. For a camera with a fixed length lens, the view angle is constant. For cameras that can change lenses or that have a zoom lens, this value may vary with each image the camera produces. In the world of computer graphics, the angle view of a camera is the angular width of the view frustum which is defined in the view transformation process.

For this experiment, angle of view was taken with reference to the horizontal width of the camera's frame. Since only a single lens was used, this value stayed constant for all of the images.

The Experiment

Choosing a subject

The Wiesner Building at MIT, which houses the Media Laboratory and the Albert and Vera List Visual Arts Center, was chosen as a subject for the experiment. The building was accessible and interesting. Unobstructed views were possible, for the most part, because large open spaces surround the building on two sides. It would be fairly easy to create a wireframe model from plans of the building because it was fairly geometric in design, with easily determined three-dimensional coordinates.

In determining the spatal data for the experiment, it was assumed that accuracies as close as 1% would be needed for any subject. Clearly, using a building as a subject allowed for this needed accuracy.

Related work at the Media Laboratory has explored the use of spatial correspondence techniques similar to those developed in [Nakame 1986] to generate sequences of motion video in which a computer rendered model of a building was rendered over images of an empty site [Purcell 1987]. The choice of an architectural subject was especially pertinent to research already being conducted at MIT.

Choosing a coordinate system

As mentioned earlier, a Cartesian system based on a local origin was chosen. The axes of the system were chosen to line up with the main structural lines of the Wiesner Building, so that the x, y, z origin was at one corner of the building, on the ground at the level of the plaza surrounding the building.

The choice of directions for the x, y, and z axes was a bit more complicated. Since the database was to be rendered using the Hewlett-Packard Starbase graphics library [Hewlett-Packard 1985], the handedness of the coordinate system was chosen to match the one used by Starbase: a left-handed one. Since by convention, the z direction is often the upward direction, it was decided to make z go up, away from the Earth, and x and y were placed so that the majority of the building was located in the first x-y quadrant, resulting in coordinates that had mostly positive numerical values. This resulted in an x axis that increases eastward, and a y axis that increases southward with respect the the building site.

Making a polygonal model of the subject

A polygonal model of a three-dimensional object is nothing more than a description of a set of polygons in three-space that represent the outer surfaces of the object. A representation convention for polygonal objects has been developed at Ohio State University [Crow 1982]. In the OSU standard, polygonal objects are represented as a set of points and a set of faces, where faces are defined as ordered lists of the points composing them. The OSU standard format unfortunately does not encourage human-readable files, so it was enhanced for this project to include comments and blank spaces at arbitrary locations in the files. This extended OSU format was used to create the polygonal model of the Wiesner Building.

To make a polygonal model, plans of the building were first obtained. Then, the points located at interesting positions on the surface of the building were assigned unique numbers, and entered into the description file for the building. Next, each face of the building was described as a sequence of numbered points and added to the description file. Care was taken to define each face with its points listed in a counterclockwise direction around the perimeter of its "front" side. This way, rendering programs can reliably determine the front side of each face, and generate shaded solid models of the building using only the polygonal data.

Rounded portions of the building were approximated by modeling each 90-degree curved section of the building as three flat faces. If a more advanced representation system had been in use, these curved sections might have been represented as curves, but for the purposes of this experiment, flat approximations to curved surfaces were adequate.

Finally, a parser was written to read the data file that was created. Two applications were developed to view the data by moving around the model in real time, one that allowed a user to examine wireframe views of the model, and one that rendered the model with solid, shaded faces. Extensions were written to allow for fly-by style animations of the scene to be generated interactively by the user. These applications were particularly helpful in verifying the correctness of the data files. Figure 1 shows two views of the Wiesner Building polygonal model using each of these programs.

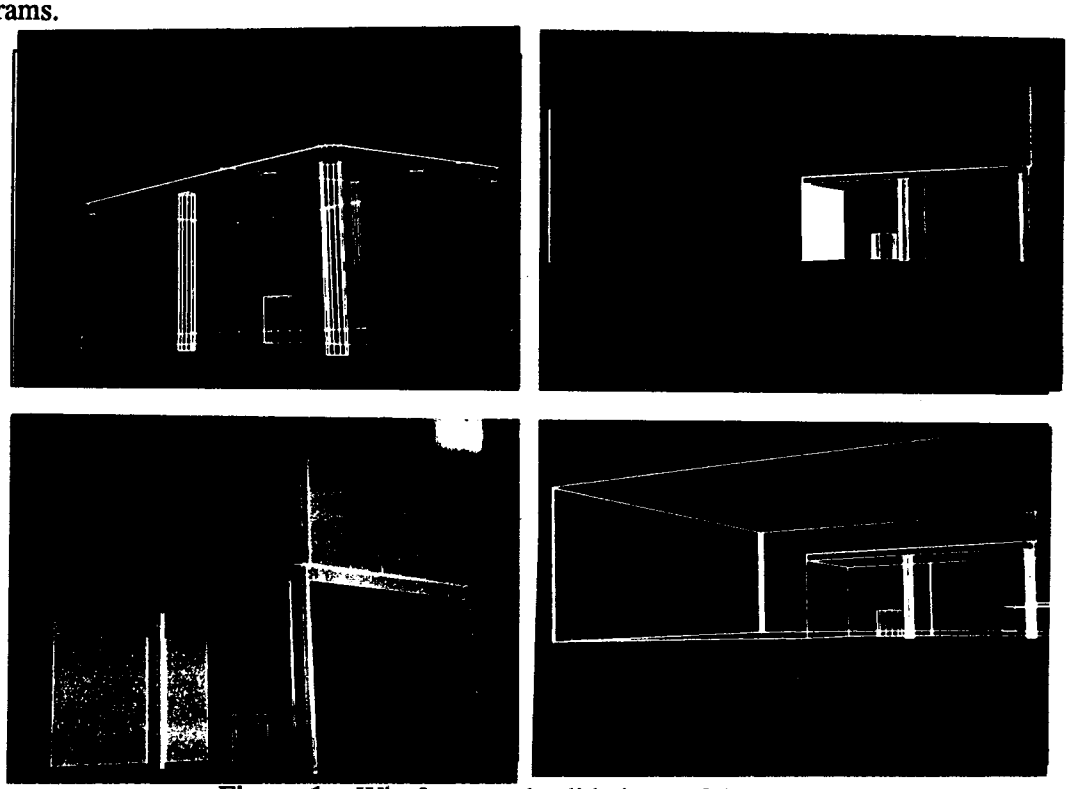


Figure 1: Wireframe and solid views of the model.

Obtaining spatially tagged images

An advantage of the Wiesner building as a subject is that its entire exterior is composed of white tiles with dark grid lines between them. This fact made the determination of *lookat* points exceptionally easy; in most cases, they could be recorded by visual inspection. Additionally, the plaza around the building is covered with tiles that are laid in even increments of feet and inches, which made the measurement of positions around the building easy and accurate.

As each image was taken, its spatial attributes were recorded in a field notebook: the x, y, z position of the camera, and the x, y, z values of the *lookat* point. As mentioned above, the view angle and aspect ratio were known constants since a single camera with a fixed lens was used.

A 35mm camera with an architectural ground glass and tripod was used to create the images, because it afforded easy positioning, and easy determinination the precise center of the image. A video camera was also tested, but the unreliability of the electronic viewfinder caused the task of centering the image to be all but impossible.

All of the images were converted into NTSC video, digitized, and then translated into RGB (Red/Green/Blue) format. Each image was then corrected to its original 1.5 aspect ratio, which had been lost during the NTSC stage, and finally combined with its view description file to produce 23 spatially tagged video images.

Back in the lab, the spatial data for each image was entered into a view description file. A parser was written which converts view description files into a computer representation of the camera's position. An example of a view description file is shown in Figure 2.

```
% File: s5.view
% Purpose: The View from slide position 5

% This is a spot near the Henry Moore Sculpture

x 232           % camera x in feet.
y -104          % camera y in feet.
z 4.5           % camera z in feet.

% We're looking at a point in the center of the East Wall.

i 104           % x of the point being looked at
j -2            % y of the point being looked at
k 33.1          % z of the point being looked at

a 1.5           % aspect ratio: width to height: 35mm film
v 58            % view angle of the camera in degrees
```

Figure 2: A view description file.

The inverse rendering process

This section describes the inverse rendering process proposed by the author.

Given a set of images, and a set of faces, the inverse-rendering algorithm considers every possible image/face pair <I, F> to determine if the face theoretically appears in the image. If it does, then the image and all of its pixels are projected onto the plane of the face. Then, only the pixels which fall within the face in question are kept, and added to the model's surface.

Running this algorithm for a polygonal model and a set of spatially tagged images of that model will result in the maximal possible textural coverage of the model by the given set of images. Potentially, the algorithm could be modified to discard pixels that get projected onto a location which is already covered, and to avoid checking fully-covered faces at all.

Figure 3 shows a pseudocode description of the algorithm.

```
for each image I in the set of images:
    for each polygonal face F in the model:
        if face F is not facing the camera, or F is obstructed,
            then:
                ignore F for this image.
            else:
                project the rectangular plane of I onto the plane of F.
                for each pixel P in I:
                    project P into the plane of F.
                    if pixel P falls outside face F,
                        then:
                            discard P.
                        else:
                            add P to the model -- it is part of F's texture.
```

Figure 3: The inverse rendering algorithm.

The actual inverse rendering process

The algorithm given in the previous section is theoretical. In practice, it was modified greatly to allow for limitations of hardware, memory size, disk space, and processor speed, and difficulty of implementation.

First, the time and space orders of growth of the given algorithm are extremely large. The order of growth is the number of images multiplied by the number of faces in the polygon, multiplied by the number of pixels in each face.. In this experiment, there would have been approximately 2,000 image-face pairs to consider (approximately 20 images multiplied by approximately 100 faces).

Rather than consider every <I, F> pair, a subset of them was manually chosen. This was done by inspecting each image visually and deciding which of the faces in the model would be most appropriately textured by that face, and arranging the choices so that each face would not be textured by more than one face. Faces that could not be adequately covered by any of the images were ignored altogether. This way, the total number of <I, F> pairs was reduced to less than the number of faces in the model.

One action performed by the algorithm is the determination of which pixels in an image I are part of face F. This is effectively three-space scan conversion. There are two ways in which this process might be performed. The first would be to use techniques similar to ray-tracing to see if the line of projection of the pixel onto the plane of F ''misses'' F, and to discard the pixel if it does.

An alternative would be to render face F onto a flat projection, and use the projection of the face to select the pixels in I that should fall in F. It was this second approach which was taken. By using a manual process (which could also be automated), the pixels that were to be part of each candidates for each face were selected from the base image by the creation of a binary mask, which eliminated those pixels which clearly did not belong to the face in question.

In this way, the theoretical algorithm was reduced to the actual algorithm shown in Figure 4.

```
for each image/face pair <I, F>:
    for each pixel P in I:
        if pixel P is not in the face's mask,
            then:
                ignore P.
            else:
                project P into the plane of F.
                add P to the model -- it is part of F's texture.
```

Figure 4: The inverse rendering algorithm that was used.

The algorithm for projecting each pixel in the image involves tracing the ray of light that created the pixel back from the camera and onto the face from which the ray came. The two-dimensional pixel is then assigned three-dimensional coordinates based on its location on the face.

Mathematically, the camera is positioned in the model space in its original position, and a two-dimensional rectangle describing its original aspect ratio is generated. This rectangle is then run through an inverse of the viewing transformation, producing a rectangle floating in three-space, somewhere along the camera's view pyramid. Then, this rectangle and any pixels within it can be easily projected onto the plane of any chosen face in the scene, simply by determining the plane in which the face lies, and applying a formula to determine the intersection of the line of sight with the plane.

Viewing the texture database

Up to this point, discussion has been devoted to the projection of pixels into three-space, but not to their subsequent viewing. Because the pixels are nothing more than dimensionless dots of color whose original purpose was to appear as single points of glowing phosphor, surrounded by niegh-boring pixels, the question of how to treat pixels that have been projected into three-space is an interesting one.

Bob Sabiston of the Media Laboratory's Visible Language Workshop uses a method which takes advantage of hardware-level color interpolation for arbitrarily-shaped polygons, in order to allow the user to rotate a full color bitmap in three-space in real-time [Sabiston 1988]. The hardware that implements this ability was originally designed for the purpose of Gouraud shading, in which colors on the surface of polygons in a three-dimensional model are shaded by interpolating smoothly between the values of the colors at the polygon vertices [Gouraud 1971]. This hardware, produced by the Hewlett-Packard Corporation for its Series 300 workstations, proved useful for the rendering stage of this experiment.

Rather than simply projecting individual pixels from the image, square "tiles" were projected instead. Each tile is a square polygon which is assigned a different color at each of its corners -- one for each of four neighboring pixels. So, it was actually these tiles that were projected onto the faces of the model. When a tile is smooth-shaded to interpolate between the pixel colors at its vertices, the illusion of a continuous textured surface is created.

In the applications developed as part of this work, the rendering of faces consists of rendering all of the tiles that have been projected onto it. The hardware smooth shading interpolates shades of color between the common vertices of the tiles.

Storing the database

Currently, the inverse rendering process is implemented as two phases. The first phase is the creation of the three-dimensional textures, and the storage of the tiles on disk. In the second phase, the database is rendered: the user chooses an angle of view for the scene, and then the application places each precalculated tile in the appropriate position in the image.

Each tile is composed of four points, and each point has the floating-point values, x, y, z, and r, g, b associated with it. This means that each tile requires 24 floating-point values of storage space, for a total storage requirement of 96 bytes. The large size of the tiles was one of the chief arguments against storing any more than a certain number of them at a given time, and severely limited the size of the texture database, as well as the speed of rendering.

At 96 bytes per tile, a source image of 500x400 pixels requires over 18 megabytes of process memory to manipulate all at once, and as much disk space just to store it. In order to allow images as large as this, and whole databases of arbitrary size, an internal swapping system for tiled textures was developed so that in-memory data structures could at the very least refer to a large number of objects and access them from disk. In the end, then, the main limit on the storage space available to the model generation program was the amount of disk storage space available on the machine.

For the development stages of the project, a workaround to the size problem was found by downsampling the source image by factors of 4, 9, or 16, depending on the amount of space reduction that was needed. Of course a loss of detail was noticeable with downsampled images; however, the use of the interpolated textures did much to make even these downsampled images visually adequate.

For the model that was finally generated, a swap space of about 100 megabytes of disk space was available. This amount of space was adequate to allow for the texturing of a significant portion of the surface area of the model, though not for all of its faces.

Results and Evaluation

A finished image

Figure 5 shows an image that was generated as part of this experiment.

Figure 5: View of the textured model.

Database generation time and rendering time

The time required to perform the inverse-rendering to produce the database from the raw images was broken down roughly into the following categories: about 15% reading the raw file from disk, 10% calculations, 75% writing the resulting data out to disk again. Clearly, a more powerful machine with faster, larger disks and significantly more core memory could have cut the inverse rendering times by orders of magnitude. Specialized hardware optimized for the purpose could easily yield a few more orders of magnitude improvement.

The rendering time for the Gouraud-shaded pixel tiles was about 1,000 tiles per second, when the majority of the tiles were small, and all of them were in memory on the host computer. Adding in swapping time could increase the time required to render a single frame of the database to as much as three minutes for a texture database with more than 20 megabytes of data. If enough disk space

were available to hold a fully-textured database of the model, where all textures were sampled at the maximum available resolution, rendering times could have been as long as 15 minutes per image.

Quality of interpolated video textures

Overall, the quality of the textures was quite high. The illusion of a continuous surface was extremely strong, even once aliasing effects became noticeable. Figure 6 shows a detail of one section of the wall. The figure shows a section of an image approximately 200 pixels across.

Figure 6: Quality of interpolated textures.

Unfortunately, the problem of aliasing of the textures along the edges was extremely noticeable. This effect was caused by the aliasing effects that are inherent in increasing the size of digitized images, but was particularly noticeable when jagged edges of textured surfaces were greatly increased in size during the projection process. Figure 7 shows a detail of two edges demonstrating different frequencies of jagged edges.

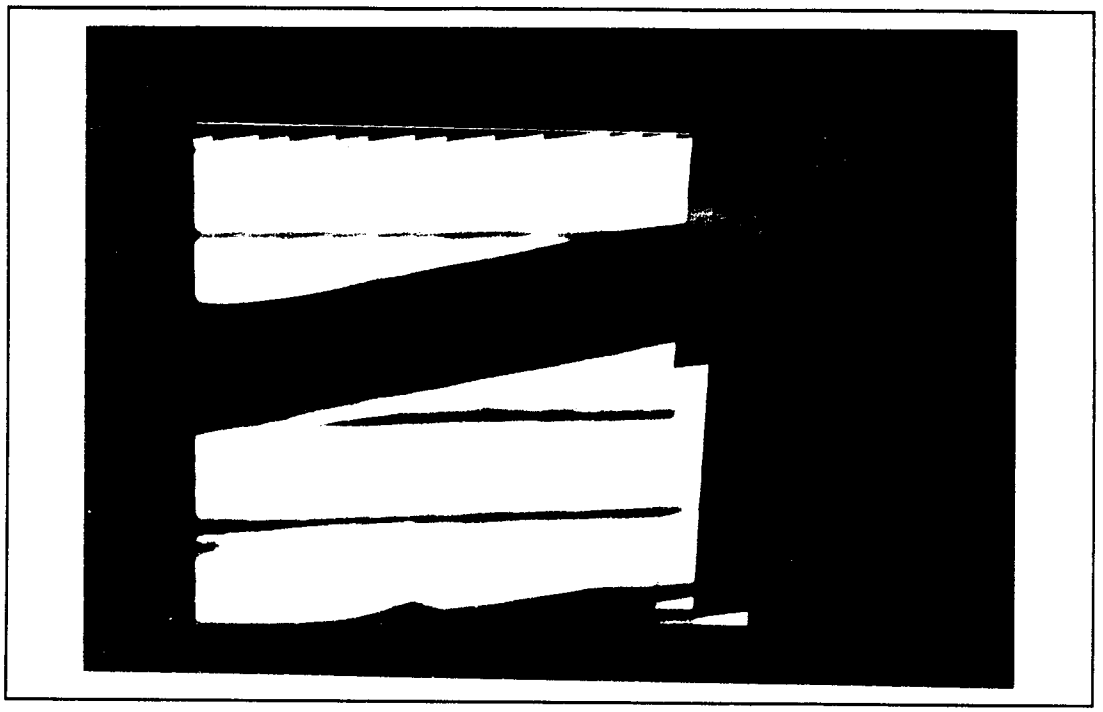


Figure 7: Jagged edges of textured surfaces.

Accuracy and sources of error

The accuracy of the wireframe model of the building was extremely high, since the data was taken from the plans for the building. The most inaccurate portions of the model occurred in rounded portions of the building, since these portions were approximated with flat surfaces. The effect of these approximations not was great enough, however, to cause significant aberration in the inverse-rendering process.

As stated earlier in the paper, a large subject was chosen so that measuring the spatial location of the camera would be subject to very small errors. This was indeed the case, as demonstrated by the ability to very accurately replicate the real camera's views using computer renderings of the scenes.

Since the images for the experiment were generated from 35mm slides, the biggest source of error, much larger than any other, was in the registration (vertical and horizontal alignment) of the images during the photographing, slide mounting, and digitizing processes. When the film is exposed by the camera, it is difficult to guarantee that the image that hits the film has the same framing as in the viewfinder. When the slides are placed in their mounts during processing, ac-

curacy of registration is impaired once again, and finally, the digitizing process is also subject to large errors in registration. These errors were visible when the texture for a face did not line up perfectly with the face after the projection process had been completed. Figure 8 shows a detail of a textured surface that is slightly rotated with respect to its face.

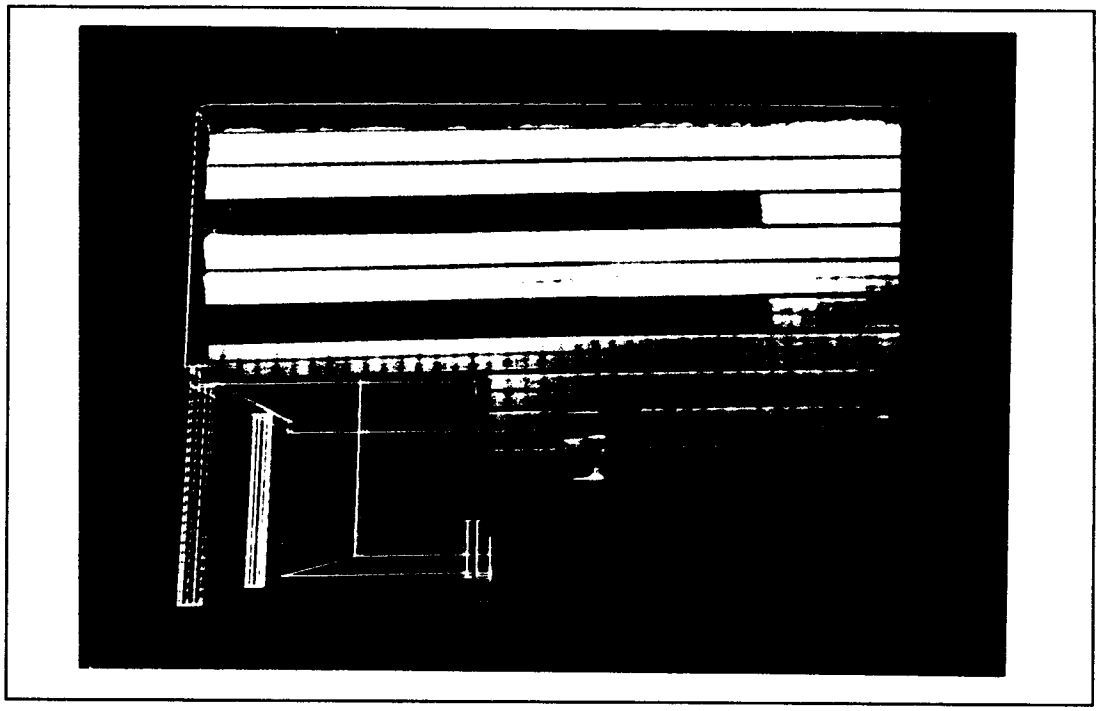


Figure 8: Registration error.

Another source of error arises from the fact that making images of the model is inherently subject to the current weather and lighting conditions at the time. For our first batch of images, the weather contiditons were bright and sunny, which caused problems with uneven lighting among the faces of the building, as well as shadows on certain walls. One way to partially compensate for this problem is to shoot the images on a day with uniformly-overcast skies, so that all surfaces are lit evenly.

Probably the most significant problem with this process is that objects such as signs, poles, trees, etc., which are not part of the wireframe model, may partially obstruct portions of the object when the photographs are taken. This process then projects images of objects onto the surface of the subject, as part of the three-dimensional texture. Flattened images of trees and power lines appear as part of the textured model.

Suggested improvements and enhancements

Since storage space was a problem for the system, it would be possible to sacrifice rendering speed to save disk space. At the time a database is generated, a record of the view transformations could be stored instead of the transformed textures themselves. This way, many of the calculations could be cached, and the textures could be recalculated as necessary.

Conclusions

Spatial tagging of images is an effective way of describing an image in a three-dimensional context. This fact has been demonstrated by the use of spatial information to retrieve texture information from two-dimensional images made by real cameras, and then combining this information with a corresponding three-dimensional computer model.

This experiment has illustrated one motivation for adding spatial attributes to images: the use of video images in computer graphic modeling, but there is a whole array of potential applications for spatially tagged film and video.

For instance, architects could use three-dimensional video models of buildings to be changed or renovated and view the effects of the renovations or changes on a realistic model. Work at the MIT Media Laboratory Film/Video Section is currently exploring the use of spatial data for moviemaking, scene description, and camera motion description. Automated editing systems can also benefit from the addition of spatial data to film sequences [Davenport 1988].

As we begin to see demand for film and video imagery which is more integrated with computer environments, we will see attention turning to the development of a spatial encoding standard, and to the creation of hardware and software tools to make spatial tagging of images commonplace.

References

[Bove 1988] Bove Jr., V. Michael, *Pictorial Applications for Range Sensing Cameras*, MIT Media Laboratory, Cambridge, Massachusetts, 1988.

[Brooks & Horn 1985] Brooks, Michael J., and Horn, Berthold K. P., *Shape and Source from Shading*, MIT Artificial Intelligence Laboratory Memo number 820, January 1985.

[Davenport 1988] Davenport, Glorianna. Current work at the MIT Media Laboratory is exploring interactive video, intelligent video editing systems, and automated descriptions of camera positioning.

[Eihachiro 1980] Eihachiro, Nakame, et al., *A Montage Method: The Overlaying of the Computer Generated Images onto a Background Photograph*, Proceedings of SIGGRAPH 1986, Dallas, Texas, Volume 6, Number 4, page 207.

[Foley & Van Dam 1982] Foley, James D. & Van Dam, Andries, *Fundamentals of Interactive Computer Graphics*, Addison Wesley, Reading, Massachusetts; 1982.

[Gouraud 1971] Gouraud, J., *Continuous Shading of Curved Surfaces*, IEEE Transactions on Computers, June 1971.

[Hewlett-Packard 1985] Hewlett-Packard Corporation, *Starbase Reference Manual*, 1985.

[Lippman 1980] Lippman, Andrew, *Movie-Maps: An Application of the Optical Videodisc to Computer Graphics*, Proceedings of SIGGRAPH 1980, Seattle, Washington, Volume 14, Number 3, page 32.

[Naimark 1979] Naimark, Michael, *Spatial Correspondence: A Study in Environmental Media*, MIT Department of Architecture Master's Thesis, Cambridge, Massachusetts, 1979.

[Pentland 1987] Pentland, Alex, *A New Sense for Depth of Field*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9, No. 4, July 1987.

[Pentland 1988] Pentland, Alex, *On the Extraction of Shape Information From Shading*, MIT Media Lab Vision Sciences Technical Report 102, March 1988.

[Purcell & Davenport 1987] Work by Patrick Purcell, Glorianna Davenport, and L. Clifford Brett explored the ability to superimpose computer renderings of a

building onto individual frames of video sequences so that an illusion could be created of driving past a site on which the building was located.

[Richards 1983] Richards, Whitman, *Structure from Stereo and Motion*, MIT Artificial Intelligence Laboratory Memo number 731, September 1983.

[Sabiston 1988] Current work at the MIT Media Laboratory's Visible Language Workshop is exploring user interaction in animation systems, and uses hardware gouraud shading to allow real-time manipulation of two-dimensional bitmaps in space.