Modeling 3D Scenes from Video

Marek Czernuszenko[1], Daniel Sandin, Andrew Johnson,

Thomas DeFanti

Electronic Visualization Laboratory (EVL),

University of Illinois at Chicago

E-mail: {marek | dan | aej | tom } @evl.uic.edu

Tel : +1-312-996-3002

Abstract

We present a technique to obtain texture-mapped models of real scenes with a high degree of automation using only a video camera and an overhead projector. The user makes two passes with a hand-held video camera. For the first pass the scene is under natural illumination, and a structure-from-motion technique recovers coarse scene geometry and textures. For the second pass a grid of lines is projected onto the scene which allows us to acquire dense geometric information. The information from both passes is automatically combined and a final model consisting of the dense geometry of the scene and a properly registered texture is created.

Key Words: Models from video, geometry recovery, virtual reality.

[1] currently with Exxon Production Research Co.

1 Introduction

The process of creating real-world scene models for inclusion in virtual environments is currently a very labor intensive task. Usually all of the objects in the scene have to be measured precisely and then manually recreated using modeling software, including hand registering photographs for use as texture maps. Recently there has been a great deal of interest in creating models from the real world based solely on photographs or video.

There are several current approaches. Image-based techniques, Light Field (Levoy and Hanrahan 1996) and the Lumigraph (Gortler et al. 1996) generate new views of a scene without recovering the scene geometry. They require a tremendous amount of storage and even a simple scene in an uncompressed form takes over 1GB of memory. The Light Field method requires additional hardware: a stepping motor and motion platform. The Lumigraph approach requires full camera calibration that involves computing both the intrinsic and extrinsic camera parameters for each camera location. In the recently proposed hybrid geometry- and image-based approach (Debevec et al. 1996), the user creates a coarse model of scene from a set of geometric primitives and then manually matches edges in the model to the edges in the photographs. The algorithm then finds the precise dimensions of the model based on images. However the coarse modeling is not automated and the whole process is only a small improvement over the traditional modeling. Yet another approach to create models is to use laser scanners (Cyberware 1997) but these scanners work only with small objects, and they are very expensive if capable of modeling human-size objects.

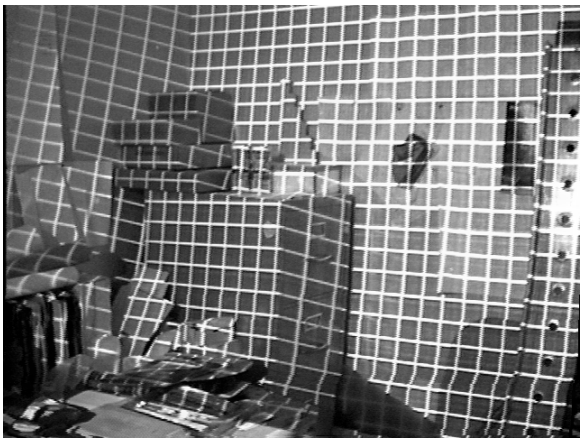

Figure 1. An example of structure light: a grid of lines projected onto the scene.

Our method consists of two videotaped passes performed with a hand-held camera. The first pass is executed under natural lighting conditions. The features are automatically selected and tracked in the video images. The 3D positions of features are then recovered using a structure-from-correspondence technique. Often there are featureless areas in the scene that will not become part of the reconstructed model as the recovered model consists of a sparse set of 3D points. This is not satisfactory because we want to obtain a much finer geometry of the scene. So, we perform a second video pass in which we project a dense grid of light onto the scene (Fig. 1). The line crossings created with the light are easy to select and track. Again, a structure-from correspondence approach is used to recover the 3D locations of these artificial features. This combines the ability to easily find corresponding points from a small baseline approach, with the more accurate positional information from a large baseline approach.

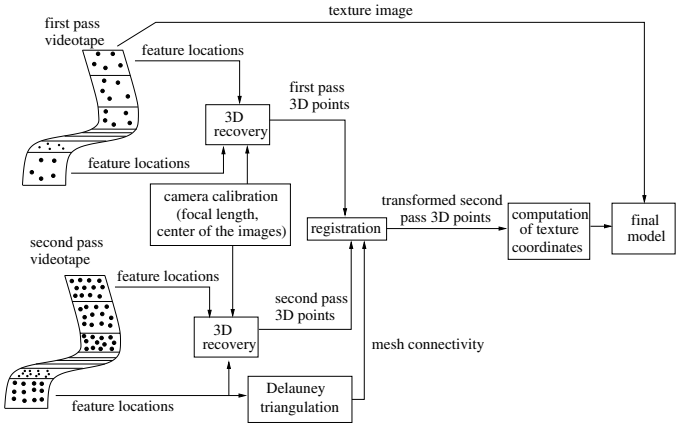

Figure 2. The proposed method

To realistically represent the original scene we need to texture map the model. The simplest approach would be to use an image from the second pass as the texture, because the texture coordinates of feature points are easy to calculate. However the images are altered by the projected grid and cannot be use as a texture. As both video passes were performed with a hand-held camera, the models are not registered and could be at different scales. However, since they describe the same scene, we can align them automatically and then use an image from the first pass as a texture for the model generated from the second pass.

2 Detailed Description of Solution

Fig. 2 shows an overview of the method, and each step is explained in the following paragraphs. We start by describing the camera calibration, as this procedure is essential for other steps of the method to succeed. Then, the first and second video passes are discussed. Finally, the registration

and calculation of texture coordinates is described.

2.1 Intrinsic Camera Calibration

Virtually all methods that recover 3D information from images have to know how the points in the scene are mapped to the image. This mapping depends on two sets of parameters: the location of the camera in the scene and the camera's internal parameters. These parameters are obtained during the external and internal camera calibration respectively. The method presented here requires only intrinsic calibration and the camera's internal parameters do not change during the videotaping. The user performs this calibration only once for the camera used in this process.

The camera is described by a simple pin-hole model. In this model, the position of a 3D point (x, y, z) and its image $(\frac{U}{W}, \frac{V}{W})$ is described by the following equation:

$$\begin{vmatrix} U \\ V \\ W \end{vmatrix} = \begin{vmatrix} -\frac{f}{s_x} & 0 & C_x \\ 0 & -\frac{f}{s_y} & C_y \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} x \\ y \\ z \end{vmatrix}$$

The matrix contains all of the important parameters of the camera: the size of a single pixel (s_x, s_y), the effective focal length (f) and the image center (C_x, C_y). A standard method for camera calibration is placing a large number of targets in front of the camera, so they fill the camera image. Then the image is recorded and the locations of the targets in the image are found. The camera parameters are calculated based on the target's 3D world locations and the corresponding locations in the image

(Tsai 1987; Heikkila and Silven 1997).

2.2 First Pass

We videotape the scene with a hand-held video camera under natural lighting conditions, keeping the interesting areas of the scene in the camera view. We have to move the camera's position significantly, panning is not enough. In general we moved the camera approximately on an arc around the scene, always pointing it at the center of the scene.

2.2.1 Feature Selection and Feature Tracking

After digitizing the video sequence, the software finds the feature points in the first video frame. Features normally are of high contrast such as corners and edges. A plain wall on the other hand doesn't have features because it is of a slowly varying or uniform intensity.

The next step is to track the selected features across the video sequence. Because the images are recorded at the rate of 30 pictures per second, the camera viewpoint doesn't change dramatically from frame to frame. Therefore sequential images are very similar, and the corresponding features are nearly at the same locations in both images. In this case, the software can track a feature from frame to frame.

Our method uses Kanade-Lucas-Tomasi (Birchfield 1998) feature tracking software that combines the feature selection and tracking processes. The features are selected depending on how well they can be tracked. This algorithm also detects some occlusions and features that do not correspond

to points in the world.

No known tracking algorithm works perfectly. Features can become occluded, be lost by the tracker, or tracked incorrectly. An occluded feature ought to be dropped by the tracker, but it is difficult to recognize every occlusion automatically. Incorrect tracking occurs when there are similar areas close together and an incorrect one is assumed to be a match. This introduces a false match. Another case is when a feature appears differently in successive images due to lighting or shadows, and the tracker is not able to match it correctly. A different problem is false features, such as edge partially occluding another edge that is farther away or a specular reflection on a surface. These are not real 3D features, because their locations in 3D change during videotaping. Because of the above problems, some human intervention in removing erroneous matches may be necessary. Usually this involves no more than 10% of the tracked features. A visualization which connects the tracked features by lines is created so that errors in tracking are easy to identify.

Tracking features from frame to frame in a video sequence, we obtain the locations of the features in the first frame and their corresponding locations in the last frame. These corresponding locations are from images with significantly different viewpoints which enables a more accurate position determination than would be possible from correspondence in adjacent frames.

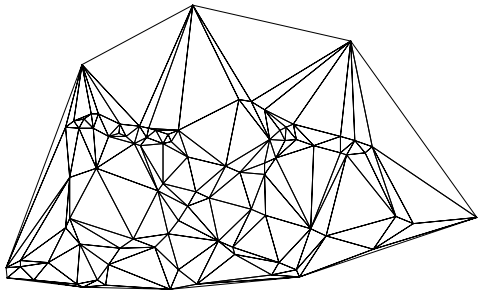

Figure 3. An example of a mesh generated for a 2D point set.

2.2.2 Structure from Correspondence

Based on the correspondence information and the camera's intrinsic properties, a structure-from-motion software (Zhang 1996) recovers 3D locations of feature points. The basic requirement is it to have at least 8 corresponding features. In practice, at least 15-20 corresponding points are necessary to obtain meaningful data.

The recovered 3D points are in a specific coordinate system. The origin of this system is the location of the initial camera location. Its axes are aligned with the camera axes. The recovered 3D points are at a specific scale factor. Based on a set of 2D images alone, we cannot determine the absolute sizes of the photographed objects. However, this is not a problem as there are objects of known size in the image.

2.2.3 Mesh Generation

So far we have obtained a set of 3D points, but representing a scene as a set of points is not very compelling to a viewer. We want to render the scene as a set of triangles or, preferably, texture mapped triangles. In

order to connect the 3D points into a set of triangles we need to consider the relationship between features in the first image from the video. The features in this image are 2D points, and we mesh them using Delauney triangulation (Fig. 3). The triangles created in this fashion are 2D primitives, laying on the same plane and not overlapping each other. Each of these 2D features has its 3D equivalent after the structure-from-correspondence recovery. We use the 2D connectivity obtained with the Delauney triangulation as the way to mesh 3D points.

The procedure described so far would suffice if enough feature points were evenly distributed in the scene. However, this is rarely the case. Often there are large featureless areas in scenes; for example, plain walls. These areas are not going to be part of the constructed model, because they don't contain features. These problems are the motivation for the second pass videotaping.

2.3 Second Pass

In this pass, we artificially create a large number of evenly distributed features in the scene using structured light. We do this by projecting a grid of lines with an overhead projector onto the scene (Fig. 1). The grid has a black background and white lines. In this manner we create artificial features in all areas of the scene, even in areas which were featureless during the first pass. The intersections of the grid lines show up very well on nearly all surfaces and it is easy for the software to select and track the crossing of such lines. The process then becomes very similar to the first pass. We

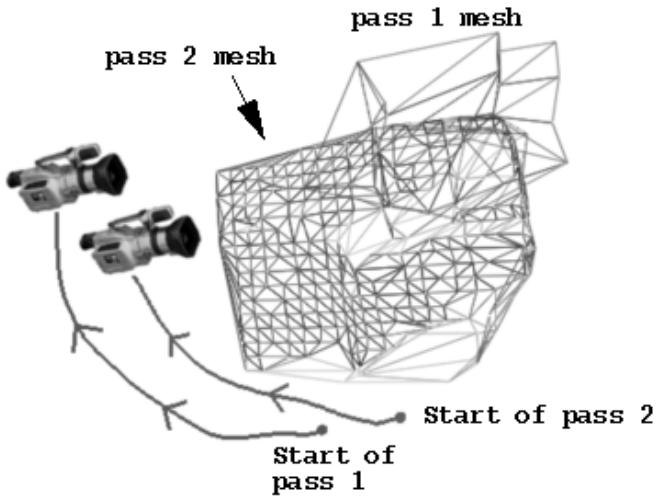

Figure 4. Camera paths and reconstructed 3D meshes.

calculate the 3D locations of the tracked features based on correspondence and camera parameters and we mesh these 3D points using the Delauney triangulation of corresponding 2 dimensional features. This generates a dense model of the scene with vertices evenly distributed in all areas of the scene.

2.4 Registration

After performing both passes, we obtain two 3D meshes. The first mesh is sparse but contains 3D point locations as well as the corresponding texture. The second mesh is much more detailed, but has no associated texture. The goal now is to combine the information from these 2 passes and obtain a dense texture mapped 3D mesh.

Both video passes were performed with a hand-held camera, so the camera paths are different (Fig. 4). The origin of each coordinate system is

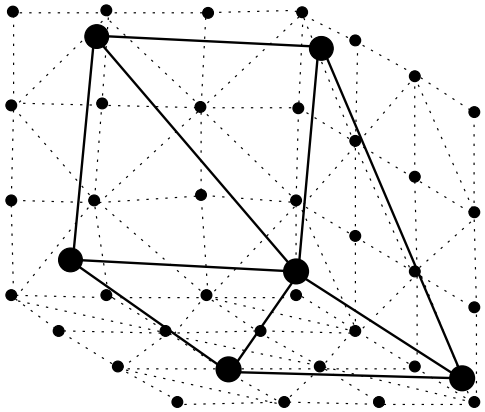

Figure 5. Two meshes describing the same object: 3 faces of a cube. One of the meshes is more detailed, the other one has fewer vertices.

the camera position upon taking the first frame. The scale of each coordinate system is chosen arbitrarily as the length of the camera path in each pass. The two models from the two passes have different scales and origins of their coordinate system. However, both structures describe the same scene, and we can transform the second mesh in such a way that the meshes become registered. This transformation T is going to contain a translation (T_x, T_y, T_z), rotation (R_x, R_y, R_z) and a scale (s).

Assume we have 2 meshes: M_1 and M_2, each described by a set of points $(p_1, p_2, \ldots, p_n)$ and $(q_1, q_2, \ldots, q_m)$ respectively. Suppose n is significantly larger then m similar to the situation in Fig. 5. The sum of the distances from the points in the M1 to surfaces of M2 is defined as:

$$\mathrm{Q(M_1, M_2)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{dist(p_i, M_2)} \tag{1}$$

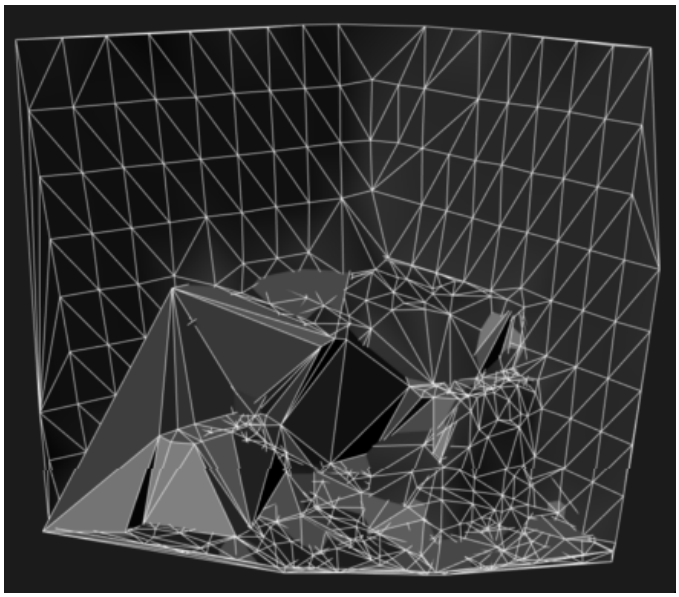

Figure 6. Meshes are registered. The texture from the first pass can be applied to the dense second mesh.

We are going to use the function Q as a way to measure how well the meshes are registered. If they are poorly registered then the value of Q is going to be large. From the two video passes we have two meshes. We want to register the second, more detailed mesh in such a way that it becomes aligned with the first one. In mathematical terms, we want to minimize the value of Q with respect to the transformation T, which is applied to the second mesh:

$$\min_{T} Q(M_1, T(M_2)) \tag{2}$$

If we are able to find a T for which the value of Q is very small, this would mean the meshes M_1 and $T(M_2)$ are aligned as in Fig. 6. Because the

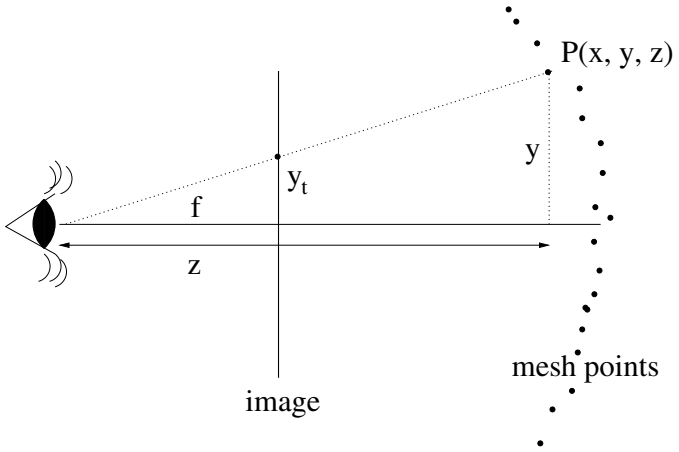

Figure 7. After calculating the transformation to register the two matches, that transformation can be used to register the texture coordinates.

transformation T contains a translation (T_x, T_y, T_z), rotation (R_x, R_y, R_z) and a scale (s), the minimization is a 7 dimensional task. To perform it, we use simple minimization routines like Multidimensional Downhill Simplex Method and Direct Set (Powell's) Method. There is no guarantee that this method will find the global minimum. However, when the algorithms are started from different initial guesses, the smallest result is often very close to the global minimum.

2.5 Texture Coordinate Calculations

In the registration process, we have found a transformation T from the coordinate system of the second pass to the coordinate system of the first one. When this transformation is applied to the second pass mesh, the mesh is placed in the first pass's coordinate system. The origin of this system

is the first pass initial camera location (see section 2.2.2). The relation between the first image from the video and the mesh is now straightforward, as in Fig. 7. In this situation we can calculate the texture coordinates of the second mesh's 3D points. For a point $P(x, y, z)$ in the mesh, its corresponding point in the image is x_t, y_t. The relation between these points is as follows:

$$x_t = f\frac{x}{z} \qquad y_t = f\frac{y}{z} \tag{3}$$

Values in these equations are expressed in world units. However, the texture coordinates (X_t, Y_t) are expressed in pixels. The size of a pixel in world units is (s_x, s_y). Therefore:

$$X_t = \frac{x_t}{s_x} = \frac{x}{z}\frac{f}{s_x} \qquad Y_t = \frac{y_t}{s_y} = \frac{y}{z}\frac{f}{s_y} \tag{4}$$

The terms $\frac{f}{s_y}$ and $\frac{f}{s_x}$ were calculated in the camera calibration process described in section 2.1. The values x, y, z specify the 3D location of a point, and they are known for each point in the mesh. Therefore we can calculate the texture coordinates (X_t, Y_t) for all points. In this fashion, we finally obtain the desired model: a dense texture mapped 3D mesh.

3 Evaluation

This method was applied to synthetic and real video sequences.

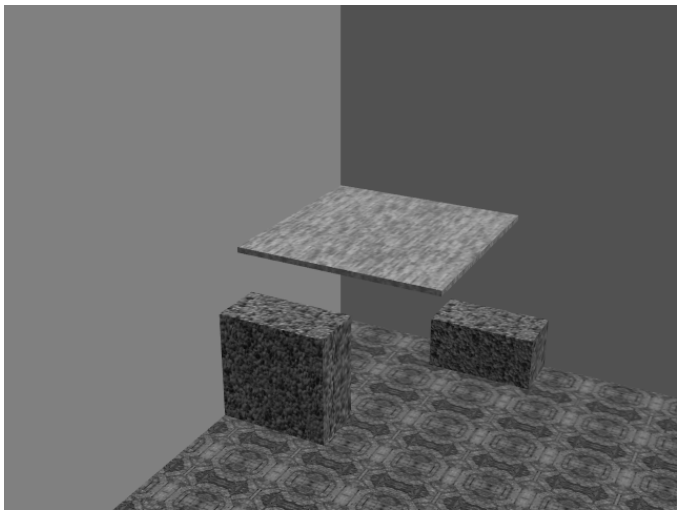

Figure 8. The model used in the synthetic tests.

3.1 Synthetic Scene

In this case the synthetic camera was moved around the model (Fig. 8) on two different curves. Camera calibration is critical to the method and it was found that 50 points were necessary to properly calibrate the camera. The feature tracking was perfect (no incorrect matches) therefore no human intervention was necessary. The mesh generated from the first pass contained 90 points, while the mesh from the second pass contained 350 points. These meshes were aligned within an average distance of 0.008 units, with very good texture alignment. The registration and texture coordinates calculations are automatic, so the whole process in the synthetic video case involved no user input.

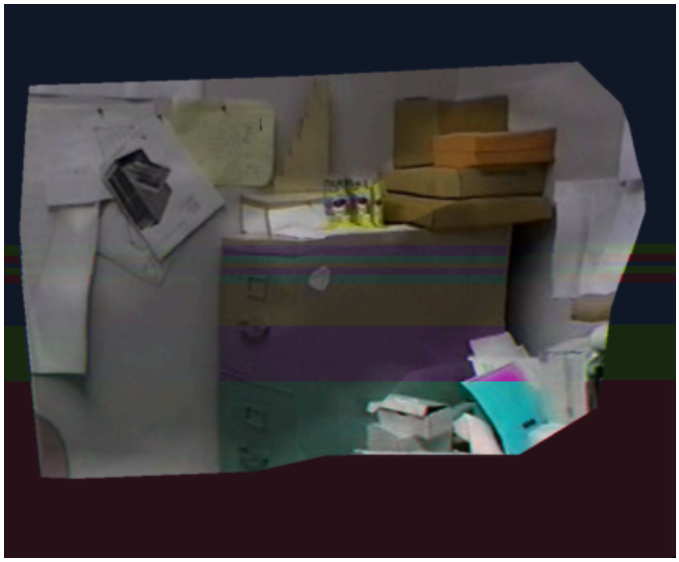

Figure 9. A model of a reconstructed office space, viewed from a new viewpoint.

3.2 Office

In order to test this technique, an office with drawers and boxes was videotaped. We used a Sony CCD-VX3 Hi8 video camera, which has 46 degrees horizontal FOV and 36 vertical FOV. The first video pass, under natural light, consisted of 113 frames. 25 features were selected and tracked automatically. A 3D mesh was created.

During the second pass, we projected a grid onto the scene, creating approximately a 3-inch-cell grid on the scene. This video pass consisted of 144 frames. 291 features (line crosses) were selected and tracked. 16 matches were obviously incorrect and were manually removed. During the registration phase, the Multidimensional Downhill Simplex routine found

a transformation for which the value of Q is very close to 0. Therefore the result is close to the global minimum (perfect registration). A texture mapped model seen from a new viewpoint is shown on Fig. 9.

3.3 Control Room

The size of the recovered model is limited with this method, because an interesting area of the scene has to be constantly visible during the recording. Using this simple approach, we are not able to construct a model that surrounds the user. However, we can construct several small models of parts of the scene, and then merge them to create a larger model. This technique was used in a control room example. The scene was divided into 3 parts, and the original technique was applied to each of these parts independently. A VR user then moved and scaled these models in such a way that they formed a single large model. The final model is shown in Fig. 10. We can also think of a more automatic way of merging these models. A user could select 3 corresponding points in 2 models, and based on this information models could be merged by software.

4 Current Limitations

There are several limitations to this method. Since structured light does not show up well on dark, non-reflective surfaces, the number of features in such an area of the model may be reduced. To overcome this, a more intense light source should be used. Also, the number of identified and tracked features is limited by the 640 x 480 pixel video image. It is hard

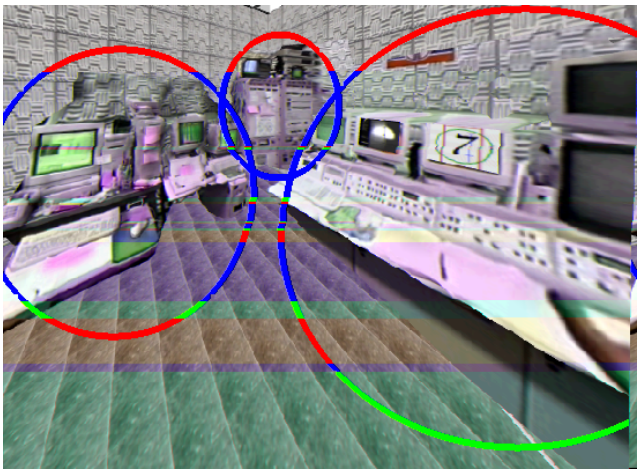

Figure 10. A control room which consists of 3 sections independently modeled. A user in a virtual environement interactively merged these models.

to achieve more than 2000 vertices in the model. Another problem with standard NTSC video is that a video image is composed of 2 interleaved fields. Each of these fields is taken at slightly different times. When the user moves fast during, the recording the images become jittered. This causes problems for feature tracking software. One solution is to use every other line-one field-from the image. This will solve the problem with jitters but at the same time will lower the vertical resolution, reducing the accuracy of the reconstructed scene. The other option is to remove "bad" frames from the sequence either manually or automatically. When a large number of features is dropped between 2 frames, that gives us a hint that the second frame could be jittered. In our tests a maximum of only 4-6 frames out of 100 were bad.

The feature points must be visible in each frame during the videotaping in order to track them. Occlusion could reduce the number of features and cause the reconstructed scene to be coarser.

The resolution of the projected grid implies that some small objects would not be a part of the reconstructed geometry. If none of the crossing grid lines shows up on a small object, then no corresponding features are created and the geometry is not captured. However, the object will still be visible in the final model because it is seen on the texture map. This works well if a small object is not far away from bigger objects. The texture gives the impression that the object is still part of the model. However, a small, isolated object, such as a thin pipe, far away from other surfaces in the scene, will appear to be incorrectly located in the final model.

When the projected grid doesn't create features just on an edge of an object, the edge is seen as a "soft edge" in the final model. This happens because the geometry edge and a corresponding texture edge are not in the same place. The remedy is to manually introduce few features just on the edge in the critical objects.

5 Conclusions and Future Work

The method proved to be a fast and simple way to obtain texture mapped models of real environments. It is convenient to be able to gather video with a hand-held camera in such a non-intrusive way. Very little training is required, because the camera path can be chosen by the user. The most work-intensive steps in the algorithm, feature selection and tracking, are

automatic. The models created were simple enough to be rendered in stereo at a high frame rate in the CAVE environment. When viewed in VR with user-centered perspective, these tests were a convincing experience for the viewers.

Enhancements are possible in both camera passes. The quality of the texture maps depends on the resolution of the first camera pass that was 640 by 480 in this work. A higher resolution camera could be used which would both generate higher resolution texture maps and a map with higher density.

Acknowledgments EVL receives major funding from the National Science Foundation (NSF), the Defense Advanced Research Projects Agency, and the US Department of Energy; specifically NSF awards CDA-9303433, CDA-9512272, NCR-9712283, CDA-9720351, and the NSF ASC PACI program. EVL also wishes to acknowledge support from Silicon Graphics, Inc. and Advanced Network and Services.

References

Birchfield S (1998) KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker. http://robotics.stanford.edu/~birch/klt/

Cyberware (1997) Cyberware Whole Body Scanning. http://www.cyberware.com/products/WholeBody.html

Debevec PE, Taylor CJ, Malik J (1996) Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. In: Computer Graphics (SIGGRAPH '96 Proceedings) pp 11-20

Gortler S, Grzeszczuk R, Szeliski R, Cohen M (1996) The Lumigraph. In: Computer Graphics (SIGGRAPH '96 Proceedings) pp 43-54

Heikkila J, Silven O (1997) A Four-step Camera Calibration Procedure with Implicit Image Correction. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition

Levoy M, Hanrahan P (1996) Light Field Rendering. In: Computer Graphics (SIGGRAPH '96 Proceedings) pp 31-42

Tsai RY (1987) A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. In: IEEE Journal of Robotics and Automation, pp 323-344

Zhang Z (1996) Structure from Motion. http://www-sop.inria.fr/robotvis/personnel/zzhang/software-SFM.html