A Visualization of the MIT City Scanning Project

by

Tara B. Schenkel

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

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1998

Author .....
Department of Electrical Engineering and Computer Science
September 14, 1998

Certified by .....
Seth Teller
Associate Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

A Visualization of the MIT City Scanning Project

by

Tara B. Schenkel

Submitted to the Department of Electrical Engineering and Computer Science
on September 14, 1998, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes the production of a computer animated visualization of the MIT City Scanning Project. The main goal of the City Scanning Project is the implementation of an automated, end-to-end system to construct three-dimensional (3D) models of urban areas. The Project encompasses both the acquisition of digital images and navigation data in the region to be modeled and the development of algorithms that use the collected data to reconstruct a 3D model of the region.

Through narration and animation, the visualization presents a comprehensive overview of the Project, including its motivation, methods, results, and future. This paper describes both the creative and technical aspects involved in the design, animation, rendering, and recording of the visualization.

Thesis Supervisor: Seth Teller
Title: Associate Professor of Computer Science and Engineering

Acknowledgments

I'd like to thank everyone who helped me with this project, especially Professor Teller, for suggesting such an enjoyable thesis project; Satyan Coorg and Neel Master, for explaining their work on the City Project to me; Bryt Bradley, for reading (several times) the narration for the visualization; Jesus Orihuela for recording some excellent grumbles; and Nick Papadakis, for helping me with every aspect of recording the visualization to tape. I'd also like to thank everyone in the City Scanning Group and the Computer Graphics Group.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis describes the design and production of a visualization of the MIT City Scanning Project. The objective of the MIT City Scanning Project is to automate construction of realistic three-dimensional (3D) computer models of urban environments [Tel97]. Accurate models of the real world are useful in such various applications as architectural visualizations, urban planning, and entertainment. Yet the construction of almost all such models currently requires significant human input [DTM96], [JLF95], making modeling on a large scale— such as a city the size of Cambridge— impractical.

The City Scanning process, which takes images of a city as input and generates a textured, 3D CAD model of that city as output, is a complex one involving many steps. The visualization of the Project is intended to provide its audience with a comprehensive overview of these steps, to convey a basic understanding of the Project, and to promote a deeper interest in the various aspects of the Project.

The Project's approach consists of acquiring digital images in the region of interest, annotating them with navigation data, and then applying computer vision techniques to the data to produce an accurate textured CAD model of the

region [CMT98]. To automate the data acquisition process, we've built a specialized camera platform [DeC98], named Argus after the hundred-eyed creature of Greek myth. Argus is a manually-propelled, wheeled platform. Its digital camera is attached to an extensible stalk and motorized pan-tilt head which allow the camera six degrees of freedom. Argus is also equipped with a navigation system, consisting of an inertial measurement unit (IMU), Global Positioning System (GPS) antenna, wheel encoders, compass, and inclinometer. These components provide the measurements necessary for annotating each image with the position and orientation (pose) of the camera at the time the image is taken. The pose data is used in subsequent image processing algorithms.

Image processing begins by blending all the the images taken from the same camera position into a single, hemispherical image. These "spherical mosaics" are then positioned in a global coordinate system, producing what we term pose mosaics [CMT98]. Pose mosaics are then used in an algorithm that identifies vertical facades to compute a textured 3D CAD model of the region [CT98].

Currently, we have acquired a collection of 81 nodes, sequences of images taken from a single camera position. These nodes are located in and around MIT's Technology Square, the office complex in which our lab is located.

The visualization incorporates information from several documents [Coo98], [CMT98], [CT98], [DeC98], [Tel97] into one comprehensive presentation. In addition to collecting information from several sources, the visualization, because of its ability to present motion, is also able to provide a more illuminating treatment of certain aspects of that information. For example, where a paper reads, "The camera is mounted on a motorized pan-tilt head which allows us to automatically capture almost a full hemisphere of imagery at each acquisition location" [DeC98], the visualization can show the motion of the camera on the pan-tilt head and sim-

ulate the acquisition of the hemisphere by tiling each image at its corresponding position.

The visualization was designed to fulfill the following objectives:

- to accurately present the motivation, goals, approaches, and results of the MIT City Scanning Project in an engaging and informative manner.
- to make effective use of animation techniques and take the best possible advantage of the audio and visual nature of video.
- to strike an appropriate balance between depth of technical material and ease of comprehension that is appropriate to an audience with a background in engineering.
- to deal creatively with the problems of visually portraying events and topics that do not have obvious visual parallels.

The remainder of this paper discusses the design and production of the visualization. Chapter 2 explains how related work affected our visualization. Chapter 3 describes the software tools used in the production of the visualization. Chapters 4 and 5 describe the audio and video portions of the movie. Chapters 6, 7, and 8 detail the rendering, compiling, and recording of the visualization, respectively.

Chapter 2

Related Work

Computer animations of algorithms and other scientific topics are becoming increasingly popular. Hausner and Dobkin argue in [HD] that visualizations are an important tool in understanding algorithms, and that every paper describing a geometric algorithm should be accompanied by an animation of the same. Animations depicting concepts ranging from simple mathematical equations and formulas to complex algorithms abound. The *Project Mathematics!* series [Apo92] combines live action video, narration, music, and simple computer animation to visualize a high-school level treatment of such topics as the Pythagorean Theorem, pi, and sines and cosines. At a more complex level, animations from the Geometry Center, which treat such topics as inversion of a sphere [LMM94], knot theory [EGea91], and toroidal space [LMTea], are both aesthetically and technically well-crafted and served as an inspiration for our visualization.

Although much work has been previously done in the area of animating mathematical, algorithmic, and other concepts, our visualization is obviously unique in its content. Previous work did, however, provide insight into the techniques, conventions, and aesthetics of creating such an animation: for example, Blinn's

notes describing the animation process of the *Project Mathematics!* series [Bli95], which include design decisions and discussions of desired effects of animations, were useful in planning the animation for the City Project visualization.

Chapter 3

Tools

We used many tools in the creation of the visualization. We used both Open Inventor and Alias Studio to construct and animate the 3D models seen in the visualization. Alias Studio was also used to render each of the frames of the visualization. We then used Adobe Premiere 4.2 to arrange the individual frames into an animation and to combine the animation with the narration.

3.1 Open Inventor

Open Inventor provides a programming interface for modeling and animating 3D geometry [Wer94]. It provides very few geometric primitives, although other shapes— such as the torus needed to model the wheels on the Argus platform— are easily constructed with C++ calls to the Open Inventor API (Application Programming Interface). Open Inventor also provides "engines," C++ classes that can be used to animate models: the output from an engine that translates the Argus model forward is connected to another engine that rotates the wheels appropriately as the platform moves forward.

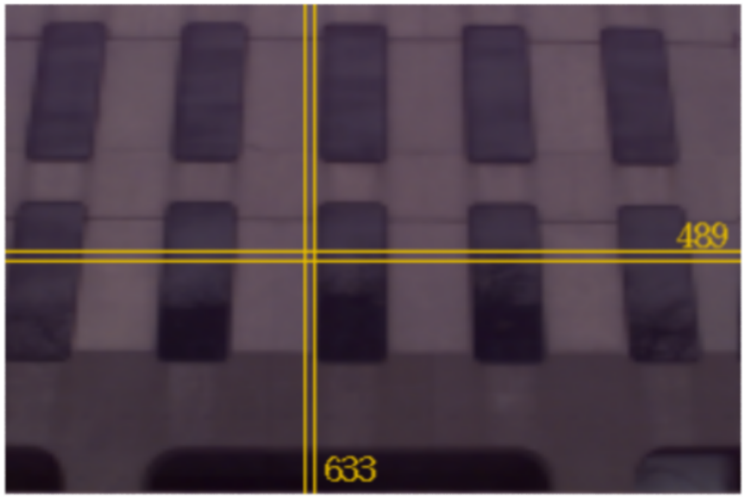

Figure 3-1: An image is simply a two-dimensional array of pixel values.

Although Inventor doesn't offer an interactive modeling and animating environment as does Alias, it does provide a relatively simple programming interface. In fact, when we needed to render the sequence of images that show a cross-hair moving over an acquired digital image while also displaying the x and y coordinates of the pixel at the intersection of the cross-hairs, we discovered that there was no simple way of doing so in Alias. Writing a few dozen lines of Inventor code produced a sequence of images that we then imported into Alias to superimpose over the texture on the picture geometry (Figure 3-1).

The Argus platform was originally modeled and animated in Inventor. We also developed two separate applications using the Argus model and the Open Inventor library. One application displays the 3D model of the platform and allows the user to view the platform from different angles by manipulating the mouse and to move the dynamic components of the platform: the pan, tilt, and extension of the camera frame, by adjusting several sliders. The second application simulates

the platform's data acquisition process. It reads the pose data files associated with each image and recreates the motion that Argus followed in acquiring the images.

3.2 Alias Studio

3.2.1 Modeling and Animating

After planning other scenes of the visualization, though, we decided to switch to Alias to complete the rest of the modeling and animation. The storyboard for the visualization included a great deal of animation, which we decided would be easier to create with Alias.

Alias is a sophisticated 3D modeling and animating tool [Ali98a]. It allows the user to key-frame objects: the user specifies a position for the object at time x, and another position at time y, and Alias interpolates motion for the object between the two given positions within the given timeframe. Alias also lets the user specify "expressions," that can be used to link one action to another. For example, if the translation of one object is key-framed, another object can be made to follow the same path by setting its translation expressions to refer to the translation of the first object [Ali98b].

Alias also provides a method of animating what it terms "shaders": the color or texture applied to a piece of geometry. For example, this capability allowed us to make certain pieces of geometry glow in order to draw attention to them. We employed a more complex use of animated shaders to visualize the painting of the buildings in the video's introduction. The "paint" is actually a rectangle that is situated slightly in front of the building surface. As the painter moves his paint roller along the surface, the rectangle grows larger, and the coordinates that specify which portion of the texture should be applied to the geometry change (Figure 3-

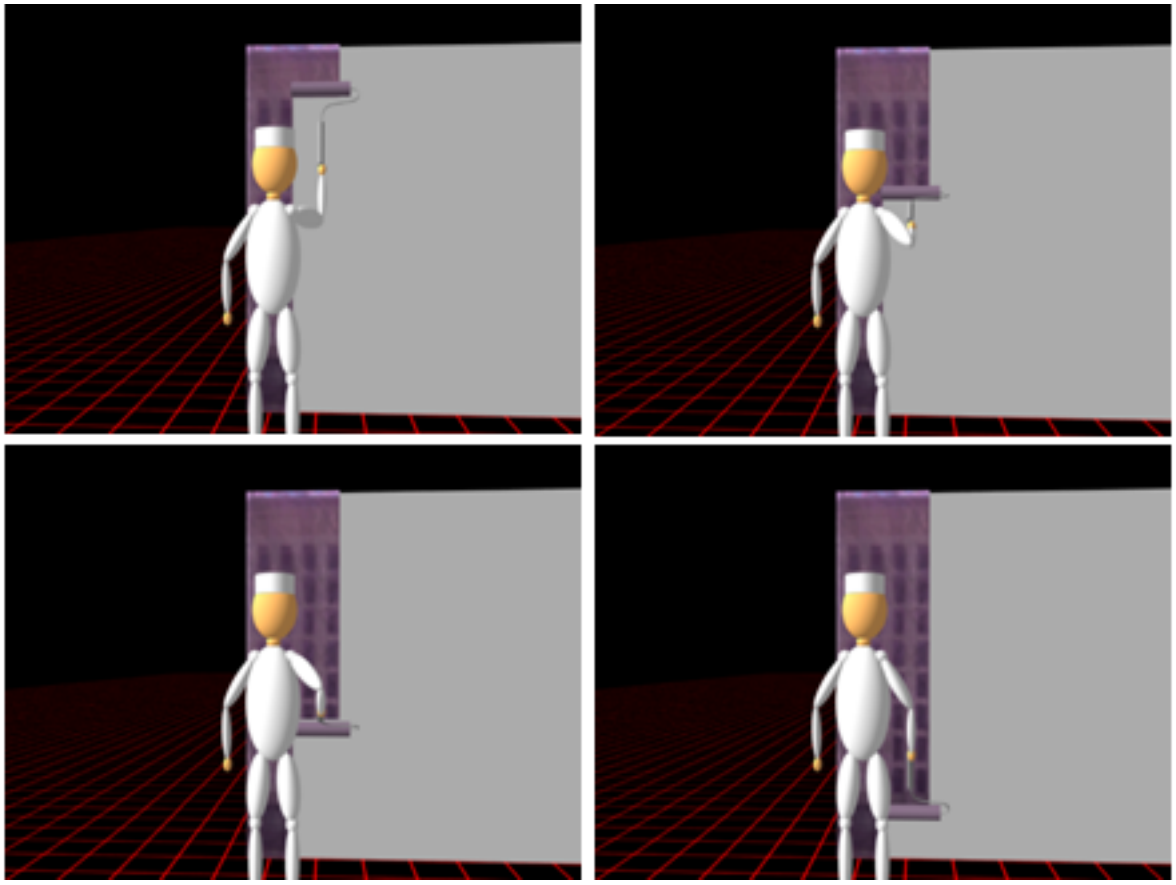

Figure 3-2: As the painter moves his roller down the surface of the building, the rectangle that models the "paint" grows, and the texture coordinates applied to the geometry change, in order to give the appearance that the paint is being rolled on.

2). If these texture coordinates weren't animated, the entire length of the texture would be applied to the rectangle regardless of its size, and we would not achieve the appearance of the texture being painted on.

Alias also simplifies the task of animating the synthetic camera. All of the camera motion in the visualization was animated along a spline automatically computed by Alias from user-supplied control vertices. This also eased the task of matching the camera position between two segments that were animated separately but between which we wanted a smooth transition: we simply had to insure

that the last control vertex in the camera path of one segment was the same as the first control vertex in the camera path of the following segment.

We chose Alias also because of its wire-framing capabilities. Models and animations can be viewed in wire-frame mode, which displays simple outlines rather than the actual surface of the geometry, and does not compute color or texture for the geometry. Thus, animations can be quickly previewed, viewed from different angles, and edited without waiting for rendering.

3.2.2 Rendering

As a large part of the animation for the video was done in Alias, we decided to also render the animation using Alias. The renderer's versatility also recommended it: its ability to quickly render low-quality, reduced size images allowed us to preview and modify an animation without waiting the long intervals necessary to produce high quality, full size renderings. The renderer can also be invoked from the command line, which allowed us to run rendering processes on remote machines easily. Our decision to use the Alias renderer did, however, pose a problem for rendering the Inventor format models. Although Alias includes an Inventor to Alias converter (`IvToAl`), it handles only a small subset of valid Inventor files and fails to correctly convert certain components— texture coordinates for example— of all Inventor files.

Alias uses two different file formats. A *wire* file format that encodes modeling and animation data is readable and writable only from the Alias user interface. For rendering, Alias converts wire files into its *SDL* (Scene Description Language) format. SDL files are human readable but, unfortunately, can not be translated back into the wire format and so can not be viewed or edited within the Alias interface. Since we could read and replicate the syntax of the SDL file, though, we

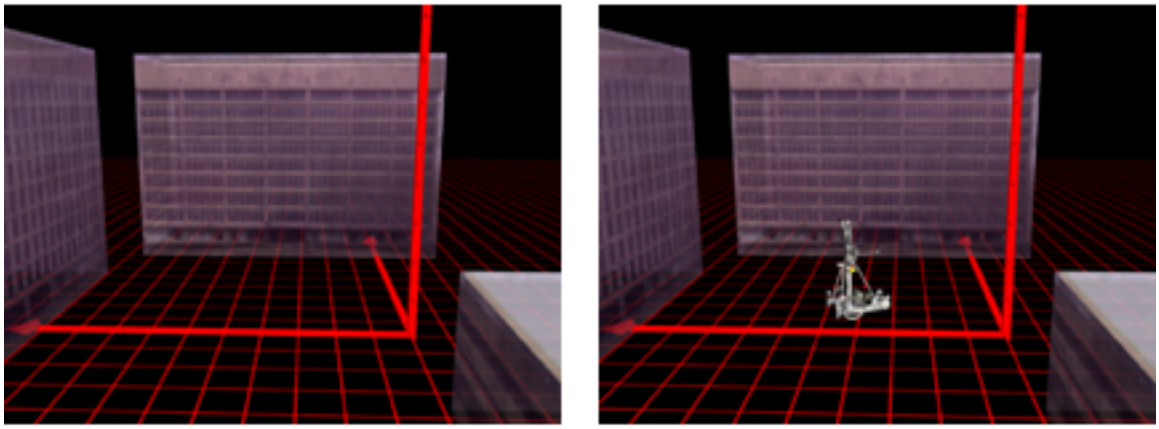

Figure 3-3: The geometry in frame shown on the left was modeled entirely in Alias. The frame on the right shows the rendered result of merging two SDL files: the one that produced the rendering on the left and one containing the Argus model translated from Inventor format.

were able to create our own SDL files and render them with the Alias renderer. We used this strategy to translate the Inventor format models— the Argus platform and the model of Tech Square produced by Coorg's algorithm— into SDL format. We were also able to combine the geometry in these files with geometry modeled in Alias by writing out both as SDL files and then merging the two into a single valid SDL file. In fact, we used this merging technique on every SDL file translated from Inventor. We used Alias to animate the camera and position the lights and any other "background" geometry that we wanted to appear in the scene, wrote out the file in SDL format, and merged it with the SDL file containing the geometry translated from Inventor format.

3.3 Premiere

We used Adobe Premiere 4.2 to combine the rendered frames and audio tracks into a single movie file. Premiere is a video editing package that offers a wide variety

of editing capabilities [Ado94]. Premiere allows editing at the single frame (1/30th of a second) level.

Premiere allows synchronization of audio and video tracks, mixing of audio tracks, and superimposition of video images. It provides a number of transitions that can be used to move from one video clip to another. Despite the large number of available transitions, we used only an additive dissolve transition, which allowed us to fade one clip smoothly into another. Premiere also lets the user apply filters to clips: filters allow modification of properties such as the brightness and contrast of the video imagery.

Chapter 4

Audio

4.1 Script

The narrative script for the visualization underwent several major revisions. The final version is included as Appendix B. A segment describing Coorg's Vertical Facade Extraction Algorithm [Coo98], which we did not include in the visualization, is included as Appendix C.

4.2 Narration

The narration was recorded with a mono microphone on an SGI O2 using the Irix `soundeditor` utility. Recording at a sample rate of 11.025 kHz (kilohertz) gave us the best sound quality with the available equipment. Higher sampling rates tended to pick up more undesirable background noise during recording. Recording was done in small, paragraph sized sections which were further split or joined as necessary in Premiere.

4.3 Sound Effects

Most of the sound effects used in the video were downloaded from the Internet at `http://www.wavcentral.com/effects.htm`. The click that indicates the camera shuttering, the stamping noise that accompanies the annotation of an image with its metadata, and the squeal of tires and breaking of glass that accompanies the collision were all available from several sites. The grumbling that represents the epithets which the annoyed painters hurl at each other near the end of the introduction was recorded with the help of one of the students in the lab. The gain of each of the sound effects was decreased using the Irix `soundeditor` utility so that they would not drown out the narration. The sound effect tracks were mixed with the narrative track and synchronized with the action of the animation using Premiere.

Chapter 5

Video

The design and implementation of the animation presented many challenges. Our foremost concern— and the most difficult aspect of the visualization design— was the ability to convey our material in a short enough time and with enough action to retain the viewer's interest for the duration of the visualization. To this end, we incorporated as much animation as possible to illustrate the concepts we wished to present and tried to minimize the length of narration that didn't have accompanying animation.

While creating the animations, we learned that the composition of a scene— placement of objects, direction of motion, and camera angle— figured just as importantly in the final product as did the technical aspects of the information we presented. We revised the initial designs of several scenes in order to make the visualization not only technically accurate, but aesthetically pleasing as well.

We also encountered several situations in which technical accuracy and aesthetics seemed at odds. For example, the Argus platform is grossly out of scale with the Tech Square model in which the visualization places it. Modeling the platform at an accurate scale with respect to the buildings didn't allow for a view of the scene

that showed the platform *situated* in the region: in any view that showed a good portion of the buildings, Argus was too small to be noticeable, and in any close-up view of the platform, only a small portion of a building facade was visible. Our solution was to increase the size of the platform so that it would be visible while still providing the viewer the knowledge of where it was situated in the scene.

We also discovered camera motion to be one of the most important aspects of our presentation. Animating not only the objects in our scenes but also the camera that views them added another dimension to the visualization. Camera motion is one of the most unique aspects of video; combined with the three dimensional nature of our models, it gave us the ability to view objects and scenes from any angle and to animate the camera around objects and through scenes. We experimented with many different camera motion paths for each visualization segment in an attempt to create one that best displayed the scene and its objects.

The visualization begins with a definition of 3D modeling and a humorous introduction to that topic, an introduction which also suggests the motivation of the City Project. The video continues with a description of the Argus platform and its data acquisition process, an explanation of the algorithms that reconstruct 3D models from image and pose data, a display of the results of Coorg's Vertical Facade Extraction Algorithm: a textured, 3D model of Technology Square, and a discussion of the Project's future plans.

5.1 Definition

The video opens by presenting a definition of the topic of the City Scanning Project, three dimensional modeling: generating a representation of an object or scene which is amenable to manipulation by a computer program. In order to empha-

size this definition, we wanted to display it on the screen in addition to having the narrator state it, yet we did not want to leave a static group of words on the screen for the period of time that it took the narrator to read the definition. Therefore, we applied two transformations familiar to any student of graphics— scaling and rotation— to the word "modeling" to add action to an otherwise inactive definition.

The modeling for this segment was done in Alias. Alias automatically computes spline surfaces for each letter. We animated the "modeling" text by keyframing the scale and rotation attributes of that group of letters. Originally, we modeled the definition in color, but after watching the segment on tape, decided to change it to white to improve its readability.

5.2 Introduction

The next part of the video provides a humorous introduction to our topic. As the narrator describes the difficulties of traditional 3D modeling, animated workers act out the process of positioning and painting a model of Technology Square. While the metaphor for the modeling process is intended to amuse the audience, the message that it conveys is vital: the difficult and time-consuming nature of 3D modeling is the motivation behind the City Scanning Project [Tel97].

Most of the modeling and animation for this section was done in Alias, the Technology Square buildings being the only exception. The model of Technology Square was originally generated in Inventor format by Coorg's Vertical Facade Extraction algorithm. We translated it into Alias wire format for the purpose of the visualization. Unfortunately, as previously mentioned, the translation fails to preserve the texture coordinates applied to the geometry. Although the texture co-

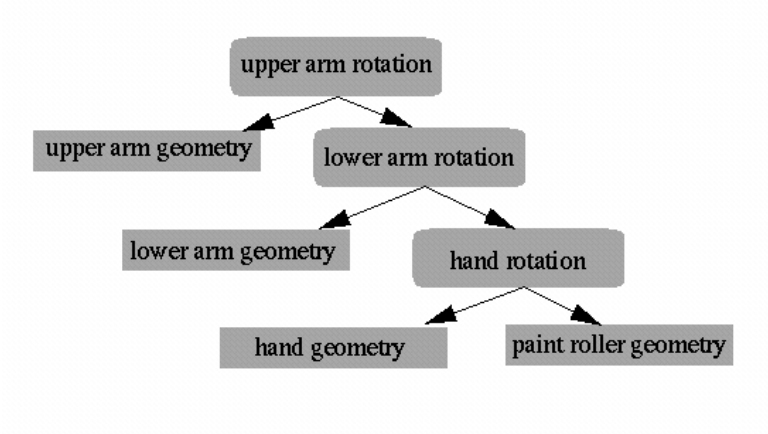

Figure 5-1: The hierarchy of the arm model: a rotation applied to the upper arm rotation node affects all of the nodes below it.

ordinates could be fixed in an Alias SDL file, we then would have had to write out two separate SDL files— one with the buildings, another with the Alias-modeled geometry— and merge them together before rendering. To save the CPU time required to merge thousands of files, we instead fixed the textures on the building geometry by hand— four buildings with five sides each meant only twenty surfaces to fix.

The rest of the modeling: workers, axes, stopwatch, and paint rollers, was done in Alias. Each worker, for example, is modeled as a hierarchical collection of less than twenty spheres. The hierarchy of the model simplifies animation. When the painter rotates his shoulder joint to raise his upper arm, his lower arm, hand, and the paint roller also move, since any transformation applied to a node also affects all the children of that node (Figure 5-1).

The animation of the workers was created by key-framing specific positions. We key-framed only six positions (Figure 5-2) for the walking animation, then repeated those key-frames with a translational offset to make each worker walk a specified distance. Some of the motion was also linked to other motion using Alias

Figure 5-2: The six key-frames that define the walking sequence for all of the modeled workers in the visualization.

expressions. While a worker is pushing a building into position, the building's translation is linked to the worker's translation so that when the worker moves forward, so does the building. Once the building is in position, its motion is "unlinked," and the worker walks off, leaving the building in place.

The rolling out of the grid at the beginning of this segment provides another example of linked motion. We wanted it to look as if the worker were carrying in and rolling out a carpet. The "carpet" begins looking like a cylinder; it shrinks in size as the carpet rolls out onto the floor. Although a physically accurate model of the carpet would be a single rectangular piece of geometry that is initially rolled up, modeling such a piece of geometry and its motion as it unrolled was unnecessarily complicated. Instead, we simplified the problem by modeling two separate pieces of geometry: a cylinder that decreases in diameter and a rectangle that increases in length as the grid rolls out. The scale of the rectangle and the cylinder are both linked to the translation of the cylinder, so that as the cylinder moves off into the distance, its diameter decreases and the length of the rectangle increases, and it

appears as if the carpet— or grid in this case— is rolling out along the ground.

5.3 Argus Description

The video continues with a description of Argus, the platform used to acquire the pose images. The main components of the platform are described: the leveling pins that stabilize the platform on uneven ground, the digital camera that takes the pictures, and the navigation equipment that provides the data necessary to annotate each image with pose information. The GPS antenna, inertial measurement unit, compass, inclinometer, and wheel encoders are all highlighted as important navigation components.

We originally modeled the Argus platform in Open Inventor; it therefore needed to be translated into Alias format for rendering. We translated the Inventor format file into an Alias wire file, and then animated it using the Alias interface. The glow effect that highlights each component as the narrator describes it was animated using the glow properties that Alias provides for the shaders applied to the geometry in question.

5.4 Data Acquisition

The simulation of Argus acquiring a portion of the dataset was also originally created with Open Inventor. The Inventor model of the Argus is hierarchical and contains linked motion, so that, for example, when the platform moves forward, the wheels rotate appropriately and all the components of the platform move in the direction of motion. The translation to Alias, which was necessary for rendering, preserved neither the state of the hierarchy nor the linked motion. If the platform were translated forward, it would leave behind some components: the

camera might end up floating in midair as the platform moved out from under it. The translation also grouped seemingly unrelated components under a single leaf node that did not allow access to the individual components: they therefore could not move independently of one another. In order to replicate the animation that had already been created in Open Inventor, we would have had to completely remodel the Argus in Alias.

In an effort to reduce the amount of work that needed to be redone, we instead wrote out every frame of the Open Inventor animation as an Inventor format file, translated to an Alias SDL file, merged the SDL file with another SDL file containing the synthetic camera, lights, buildings, and axes, and then rendered the merged SDL file using the Alias renderer.

The simulation of Argus acquiring data reads the image and pose data originally collected by the camera platform and stored on networked disk, to accurately recreate the platform's movements. The simulation not only shows Argus "in action," but also visualizes the nature of the data that Argus collects: the animation visualizes Argus's camera shuttering and acquiring images by funneling them into the camera, and also visualizes the hemispherical tiling of the images of a node by leaving a copy of each image at its original location (Figure 5-3).

5.5 Mosaicing

The next step in the process is to "mosaic" the images into a single, high resolution, hemispherical image. The mosaicing algorithm correlates the pixels on each of the acquired images to the pixels on the combined hemispherical image [CMT98]. We chose to visualize this by simply showing the distinct images fading into the final, hemispherical image.

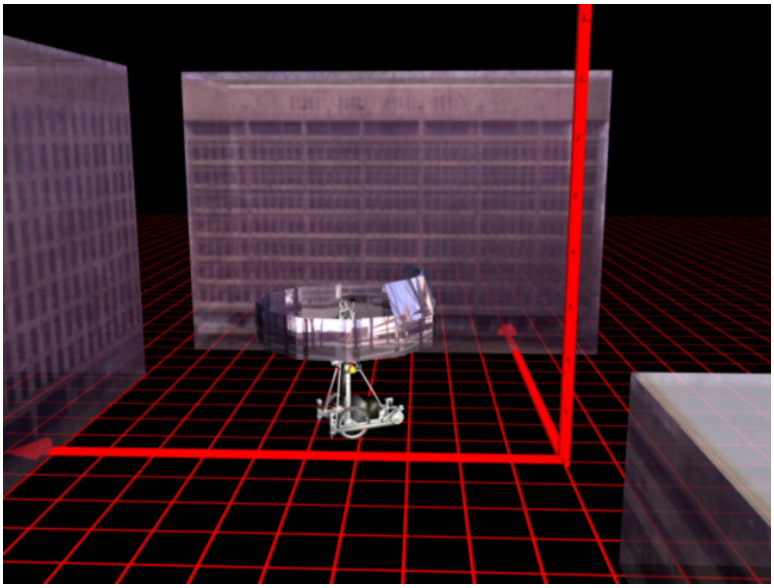

Figure 5-3: A visualization of the Argus platform acquiring the images of a node.

We created the animation for this section with C code that outputs Alias SDL files. For each of the images in the node, the code opens and parses the file that contains the pose data for that image. It places a model of the image in the appropriate position in the scene. In a sequence of SDL files, each image is translated from its original position along a vector toward the center of the hemisphere, fading at each step, so that it is entirely transparent when it reaches the boundary of the hemispherical image. Each of these SDL files is then merged with an SDL file containing the background geometry and camera animation.

5.6 Reconstruction

The next scene explains the high level concept of reconstructing 3D structure from the 2D imagery and navigation data collected by Argus. We illustrate the difficulty of the problem by showing that an image is only a two-dimensional array of pixels.

As the narrator explains how we use the combination of image and navigation data to determine existence, then calculate position, orientation, and appearance information for building facades in the scene, the animation fades in a camera "icon" at each camera position that observed a specific point on the facade, and highlights the sight-line from the camera, through the image, to the specified 3D point.

To determine which camera positions observed the chosen 3D point, we first ran Coorg's texture estimation algorithm [CT98] for the facade on which the point lies. The algorithm reports each image that it uses to determine the appearance of the facade. We wrote a C program to parse the output, look up the pose associated with the image, and project the 3D coordinate of the point back through the camera transform to determine if the specific point in which we're interested is visible in the image. If the point is visible, we extract the color of the pixel that observes it, and apply that color to a sight-line from the camera to the point to produce the frames that illustrate how appearance information is determined for portions of building facades.

5.7 Results

We showcase the current results of the Project with a flythrough of the final, textured model produced by Coorg's Vertical Facade Extraction Algorithm. Although the model was originally in Inventor format and is the only geometry in this scene, we still needed to translate to Alias format to render the animation frames. Since the Inventor to Alias translation doesn't preserve texture coordinates and the model includes hundreds of texture mapped polygons, we wrote a Perl script which corrects the texture coordinates in the SDL file output by `AlToIv`. The SDL file con-

taining the Tech Square geometry is then merged with a sequence of SDL files which describe the camera motion.

Of course, the 3D model generated as output from the Project's algorithms can be used for many purposes. We illustrate three in the video: line-of-sight determination, path planning, and collision detection. Our original script called for animated characters to peek around the corners of the buildings at each other to illustrate line-of-sight determination, and for a car or other vehicle to crash into a building to illustrate collision detection. Since the entire segment lasts less than ten seconds, though, we decided that trying to include such complex models for so short a period of time would look too cluttered and rushed, so we simplified the segment by making all of the objects simple spheres. The modeling and animation for this segment was done in Alias.

5.8 Conclusion

The next segment proposes the future plans of the Project and concludes the video. As the narrator discusses the work that we've completed so far, a representation of each of the 81 nodes that have been acquired pops in one at a time. As the narrator explains that the City Scanning Project's plans for the next three years include the modeling of the entire MIT campus, the synthetic camera pulls back from its view of Tech Square to the other end of campus.

The animation for this segment was created with a combination of Alias and C code. As for every animation sequence, the camera motion was animated in Alias. The Tech Square buildings and map of the MIT campus were positioned using the Alias interface. C code that parses the pose data files for each node output SDL files with geometry representing each node appropriately positioned.

Chapter 6

Rendering

We rendered each of the frames of the video using one of the renderers included in the Alias package, specifically the executable named `renderer.i6.5k`. We rendered approximately 7,000 images in TIFF (Tagged Image File Format) format. Each high-quality, full-size image took between one and eight minutes to render, depending on the complexity of the scene being rendered and the load on the machine on which the process was run. The image files are between 300 and 500 kilobytes each.

6.1 Rendering Parameters

While we rendered at a small size and low quality throughout the development and editing of the animation sequences, we rendered at high quality and full NTSC frame size (which Alias defines as 645 by 486 pixels) for the final renderings. To determine appropriate parameter settings for final rendering, we experimented with different settings until we discovered those that resulted in the highest visually detectable image quality in the least amount of time. For final rendering,

we increased the adaptive subdivision levels, causing Alias to subdivide surfaces into smaller triangles depending on their curvature. We also increased the anti-aliasing levels, set jitter on, and had Alias apply a post-process anti-alias filter to the rendered images [Ali98d].

6.2 Lighting

We rendered all of the frames of the visualization subject to the same lighting conditions. We used only two lights: one ambient and one directional. The ambient light illuminates every object in the scene equally, regardless of distance or orientation with respect to the light: we used it to insure that no piece of geometry was rendered entirely black because of a lack of illumination. We chose the directional light to emulate the effects of the sun: light rays emanating from a directional light are effectively parallel and do not decay with distance.

6.3 Environment

Alias also allows the user to set a wide variety of parameters within what it terms the "environment." Perhaps the most notable of these is the background color (or texture) of the rendered images, a setting that we left at its default, black. The only parameters that we did change from their defaults were those that affected the glow properties of the shaders. Each shader in Alias has a glow intensity parameter, which can be set to a value between zero (no glow) and one (maximum glow). The environment, however, contains several other parameters that affect the appearance of a shader's glow in the rendered image. We modified these parameters to add noise and a give wider spread to our glows, an effect that we felt provided a more realistic looking glow than did the default settings.

Chapter 7

Compiling the Movie

We used Adobe Premiere 4.2 to compile the rendered frames and audio tracks into a Quicktime movie. We used Premiere on a PC rather than the SGI because of the lack of local disk space on the SGI at the time.

We constructed the movie in small segments in order to keep both the project within Premiere and the file size of the output movie manageable. Each minute of animation and narration compiled into a more than 1 gigabyte (Gb) movie file. For each segment, we synchronized the audio and video by marking locations in both tracks and then aligning the marks in the timeline that Premiere provides for the project. Many of the animation sequences had been designed and rendered to exactly match the length of the accompanying audio track. For these segments, we used Premiere simply to line up the audio and video and compile them into a movie file. For some sections, the audio track needed to be split into smaller sections to allow more time for the action of the animation. In other instances, the duration of a single frame was extended to leave a static image on the screen, either to emphasize that particular image, or to accompany narration that required no animation.

The title and credit segments were also created with Premiere. We created each distinct screen as a separate Premiere title file, and then faded them in and out by applying a dissolve transition between the title and a black screen. All of the citations which appear at the bottom of the screen during the visualization were also created as Premiere title files and then superimposed over the rendered frames. Each citation was compiled into an approximately 8 second Quicktime movie with a fade in at the beginning and a fade out at the end. Compiling each citation into a movie allowed us to have them fade in and out. We found that having the citations appear and disappear with no fade was too abrupt and possibly distracting to the viewer.

In a few instances, we also used Premiere to extend the duration of an animation to match a longer narrative track. Premiere lengthens a clip by repeating frames: if, for example, we had a five second animation but wanted to extend it to match ten seconds of narration, Premiere would replicate every frame in order to make the animation last twice as long. During playback, every frame appears twice, effectively giving a 15, rather than the desired 30 frames per second (fps) playback rate. We discovered that smooth motion at 30 fps may be unacceptable choppy at 15 fps. In almost every instance that we wanted to lengthen the animation, we went back and re-rendered the correct number of frames in order to insure playback at a true 30 fps.

Each visualization segment was compiled into an uncompressed Quicktime movie with a frame size of 645 by 486 pixels and single channel audio sampled at 11.025 kHz. We had originally hoped to be able to record the uncompressed visualization onto VHS tape, but the SGI O2 that we used to play the movie can achieve a 30 fps playback rate only if the movie file is JPEG (Joint Photographic Experts Group) compressed. Since the PC on which we ran Premiere didn't offer

JPEG compression, we output uncompressed Quicktime files on the PC, and then later compressed the movie file on the SGI.

The SGI O2 from which we recorded the visualization onto tape performs JPEG decompression in hardware, and is thus able to play back JPEG compressed movies at 30 fps. Unfortunately, JPEG is a lossy compression scheme which is particularly bad at maintaining the quality of computer generated images. Although compression caused a noticeable decrease in the image quality of the movie, the compression was necessary to achieve the required 30 fps playback rate.

After experimental recording of the movie onto VHS tape, we made a few small changes to the movie using Premiere on the SGI. Since outputting to the VHS tape seemed to considerably lighten the colors in the animation, we applied a filter that reduced the brightness of the movie. We used another filter to replace the pure red used to color the grid and axes throughout the animation with a softer red, as pure red tends to become over-saturated and bleed into neighboring pixels in NTSC (National Television System Committee) format. We also clipped a few pixels from each side of our frames in order to make the size of our final movie 640 by 480, to be compatible with the size restrictions of the JPEG hardware. We used Premiere to output the final digital movie at 640 by 480 pixels using Motion JPEG A compression at the highest available quality setting and with mono audio sampled at 11.025 kHz. The final version of the digital movie is eight minutes, twenty-six seconds long and has a file size of 1.6 Gb.

Chapter 8

Recording to VHS

We used the SGI IRIX utility `dmplay` (digital media file player) on an SGI O2 to record the JPEG compressed Quicktime movie stored on local disk onto a Panasonic AG-7355 SVHS Video Cassette Recorder using the NTSC standard. Although we used the SVHS deck and SVHS video cable in an attempt to increase the quality of the movie on the tape, we recorded onto a regular VHS tape for compatibility reasons. Storage of the movie file on local disk was vital to the machine's ability to achieve the required 30 frames per second playback rate.

We tried several different methods of outputting video from the O2 before we decided on `dmplay`. Premiere has a "Print To Video" function that plays the selected movie on the screen, captures the signal from the screen, and sends that signal to the video out line. Premiere was not able to achieve a reliable 30 fps playback rate, however, and appeared to be dropping as many as one out of every five frames during playback. We then attempted to use `mediaplayer`, the SGI media file player, to play the movie on the computer monitor while we ran `videoout`, a video capture utility, to capture the video signal from the screen and send it to the video out line. `mediaplayer` did play the movie at 30 fps, but it also introduced

a "tearing" effect, in which the imagery would appear to tear as some pixels from the previous frame were not updated at the next frame. Examination of the individual frames of the movie revealed that this was not a problem with the movie itself but rather an artifact of playback.

`dmplay` was able to solve both of these problems: it played the movie at a reliable 30 fps without tearing. `dmplay` has the ability to output the video signal directly to the video out line, rather than outputting the signal to the computer monitor and then capturing it from the monitor to output to tape. Since tearing was an artifact of playback to the monitor, bypassing output to the monitor eliminated tearing and noticeably increased the image quality of the final VHS tape.

Despite the improved results achieved with `dmplay`, we were still disappointed with the image quality of the movie on the VHS medium. Even after adjusting the brightness levels and replacing the pure red with an NTSC-safe color, the imagery on the VHS tape is much less sharp than the digital source imagery. Taking into consideration that VHS is not as high quality a medium as digital video, however, we feel that we've achieved the best quality possible with the available equipment.

Chapter 9

Future Work

The visualization presents an overview of the current state of the City Project. The various aspects of the Project continue to be developed, expanded, and improved: as the Project matures, the video might be updated to reflect new developments.

Several aspects of the Project are currently under development. The Argus platform is continually being improved: its appearance has changed in several minor ways since we created the original model. Argus also collects more nodes of image and navigation data daily.

There are also several components of the Project only quickly mentioned in the visualization that might be expanded. The original plan for the visualization included a description of Coorg's Vertical Facade Extraction Algorithm [Coo98], a segment that was eventually cut because of time constraints. The animation of this algorithm was intended to present information at a more technical level than the other parts of the visualization. It would have animated the detection of horizontal edges in the pose imagery, the histogramming technique that determines the orientation of the facades that the edges imply, the sweep plane technique that determines the locations of those facades, the link and commit step which joins small

pieces of the same facade together, and the estimation of the texture of the facade.

Several other algorithms, which are mentioned in the video only as "several complementary algorithms for inferring 3D structure from geo-located imagery," are also under development. One algorithm matches dense regions of texture from many images to infer the existence of oriented surface elements [MLPT97]. Another algorithm generalizes temporal tracking methods to a kind of spatial tracking of sparse features such as corners, edges, and polygonal facades [CT97a]. A third algorithm uses plane plus parallax and level set evolution methods to determine fine-scale geometry of modeled structures [Amr98]. Animation of each of these algorithms might be included in the visualization as they mature.

Chapter 10

Conclusion

This paper has described the design and production of a computer animated visualization of the MIT City Scanning Project. It has detailed each phase of the process, from writing the script and designing, modeling, animating, and rendering the visualizations, to compiling the disparate parts into a single Quicktime movie file and recording it onto VHS tape. We've presented both the aesthetic and technical challenges encountered throughout the project and detailed our solutions to them.

We are enthusiastic about presenting the video to a wide audience. We intend to distribute the video to the Project's sponsors and other labs in an effort to increase the knowledge of and interest in the City Scanning Project.

Appendix A

Still Images from the Visualization

Figures A-1 through A-6 show still frames from the visualization. Each frame is representative of the segment from which it was taken: Figure A-1 from the introduction, A-2 from the description of Argus, A-3 from the mosiacing segment, A-4 from 3D reconstruction, A-5 from the flythrough of the Tech Square model, and A-6 from the visualization's conclusion.

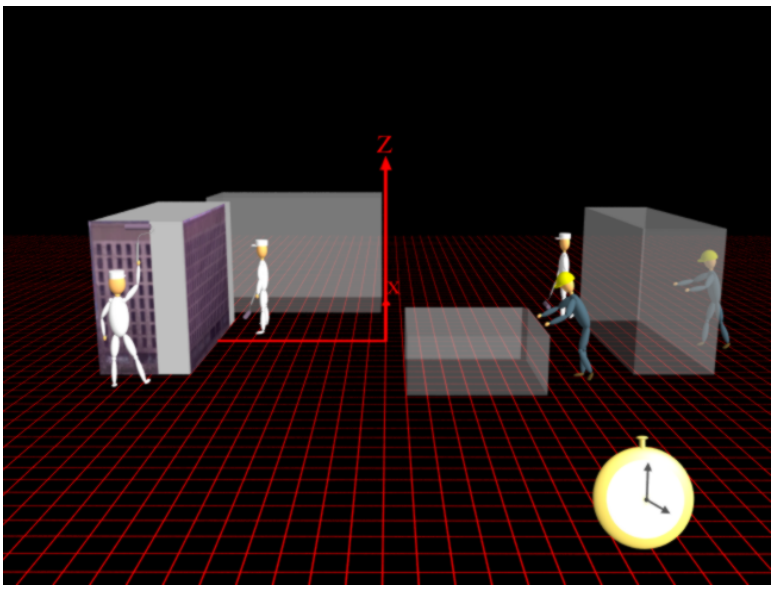

Figure A-1: Our metaphor for the 3D modeling process: workers position and texture the building geometry.

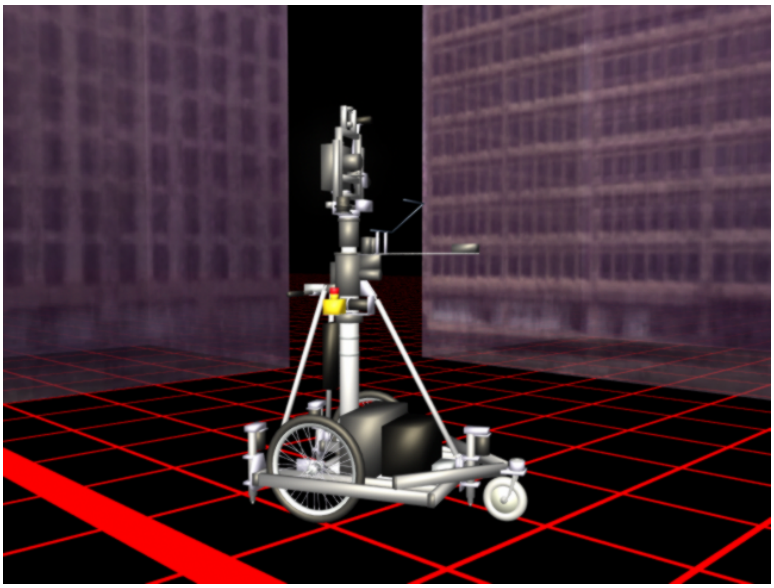

Figure A-2: The model of the Argus platform, situated in the model created from the data that it acquired.

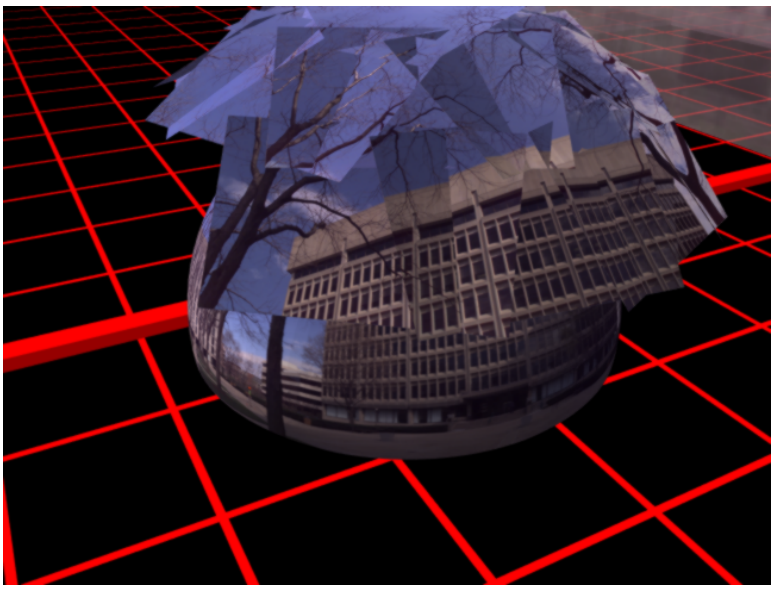

Figure A-3: A visualization of the mosaicing process blending all of the images acquired from one camera position into a single, hemispherical image.

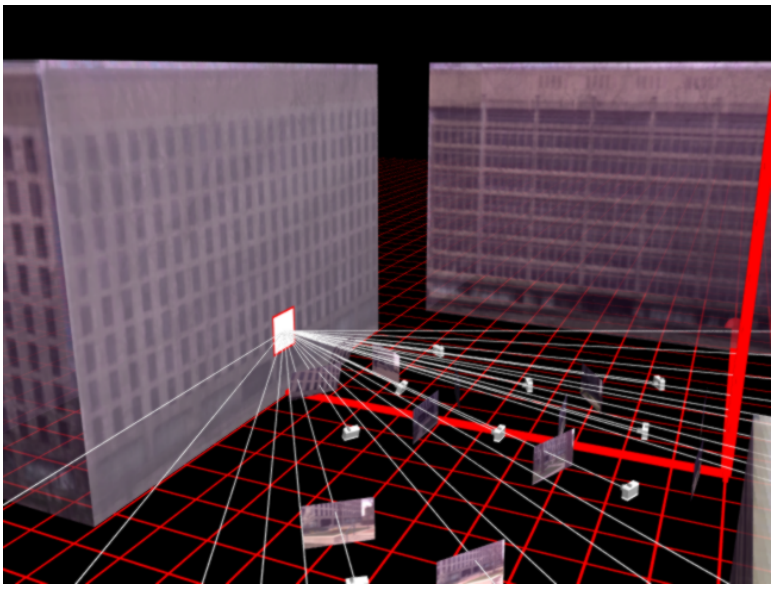

Figure A-4: Some of the camera positions that observed one specific point in the three-dimensional region of interest.

Figure A-5: The 3D CAD model of Tech Square generated by Coorg's Vertical Facade Extraction Algorithm and converted to Alias format for rendering.

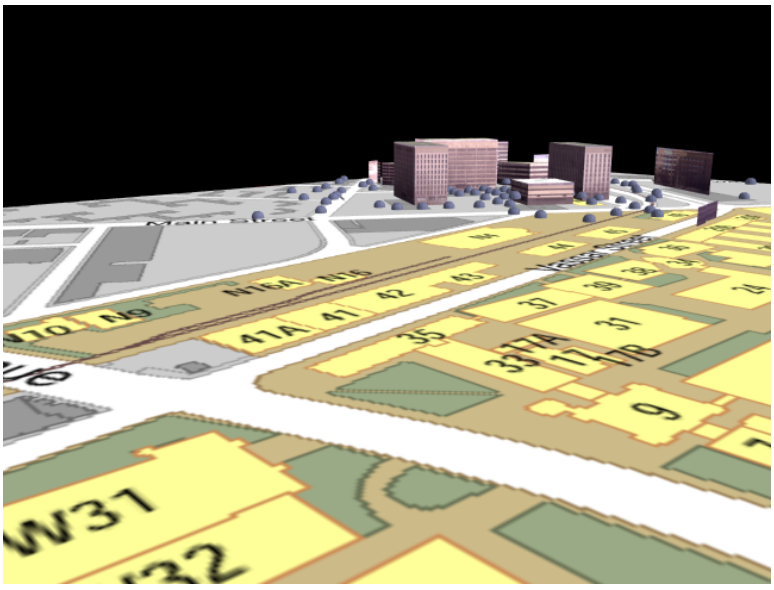

Figure A-6: A hint of the MIT City Scanning Project's future plans: to model the entire MIT campus.

Appendix B

Script

[In the following, quoted text is spoken by the narrator; stage directions are bracketed; and text to be displayed is indicated.]

City Scanning Video

(Title)

[Fade in] Acquiring Models of Urban Scenes from Geo-Located Images

[Fade in] MIT Computer Graphics Group

Seth Teller, Eric Amram, Mike Bosse, George Chou, Satyan Coorg, Barb Cutler, Doug DeCouto, J.P. Mellor, Tara Schenkel

(Introduction)

[Open to black screen]

"The modeling process – generating a representation of an object or scene which is amenable to manipulation by a computer program – is the starting point for most operations in computer graphics, computer aided design, and computer simulation." [Perhaps display text: Modeling, (italics) vt., generating a representation of an object or scene which is amenable to manipulation by a computer program."

"Modeling an existing urban area with three-dimensional geometry can be a tedious task. First, a coordinate system must be established."

[Workers roll out 2D coordinate grid to infinity, then erect a Z axis with tick marks.]

"Geometric structures must be crafted, then situated and oriented appropriately in the modeled world."

[Workers shove and drag buildings (geometry only) from offscreen, then rotate them a bit and settle them into place.]

"Texture or reflectance maps must be associated with building surfaces."

[Workers paint texture maps onto building surfaces.]

(Motivation)

"This process can require minutes or hours of human effort for each structure to be modeled..."

[Stopwatch running in lower right corner.] [At bottom, relevant paper cites subtly fade in and out:] [Cite Jepson et al., 1995] [Cite Debevec et al., 1996]

"... therefore suffering from a lack of scaleability. Adding people to the modeling process doesn't necessarily help."

[Workers get in each other's way attempting to help. OR, Workers pull in more parts from offstage, but they don't fit due to differences in size or shape, or inappropriate overlap or gaps.]

(Project Overview)

"In contrast, the goal of the MIT City Scanning Project is a scaleable, automated system for modeling urban environments, using computer vision techniques." [Cite Teller, IUW 1997, at bottom of screen.]

Since the bottleneck of existing modeling processes is the human component, our goal is to automate the process, removing humans from the loop entirely." [Hook pulls workers offscreen.]

[By now, Tech Square model is assembled and appropriately framed. It fades markedly, to become a translucent backdrop to the material which is to come next.]

"Our approach consists of acquiring many observations – in the form of digital images and navigation data – in the area to be modeled, then applying computer vision techniques to produce CAD models consistent with the acquired observations." [Show representative images.]

(Argus Overview)

"Navigation instrumentation attached to the camera annotates each acquired image with *metadata* recording estimates of the camera's position and orientation, as well as the acquisition time. Position and orientation are expressed in a natural, literally global coordinate system – namely, latitude, longitude, and altitude, and a quaternion describing the camera orientation." [A tag hangs twisting from the image. A view of it expands to readability. On it can be read something like:

Image Number:	11749
Camera:	Kodak DCS 420C
Resolution:	1524 x 1012 Pixels
Latitude:	42 21 54N
Longitude:	71 06 29W
Altitude:	12 Meters
Orientation:	0.604 0.610 0.365 -0.360
Time:	Wed 08 Jul 1998 14H:34M:45S

As each key word is read, the appropriate text line glows.]

"The instrumentation platform we use to acquire this data is named "Argus" after the hundred-eyed creature of Greek myth." [Inset picture of Greek Argus fades in]

"Argus is a manually propelled three-wheeled cart which is moved into and around the region of interest."

"A collection of actuators serve to level the cart [leveling pins glow] and shutter the camera [camera glows, picture flows into camera]."

"Argus's complement of navigation sensors includes phase- carrier GPS, an inertial navigation system, a flux-gate compass, an inclinometer, and quadrature wheel encoders." [Each instrument glows briefly as it is mentioned.]

"An on-board computer uses a Kalman Filter to combine all of the sensor readings, producing a single robust estimate of position and orientation for each image." [Argus' motion accelerates, as it moves about and snaps many images; the tag glows for each image as it is acquired and funneled in to the camera.]

"Argus acquires, at a collection of locations in and around the region of interest, high-resolution images grouped into 'nodes' – images which tile a sphere about a common optical center." [Zoom in on Argus, or blow up one position's acquired images, to show that they tile a sphere as they are acquired.]

"The raw imagery at each node is automatically mosaiced into a very high resolution, effectively hemi-spherical image, which we call a pose mosaic." [Show the faceted raw imagery smoothing into a single spherical mosaic.]

"The geo-located imagery shown here was acquired with an early, largely manual prototype of the Argus. Semi-automated software techniques were used to refine

the initial pose estimates." [Cite Coorg/Master/Teller, CVPR 1998, at bottom of screen.]

"The Argus navigation subsystem is currently under development, and slated for eventual full automation." [Cite DeCouto MEng thesis, MIT 1998, at bottom of screen.]

(3D Reconstruction)

"Of course, geo-located imagery alone isn't sufficient to serve as a model of the scene. We must somehow infer three-dimensional structure, and texture, of the observed urban scene."

"An image is simply a two-dimensional array of pixel values." [Show an image, inset, and have a crosshair move along it with the x, y coordinates changing and reported at the ends of each crosshair bar.]

"Each 2D pixel value arises from an observation of radiance, roughly color and brightness, along some 3D line of sight in the world." [Image and crosshairs rotates to be viewed obliquely; camera icon appears; ray from the world extrudes *back* through the image and into the camera. Radiance somehow represented, as a blob of color coming to rest on a pixel on the image plane.]

"The fundamental idea of the City Scanning Project is to deduce, from a large number of geo-located digital images, three things for each region of space." [Large number of situated images appear around a ghostly Tech Square. A small region

of one building is isolated, and all the cameras within range of it glow; their sight-lines to the patch are isolated, each represented with pixel color.]

"First, we wish to establish whether there exists built structure in the region."

"Second, if structure is present, we estimate its location and orientation in space." [tag appears on building patch notings its position and normal, which is visualized.]

"Third, we deduce appearance information for the geometric element there: color, texture, or BRDF (Bi-directional Reflectance Distribution Function), depending on available information." [Somehow we indicate inference of the named quantities.]

"We are developing several complementary algorithms for inferring 3D structure from geo-located imagery."

One algorithm detects significant vertical facades in the scene, and estimates for each facade a high-resolution texture map." [Cite Satyan Coorg, 1998 PhD thesis.]

(Results)

"The result of these computations is a textured, three-dimensional CAD model representing the structure and appearance of the acquired region of the world. The model is expressed in the same global coordinate system in which the image data were acquired, facilitating the merging of models acquired by several sensors operating in parallel, for example from ground and aerial platforms."

[Flythrough of final, textured model]

"Such CAD models form the starting point for a variety of useful simulations; not only computer graphics rendering, as we show here, but other physically-based simulation operations such as line-of-sight determination, path planning, and collision detection."

[As these last three are said, show

1. two characters peeking in and out, with a solid ray joining their eyeballs whenever they can see eachother;

2. show an aerial view; then two dots glow in, "start" and "finish", then and a dashed line, with its progress animated, tracing a crooked path from one corner of the square to a far corner;

3. show a character or car veering off the path, and bouncing off the buildings?]

(Conclusion, Future)

"As of Summer, 1998, we have acquired and processed a collection of 81 nodes – almost 4,000 geo-located images – to generate a textured CAD model of MIT's Technology Square, and portions of its neighboring structures. Our plans for the next three years include the acquisition and modeling of MIT's campus, which we estimate will require several Terabytes of ground-based and aerial imagery and navigation metadata."

[Pull back to show model sitting on 2D view of MIT campus]

(Credits)

Video Production

TARA SCHENKEL

Script

Seth, Tara

Narration

Britton (Bryt) Bradley

Thanks to:

City Scanning Group

Computer Graphics Group

MIT Lab for Computer Science

This video was made using

Alias Studio, Pixar Renderman, etc.,...

Appendix C

Proposed Script for Algorithm Segment

(Satyan's Algorithm)

"The vertical facade algorithm works as follows."

"Since we are modeling an urban scene, most edges observed will be either horizontal or vertical, and many will arise from vertical facades. It is these facades which our algorithm detects."

[Spherical node "unrolls" to form 4 sides of cubical environment map.] [Edge detection occurs on each environment map face; highlight edges.]

"Each presumed-horizontal edge, along with orientation information about the camera, yields an orientation for the vertical facade with respect to some reference direction (say, due East)."

[Show image plane; highlight edge, then its extrusion into a triangular wedge. Now intersect this wedge with a slowly rotating vertical facade, with a glowing segment of intersection between the wedge and the facade, a glowing "Theta = XX degrees" and swept arc between the facade and a line on the ground plane going off to screen right. This Theta designation changes as the facade rotates. As the line of intersection becomes horizontal, it glows again, and a persistent trace of the facade, at the correct orientation, is left behind.]

"This orientation is deduced for each detected edge, creating a histogram of orientations."

[Above process is repeated once more quickly, then once very quickly; along bottom of screen a series of buckets labeled Theta: 10 .. 20 .. 30 .. .. 360 is shown; each deduced theta above adds a bit of height to its corresponding bucket. Now the process speeds up; each edge flashes in turn and its corresponding contribution flashes. The histogram takes shape, revealing several clear peaks.]

"Each peak in this histogram corresponds to the orientation of one or more significant vertical facades in the scene."

[Make each peak glow; as it does, make the corresponding face or faces in the dim Tech Square model glow.]

"However, the histogramming technique does not yield any information about the *location* of the vertical facades. To estimate their location, we use a sweep-plane technique. The region of interest is divided into a 2D grid."

[Show grid cells papering the region of interest.]

"Through each grid cell, we sweep a plane whose orientation matches that of the sought facade. That plane position which explains the largest number of edge observations, as measured by correlation of edges, is the facade we seek."

[Show plane section moving across grid cell; show edges reprojected onto this plane; show at bottom of screen a function y=f(x) being swept out from left to right as the sweep plane approaches, then occupies, then recedes from the correct facade position. The "occupying" position should of course show a peak in y=f(x); this data should be available from Satyan.]

"A 'link and commit' step links fragments of vertical facades so identified with their neighbors, and handles occlusion constraints. Any resulting outliers – facades ending up with an area less than one hundred square meters – are discarded."

[Facades gently "fade in" from many different grid cells, colored distinctly (say, according to grid cell id); they then relax to a set of pastel colors, one per facade, or one per orientation. Small facade sections fade away.]

"Each facade is then extruded to the ground plane. This completes the extraction

of gross geometric structure from the scene."

[Vertical extrusions smoothly drop down to the ground plane from any facades that do not already rest there.]

"The next step is to estimate reflectance information for each geometric facade. The registered imagery contains many rectified images of each facade. However, these images are typically partially occluded by foliage, or by other structures between the camera and the facade."

[Highlight facade to be subject to texture estimation; line up 4-8 rectified observations of facade at screen bottom.]

"We estimate a texture map for the facade by discretizing its surface into thousands of small rectangles... "

[Show blank facade, diced up into grid.]

"... then estimating the color of each rectangle based on all available observations. We use a weighted median of observations to estimate the texture; this is robust against outliers, and yields textures superior to those generated by averaging."

[Here is one possible way of showing this:

1.) View rotates to show target facade full frontal, with a grid of initially unassigned texels. Below the facade are arranged 4-8 rectified views.

2. Upper left texel square is high lighted. Then, upper left pixel from each rectified view is highlighted, extracted, and moves into a horizontal row of single pixels.

3. The horizontal row is "sorted," say by luminance, to produce a row that goes from (say) dark to light.

4. An outline around the median (middle) pixel glows, then the pixel follows an animated path up to its destination texel, where it glows, then comes to rest.

5. Steps 1)-4) repeat much more quickly for remaining texels, thus filling in remainder of target facade.

6. View withdraws from target facade, to show the remainder of the facades filling in (or already filled in) as well as the view encompasses all of Tech Square.]

"Texture estimation is performed for every identified facade."

Appendix D

Implementation Details

D.1 Summary of Tools

All software used in the creation of the visualization was run either on an SGI O2 running Irix 6.2, or a PC running Windows NT. Our O2 has a MIPS R5000 processor and a gigabyte of memory. The PC has 261 megabytes of memory and a 3D Labs GLINT Twin-500TX + GLINT delta Graphics board and NeTpower ULTRAfx Video Accelerator Adapter. Table D.1 summarizes the tools used during the creation of the visualization.

Tool	Platform
Adobe Premiere 4.2	PC and SGI
Alias Studio 8.2	SGI
IvToAl	SGI
Open Inventor	SGI
dmplay	SGI
ivfix	SGI
renderer.i6.5k	SGI
soundeditor	SGI
write_multiple_sdl	SGI

Table D.1: Summary of tools used in the implemenatation of the visualization.

D.2 Data Acquisition Simulation Development Files

Files for the Argus data acquisition simulation Open Inventor application are in the /home/etara/Argus/ directory. The C++ files and executable which writes out an Inventor format file for each frame of the data acquisition simulation (to be later translated to Alias format) is in /home/etara/Argus/toAlias2/.

D.3 Animation Development Files

Tables D.2 and D.3 describe the Alias wire files, Perl scripts, and C files that were used to create the animation frames and sound files on the SGI.

After rendering, the frames and audio files were transferred to the PC to be compiled into Quicktime movie files. The individual frames and audio files, as well as

Subdirectory	Description
MosaicSphere/	creates a texture-mapped hemisphere in Alias SDL format
Smooth/	files for animation of mosaicing of images into a spherical image (see also /d4/video/Smooth/)
Tiles/	files that create an SDL file containing one node of positioned images
cameraPos/	files that make an SDL file with all the cameras that view a specified 3D point in the model
sightlines/	files that output an SDL file showing sightlines from cameras to a 3D point

Table D.2: Description of animation development files under the directory /home/etara/tsq/.

the Premiere project files created for each segment of the visualization are stored on the C: and D: drives of the machine contrast.lcs.mit.edu, in directories named *tara* on both drives.

Tables D.4 and D.5 describe the intermediate movie segments produced by Premiere. The final Quicktime movie file that was recorded to tape is named final.mov and is stored in the directory /local/tara/ToTape on the machine mosiac.lcs.mit.edu. It has a file size of 1,797,003,053 bytes.

Subdirectory	**Description**
ArgusSection/	wire files for animation of the Argus platform
Fade/	files that fade the textures on the Tech Square buildings
FixRoll/	wire file of first construction worker rolling out the grid at a slower rate than originally rendered
Flythrough/	files to animate flythrough of the Tech Square model
Intro/	wire files for introduction segment animation
Map/	files to create animation of the MIT campus map fading in and the animation of the camera pulling back across campus
Modeling/	wire files for animation of the 3D modeling definition
Narration/	narration sound files
Nodes/	files for popping in each of the nodes one at a time
Recons/	files for creating 3D reconstruction segment
Satyan/	wire file for short segment of animation that plays when Satyan Coorg's thesis is cited
SimSection/	wire files for visibility, path planning, collision detection segment
SimulationFrames/ SimulationWork/	files which control translation of the data acquisition animation from iv to SDL format
Smooth/	more files for animation of mosaicing of images into a spherical image (see also /home/etara/tsq/Smooth and /home/etara/tsq/MosaicSphere)
Sounds/	sound effects (camera, grumbling, etc)
tsqModel/	files that translate the Inventor model of Tech Square into Alias SDL format

Table D.3: Description of animation development files under the directory /d4/video/.

File Name	**Size** (bytes)
argus1.mov	1166678793
argus2.mov	1106568513
conclusion.mov	1217670153
credits.mov	812246434
intro1.mov	1982358512
intro2.mov	1007537437
flythrough.mov	1505814200
model.mov	767293049
recons.mov	1696021985
smooth.mov	993134924
sts.mov	74809142
wave.mov	278035257

Table D.4: A list of the uncompressed Quicktime movie files compiled with Premiere on the PC and stored on mosaic.lcs.mit.edu in the directory /local/tara/Uncompressed/.

File Name	**Includes**	**Size** (bytes)
argus-all.mov	argus1.mov argus2.mov sts.mov	345637941
end.mov	conclusion.mov credits.mov	275043537
first.mov	intro1.mov intro2.mov modeling.mov	482519273
flycom.mov	flythrough.mov	154666593
mid.mov	recons.mov smooth.mov	522305937
wave-com.mov	wave.mov	2750902

Table D.5: A list of Quicktime movie files compiled with Premiere on the SGI with Motion JPEG A compression after being filtered for brightness, size and color. These files are stored on mosaic.lcs.mit.edu in the directory /local/tara/ToTape/. Each of these files is a combination of of one or more of the uncompressed Quicktime files listed in D.4, as indicated.

Bibliography

[Ado94] Adobe Systems Inc., Mountain View. *Adobe Premiere 4.0 – User Guide*, 1994.

[Ali98a] Alias-Wavefront, a division of Silicon Graphics Ltd., Toronto. *Alias Basic Tools Version 8.5*, 1998.

[Ali98b] Alias-Wavefront, a division of Silicon Graphics Ltd., Toronto. *Animating in Alias Version 8.5*, 1998.

[Ali98c] Alias-Wavefront, a division of Silicon Graphics Ltd., Toronto. *Learning Alias Level One*, 1998.

[Ali98d] Alias-Wavefront, a division of Silicon Graphics Ltd., Toronto. *Rendering in Alias Version 8.5*, 1998.

[Amr98] Eric Amram. A variational technique for three-dimensional reconstruction of local structure (to appear). Master's thesis, MIT, 1998.

[Apo92] Tom Apostol. *Project Mathematics! Sines and Cosines*, Parts 1-3. (video), 1992.

[Bli95] James F. Blinn. Designing the animation for Project Mathematics!, Siggraph 95 course notes. http://www.research.microsoft.com/research/graphics/blinn/MATHDESN.HTM, 1995.

[CMT98] Satyan Coorg, Neel Master, and Seth Teller. Acquisition of a large pose-mosaic dataset. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[Coo98] Satyan Coorg. *Pose Imagery and Automated Three-Dimensional Modeling of Urban Environments*. PhD thesis, MIT, August 1998.

[CT97a] George T. Chou and Seth Teller. Multi-image correspondence using geometric and structural constraints. In *Proceedings of the 1997 Image Understanding Workshop*, 1997.

[CT97b] Satyan Coorg and Seth Teller. Matching and pose refinement with camera pose estimates. In *Proceedings of the 1997 Image Understanding Workshop*, 1997.

[CT98] Satyan Coorg and Seth Teller. Automatic extraction of textured vertical facades from pose imagery. Technical Report TR-729, MIT LCS, 1998.

[DeC98] Douglas S. J. DeCouto. Instrumentation for rapidly acquiring pose imagery. Master's thesis, MIT, June 1998.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modelling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH 96, Computer Graphics* Proceedings, Annual Conference Series, 1996, pages 11–20, August 1996.

[EGea91] David Epstein, Charlie Gunn, and et al. *Not-Knot*. (video), 1991.

[Eva94] Arthur Evans. *Open Inventor C++ Reference Manual*. Addison-Wesley, Reading, Massachusetts, 1994.

[FLR95] Olivier Faugeras, Stéphane Laveau, and Luc Robert. 3-D reconstruction of urban scenes from sequences of images. Technical Report 2572, INRIA, June 1995.

[HD] Alejo Hausner and David Dobkin. Making Geometry Visible: An Introduction to the Animation of Geometric Algorithms. In *Handbook for Computational Geometry*. Sack and Urrutia, to appear. Also at http://www.princeton.edu/~ah/alg_anim/animation/index.html.

[JLF95] William Jepson, Robin Liggett, and Scott Friedman. An environment for real-time urban simulation. In *1995 Symposium on Interactive 3D Graphics*, 1995.

[LMM94] Silvio Levy, Delle Maxwell, and Tamara Munzer. *Outside In*. (video), 1994.

[LMTea] Stuart Levy, Tamara Munzer, Lori Thomson, and et al. *Shape of Space*. (video).

[MLPT97] J. P. Mellor, Tomás Lozano-Pérez, and Seth Teller. Dense depth maps for epipolar images. Technical Report 1593, MIT AI Lab, 1997.

[TD95] Ayellet Tal and David Dobkin. Visualization of geometric algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 1(2), 1995.

[Tel97] Seth Teller. Automatic acquisition of hierarchical, textured 3D geometric models of urban environments: Project plan. In *Proceedings of the 1997 Image Understanding Workshop*, 1997.

[Wer94] Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley, Reading, Massachusetts, 1994.