PHOTOMETRIC SCENE RECONSTRUCTION FROM MULTIPLE CAMERA VIEWS

by

Grant Lipelt
Bachelors of Computer Science, University of North Dakota, 1998

A Project

Submitted to the Graduate Faculty

of the

University of North Dakota

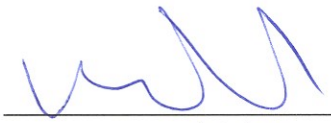
in partial fulfillment of the requirements

for the degree of

Masters of Science

Grand Forks, North Dakota
September
2003

This report, submitted by Grant Lipelt in partial fulfillment of the requirements for the Degree of Masters of Science in Computer Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

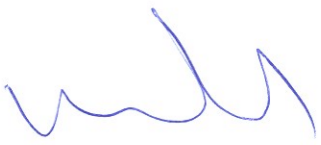


Advisor

9/20/03

Date

This project document meets the standards for appearance, conforms to the style and format requirements of the Computer Science Department of the University of Nortl Dakota, and is hereby approved.



Graduate Director

9/22/03

Date

PERMISSION

Title PHOTOMETRIC SCENE RECONSTRUCTION FROM MULTIPLE CAMERA VIEWS

Department Computer Science

Degree Master of Science

In presenting this report in partial fulfillment of the requirements for the graduate degree from the University of North Dakota, I agree that the Department of Computer Science shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my independent study work or, in his absence, by the Chairperson of the Department.

Signature 

Date 17 Sept 2003

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEGMENTS

The author wishes to express sincere appreciation to the University of North Dakota Computer Science Department for the years of dedication to its students.

ABSTRACT

Generation of a three-dimensional scene representation from a series of two-dimensional camera images is a classical problem of computer vision. Typically, 'scene reconstruction' techniques have been based on contour or features, but have been problematic in dense shape estimations. This is because accuracy is ensured only in places where features can be detected.

A novel scene reconstruction technique was developed at the University of Wisconsin by Steven Seitz and Charles Dyer. The 'Photorealistic Scene Reconstruction by Voxel Coloring' [1] technique first starts by working in a discretized three-dimensional scene space, by partitioning the scene into a discrete number of 'voxels'. Rather than approaching the problem as shape reconstruction, the approach follows color reconstruction. By assigning each voxel the associated color of the scene element, a realistic representation of the original scene can be constructed. Adjusting the resolution of the voxel size can then control the accuracy of the scene reconstruction.

CHAPTER I
Introduction

1.1 Problem Statement

The goal of this project is to construct a three-dimensional model which is representative of the actual world source scene. The generated model shall be geometrically representative of the scene, as well as color representative.

1.2 System Inputs/Outputs

The inputs to the system will include a series of camera images of the scene to be modeled, as well as calibration data files used to determine the camera pose for each of the given camera images. Additionally, user specified parameters, which will define the resolution, accuracy, and size of the model representation will be inputs to the system. The output of the system will be a three-dimensional representation of the world scene, either displayed to the screen, stored to secondary storage, or both.

The camera images will be taken of a static scene, from differing poses. Care to ensure sufficient scene color representation should be taken when generating these camera images. This includes, but is not limited to, minimizing reflective scene surfaces, and care in scene light sources. Shadows and reflective surfaces may result in significant color variations and limit the accuracy of the scene reconstruction.

CHAPTER II
Virtual Scene Construction

2.1 Discrete Virtual Scene Space

In order to begin construction of the virtual scene, we need to know the size of the scene, and the desired resolution of the virtual scene. The virtual scene can then be constructed by compartmentalizing a three-dimensional virtual scene space of the desire scene size into a set of voxels of the desired size (figure 1). The size of each voxel therefore defines the resolution of the generated scene model.

2.2 Voxel Classifications

Each of these voxels in the virtual scene will either map to dead space in the actual scene (an area where no tangible object exists), or to a tangible three-dimensional point that is an element of the scene. Determining if each voxel maps to a tangible or intangible point in the scene, voxel classification will be addressed in the next section.

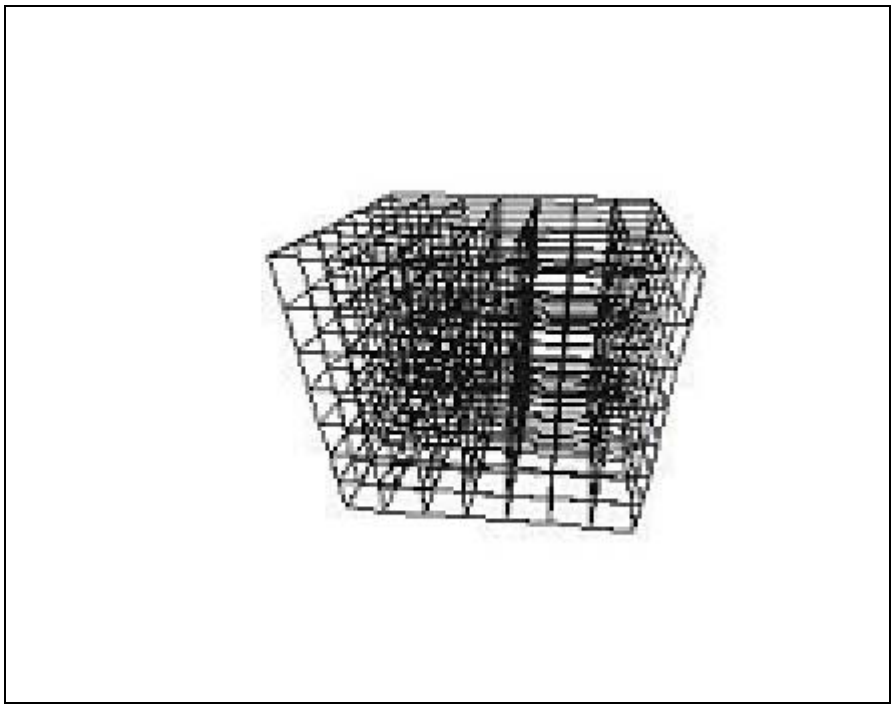


Figure 1

CHAPTER III
Voxel Classification

3.1 Voxel Classification Definition

In its simplest form, classification of each voxel as valid (part of the scene) or invalid (part of dead space) is done by mapping the voxel to the pixel(s) in each scene image. If the pixels are all comparably colored, then the voxel is considered to be valid. If the pixels vary significantly in color, then the voxel is considered to be invalid. Classifying the entire set of voxels will result in a subset of valid voxels and a subset of invalid voxels. Valid voxels comprise tangible elements of the actual source scene, invalid voxels map to dead space in the source scene. The valid voxel subset therefore would be representative of a three-dimensional model of the source scene. If each valid voxel stores color (comprised of the mean color of the corresponding image pixels) you will be left with a photometric three-dimensional scene representative model (refer to Algorithm 1).

3.2 Associating 3D Voxel Coordinates to 2D Camera Image Coordinates

The most significant step in this process is mapping the three-dimensional voxel coordinate to a two-dimensional image coordinate. For this, the project utilized the Tsai Camera Calibration technique [2] which provides methods for mapping three-dimensional world coordinates to two-dimensional image coordinates. This will be expanded on in the following section.

3.3 Algorithm 1

```
For each image in scene image set loop
        For each voxel in voxel list loop
          Map voxel world-coordinates to the set P(I) image coordinates in current
            camera image
          Add contribution of pixel P(I) to color mean and color standard deviation
        End loop –voxel
End loop –image

For each voxel in voxel list loop
        If voxel color standard deviation < standard deviation tolerance then
                Mark voxel as valid and assign color to value of color mean
        Else
                Mark voxel as invalid
        End if
End loop
```

CHAPTER IV
Tsai Camera Calibration

4.1 Use

The Tsai Camera Calibration was used for coordinate transformations, from the virtual scene world coordinates (voxel coordinates), to the camera image coordinate system. The required inputs for this technique are input data files which map known world coordinates to known image coordinates, for each of the camera images of the scene.

4.2 Camera Calibration File Format

Each image calibration data file is of the form

Wx_0 Wy_0 Wz_0 Ix_0 Iy_0
.
.
.
Wx_n Wy_n Wz_n Ix_n Iy_n

Each consisting of a point in the scene in world coordinates (Wx, Wy, Wz), followed by the associated image coordinates (Ix, Iy) where the world coordinate is found in the camera image.

The results of this project were based using manually created calibration data files, manually mapping the world coordinate to image coordinates and creating the calibration data file. This proved to be quite time-consuming, and limited the number of reconstructed scenes as well as the number of input camera image files.

4.3 Manual Generation of Calibration Data Files

The colorized calibration grid shown in each image supports the known world coordinates. The crosshairs identify the exact center of each colorized disk. These crosshairs define a two-dimensional planar world coordinate system. This coordinate system is stored in a file grid.wc in the directory which the camera images are stored. Each of these visible crosshairs are manually located in the camera imageX.ppm and stored in a corresponding imageX.ic file. Utilizing the Unix paste command then supported generating the necessary calibration date file imageX.dat by pasting the image coordinate file to the right of the grid world coordinate file.

4.4 Automated Generation of Calibration Data Files

An initial side-effort for this project was to automate the generation of the camera calibration files to remove/reduce the manual effort of creating the calibration data files which can prove to require significant time. While I believe there to be potential for completing this technique, it proved to be more difficult than expected. The primary hurdle which stands in the way is a general algorithm for locating the disks of known color while filtering out the regions of the scene of similar color. The disk coloring scheme identifies the unique world coordinate and is loosely based on work by Bruce Culbertson [3].

CHAPTER V
Simple Scene Reconstruction Results

5.1 Initial Reconstruction Results

Reconstruction of a simple scene shown in figure 2 is demonstrated in figure 3. This scene reconstruction works solely because of the lack of occlusions in the scene, since the scene is completely coplanar. Since typical scenes will not be entirely coplanar, we need to revisit our voxel classification method to allow occlusions. This will be the focus of the following section.

Figure 2

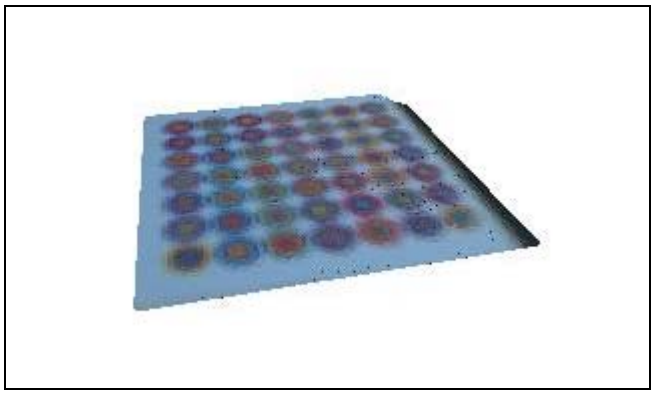


Figure 3

CHAPTER VI
Voxel Classification Revisited

6.1 Voxel Classification Revisited

We have so far addressed voxel classification in its simplest of forms, those in which the scene is entirely coplanar. This form will only work in the absence of occluding voxels that make up the scene (e.g. planar scene in figure 2). Because typical scenes are not planar, and will result in occlusions, we must readdress our classification technique.

6.2 Occlusions

To clarify the issue of occlusions, consider the image projection of the Rubik's© cube in figure 4. Voxels that comprise the opposed side of the cube will map to similar pixels that the facing cube side will map to. Since the opposed Rubik's© cube sides colors differ, the virtual scene voxels will be invalidated due to significant color variations between the scene images from facing camera poses. We must therefore consider that not all voxels that map to the image coordinate system are visible from all camera perspectives. Some voxels are occluded from the camera view by other voxels. If we revisit the voxel classification algorithm and update it to only take into consideration images in which the voxel is not occluded, the scene reconstruction technique will work for the general case. Refer to the updated reconstruction pseudocode (algorithm 2).

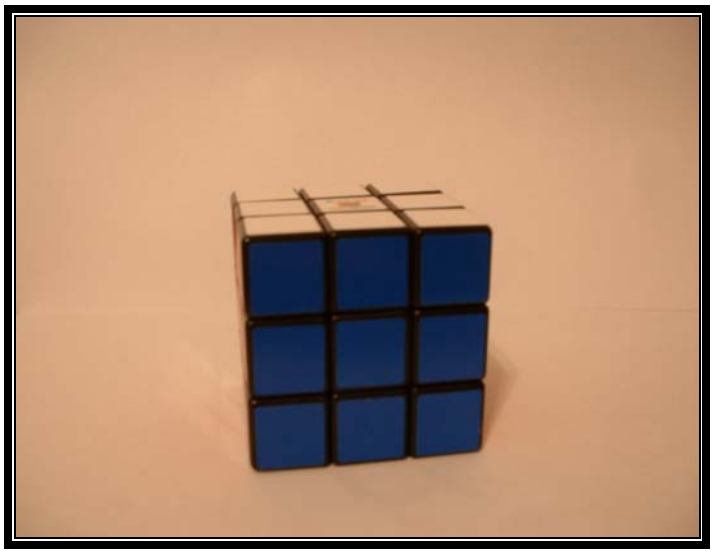


Figure 4

6.3 Algorithm 2

```
For each image in scene images loop
        Sort voxel list with respect to distance from camera pose in ascending order
        Clear image occlusion bit mask

        For each voxel in voxel list loop
          Map voxel world-coordinates to the set P(I) image coordinates in current
            camera image
          If bit mask pixels P(I) are not occluded then
            Add contribution of pixel P(I) to color mean and color standard deviation
            Mark bit mask pixels P(I) as occluded
          End if
        End loop
End loop

For each voxel in voxel list loop
        If voxel color standard deviation < standard deviation tolerance then
                Mark voxel as valid and assign color to value of color mean
        Else
                Mark voxel as invalid
        End if
End loop
```

CHAPTER VII
Final Results

7.1 Project Results

We've completed identifying the general-purpose scene reconstruction algorithm utilized in this project. This section demonstrates the results encountered in the project. The following table identifies the key variables used in the scene reconstruction, with example camera source images used in the scene reconstruction as well as a screen snapshot of the reconstructed model.

Model Name	**Voxel Size**	**Images**	**Scene Size**	**Tolerance**	**Model Figure**	**Sample Source Camera Image**
DCIM-A/100_FUJI/MoonMuncher/scene—vs0.10-i8-t20.0-sz2.0.mdl	0.10 mm	DCIM-A/100_FUJI/MoonMuncher/dscf000[1-8].ppm	2.0 mm	20.0	Figure 5	Figure 6
DCIM-A/100_FUJI/Rubiks/scene—vs0.10-i16-t20.0-sz1.75.mdl	0.10 mm	DCIM-A/100_FUJI/Rubiks/dscf00[1-16].ppm	1.75 mm	20.0	Figure 7	Figure 8

Figure 5

Figure 6

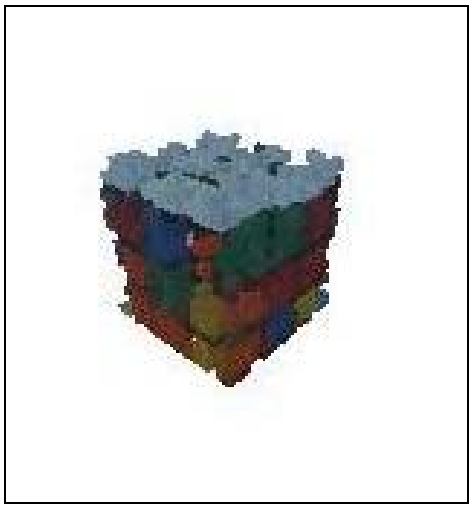


Figure 7

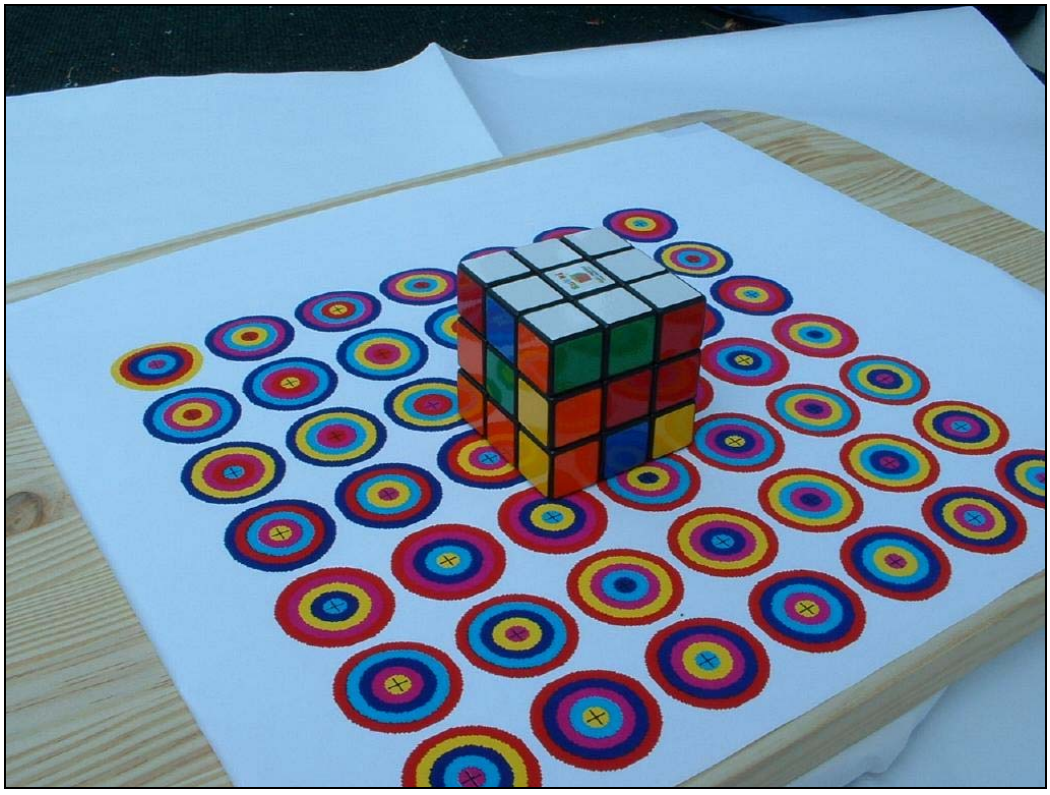


Figure 8

APPENDIX
Software Installation Instructions

This software was designed and tested on an Intel Pentium P166 based Compaq PC, running Redhat Linux 7.1.

1. Install Mesa libraries per instructions located at http://www.mesa3d.org/download.html.
2. Install GNAT compiler and runtime libraries in the typical manner for rpm package installation.
3. Copy the voxelReconstruction subdirectory structure to the desired location
    a. cd Tsai-Method-v3.0b3
    b. run ‘make’
    c. cd ..
4. run ‘make’
5. The reconstruct executable should now be successfully built. Consider running one of the software usage examples.

Software Usage

./reconstruct –h

This command will display a short list of command arguments as well as a brief description of the expected behavior.

./reconstruct –b

This command specifies batch mode, running in this mode will not display the scene regeneration in the display manager. Utilizing this mode along with the store model mode can be useful for executing a series of scene reconstructions in a batch.

./reconstruct –t <tolerance>

This command specifies the standard deviation tolerance used in the scene reconstruction. The argument is expected to be in floating point notation.

./reconstruct –s <filename>

This command will store the resultant scene reconstruction model as the specified filename.

./reconstruct –r <filename>

This command will retrieve the scene reconstruction model specified by the file and display the regeneration in the display manager. This is often useful for examining the results of previously regenerated scene models.

./reconstruct –fp <input filename prefix>

This command allows specifying the filename path for the reconstruction input camera images and coordinate transformation data files.

./reconstruct –v <voxel size>

This command allows specifying the size of the voxel to be used in the regeneration. This is used to specify the reconstruction resolution of the scene reconstruction. The argument is expected to be in floating point notation.

./reconstruct –sz <scene size>

This command allows specifying the size of the scene. The scene is generated of specified size contained in a 3D cube. The argument is expected to be in floating point notation.

./reconstruct –mc

This command allows marking all calibration points in the camera input file images and outputting them to “out*ppm” files in the current working directory.

This is helpful to isolate camera calibration errors and confirm the world coordinate to image coordinate transformations are correct.

./reconstruct –m

This command will create a scene mosaic, by mapping the camera input images to the virtual scene. This is useful to ensure that the camera angles fully encompass the scene, as well as confirm that camera calibration is correct.

./reconstruct –I <num images>

This command specifies the number of camera images to use in the scene reconstruction. Specifying 6 images will result in the software loading camera calibration data files <input filename prefix>000[1-6].dat and camera images <input filename prefix>000[1-6].ppm. This command is intended to be used in conjunction with the –fp specifier. The argument is expected to be in integer notation in range of positive integers.

Examples w/Description

./reconstruct –fp DCIM-A/100_FUJI/dscf –i 5 –v 0.10 –sz 3.0 –t 20.0 –s model1.mdl

Will result in reconstructing a scene of size 3.0 mm with a voxel size of 0.10 mm, and store the resultant model as model1.mdl in the current working directory. The 5 input camera images and camera calibration files DCIM-A/100_FUJI/dscf[1-5].dat and DCIM-A/100_FUJI/dscf[1-5].ppm will be used in the reconstruction.

./reconstruct -b –fp DCIM-A/100_FUJI/dscf –i 5 –v 0.10 –sz 3.0 –t 20.0 –s model1.mdl

Similar behavior as above can be expected, but specified in batch mode will result in no final displaying of the scene image, the software execution will terminate after storing the model.

./reconstruct -m –fp DCIM-A/100_FUJI/dscf –i 5 –v 0.10 –sz 3.0 –t 20.0

Will result in mapping the 5 images to the scene model voxels (generating a scene mosaic) and display the results in the display manager. Inconsistencies in the mapping may be a result of improper camera calibration.

./reconstruct -m –fp DCIM-A/100_FUJI/dscf –i 5 –v 0.10 –sz 3.0 –t 20.0 –s model1.mdl

Similar behavior as above, but prior to displaying the results the model is stored in the specified model name.

./reconstruct -b –mc -fp DCIM-A/100_FUJI/dscf –i 5

Will result in reading in the specified camera calibration files and camera image files, will mark each of the grid world coordinates, and write the results to output image files out[1-5].ppm. The user can then confirm the camera calibration is successful for each camera image by reviewing the output files.

EQUIPMENT USED

1. FujiFilm FinePix 2650 Digital Camera 2.0 Mpixel
2. Hp Design Jet 755CM Plotter
3. Camera Tripod
4. Kitchen LazySusan Turntable

The equipment utilized in this project is easily attained, within a modest budget. The only exclusion to this is the Hp Design Jet [2] plotter. Manual camera world coordinate to image coordinate mapping makes this equipment unnecessary. Additionally, the turntable and tripod were used solely for convenience, since the pose of the camera is determined by the camera calibration the need for consistent camera positioning is unnecessary, but it was found useful for keeping the entire scene within the camera frame.

REFERENCES

[1] Steven M. Seitz and Charles R. Dyer, Photorealistic Scene Reconstruction by Voxel Coloring, IEEE Proceedings of Computer Vision and Pattern Recognition Conference (CVPR 97), pp. 1067-1073, 1997.

[2] Roger Tsai, Tsai Camera Calibration Software, http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html

[3] Bruce Culbertson, Chromaglyphs for Pose Determination, http://www.hpl.hp/com/personal/Bruce_Culbertson/ibr98/chromagl.htm

[4] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck, Machine Vision, McGraw-Hill, 1995.