SIGGRAPH 99
26th International Conference
on Computer Graphics and
Interactive Techniques

COURSE NOTES 8
CASE STUDY: Scanning Michelangelo's Florentine Pieta'

Sunday, August 9, 1999
Two Hour Tutorial

ORGANIZER
Holly Rushmeier
IBM TJ Watson Research Center

LECTURERS

Fausto Bernardini
Joshua Mittleman
Holly Rushmeier

IBM TJ Watson Research Center

ABSTRACT

We describe a recent project to create a 3D digital model of Michelangelo's Florentine Pieta'. The emphasis is on the practical issues such as equipment selection and modification, the planning of data acquisition, dealing with the constraints of the museum environment, overcoming problems encountered with "real" rather than idealized data, and presenting the model in a form that is suitable for the art historian who is the end user.

Art historian Jack Wasserman, working with IBM, initiated a project to create a digital model of Michelangelo's Florentine Pieta' to assist in a scholarly study. In the course of the project, we encountered many practical problems related to the size and topology of the work, and with completing the project within constraints of time, budget and the access allowed by the museum. While we have and will continue to publish papers on the individual new methods we have developed in the in course of solving various problems, a typical technical paper or presentation does not allow for the discussion of many important practical issues. We expect this course to be of interest to practitioners interested in acquiring digital models for computer graphics applications, end users who are interested in understanding what quality can be expected from acquired models, and researchers interested in finding research opportunities in the "gaps" in current acquisition methods.

ABOUT THE SPEAKERS

Fausto Bernardini
30 Saw Mill River Road
Hawthorne, NY 10532
914/784-7475 (voice)
914/784-7667 (fax)
fausto@watson.ibm.com

Fausto Bernardini is a research staff member at the IBM T.J. Watson Research Center. He received a Laurea degree in Electrical Engineering from the University “La Sapienza” of Rome in 1990, and a Ph.D. in Computer Science from Purdue University in 1996. He has written numerous papers on the reconstruction of shape from 3D scans, as well as other topics in computer graphics and geometry.

Joshua Mittleman
30 Saw Mill River Road
Hawthorne, NY 10532
914/784-7671 (voice)
914/784-7667 (fax)
mittle@watson.ibm.com

Joshua Mittleman is a senior programmer at the IBM T. J. Watson Research Center. He received a B.A.(1983) in Chemistry from Cornell University and an M.A. (1996) in Computer Science from Rutgers University. Since 1990 he has been working on algorithms for interactive viewing of very complex geometric models.

Holly Rushmeier
30 Saw Mill River Road
Hawthorne, NY 10532
914/784-7252 (voice)
914/784-7667 (fax)
holly@watson.ibm.com

Holly Rushmeier is a research staff member at the IBM TJ Watson Research Center. She received the BS(1977), MS(1986) and PhD(1988) degrees in Mechanical Engineering from Cornell University. Since receiving the PhD, she has held positions at Georgia Tech, and at the National Institute of Standards and Technology. In 1990, she was selected as a

National Science Foundation Presidential Young Investigator. In 1996, she served as the Papers chair for the ACM SIGGRAPH conference and in 1998 the Papers co-chair for the IEEE Visualization conference. She is currently Editor-in-Chief of ACM Transactions in Graphics. She has published numerous papers in the areas of data visualization, computer graphics image synthesis and thermal sciences. In the area of global illumination she has worked on the problems of comparing real and synthetic images, imaging participating media, and combining ray tracing and radiosity methods. Most recently she has worked on accurate tone reproduction for high dynamic range images, and systems for acquiring physical data for realistic rendering.

SYLLABUS

Project Definition and Requirements
Holly Rushmeier
- notes section 2

Project Planning and Hardware Selection
Fausto Bernardini
- notes section 3

Working on Site
Holly Rushmeier
- notes section 4

Software Development for Model Construction
Holly Rushmeier and Fausto Bernardini
- notes section 5

Presenting the Model to the User
Joshua Mittleman
– notes section 6

TABLE OF CONTENTS

CASE STUDY:
Scanning Michelangelo's Florentine Pieta`

Project definition and requirements

The Florentine Pieta` is the second of Michelangelo's Pieta`s, the first being the famous work on view in the Vatican. Pieta` is a general term referring to Mary grieving over Christ's body. The scene shown in this work is actually the deposition of Christ's body into the tomb. The central figure is Christ, the other figures are believed to be Mary Magdalene on the left, Mary the mother of Jesus on the right, and in the back is Nicodemus.

Created by Michelangelo late in his life ~ 1550

Michelangelo broke off pieces of the statue, repaired by Calcagni ~ 1555-56

Placed outside, in a basement 1562-1721

Placed in the Duomo 1721-1980

Currently in the Museum of the Duomo, Florence, Italy

This work is being studied by art historian Jack Wasserman.
There are a number of unique features of the work and its history that lead him to believe that a digital three dimensional model would be useful. In particular, the statue as it can be seen today has been repaired. Michelangelo himself broke off several pieces. While he was still alive, he gave permission for the work to be repaired and sold. Why Michelangelo broke the statue is still an open question. Being able to view the work with pieces removed may provide insight into the artist's motivation.

A Comprehensive Study
By Art Historian Jack Wasserman:

X-rays
View Under Ultraviolet
Historical Record
Religious Significance
Digital Model 

The digital model is just one part of a comprehensive study
being conducted by Dr. Wasserman. He is synthesizing information from many historical documents and scientific
studies. Just seeing the cracks on the outside of the work can't tell us what was broken, but high power X-rays can reveal where metal pins were inserted. Just seeing the statue with pieces removed doesn't provide the naive viewer with insight about Michelangelo's motivation.
It is an aid to an historian who also knows the rest of Michelangelo's work, the environment
he worked in and the contemporary records of events.

From a technical perspective:

Challenging project

Opportunities for developing new algorithms

But what is our final goal?

There are many possible motivations for acquiring a 3-D model -- making reproductions, distributing for educational
purposes, documenting for conservation efforts. Different applications have different requirements. Making small scale reproductions would have relatively low demands on
the resolution required. Documentation for conservation efforts would require detailed measurement of material properties -- perhaps with imaging outside the visible range.
Our particular goal was to provide tools to the art historian
to answer questions about form and technique.

Challenge:

Can an expert, with free access to the actual work, learn anything from a digital model?

The digital model can give the average person many views
of the statue that wouldn't be available to the everyday visitor at the museum. However, a professional art historian
has significantly greater access to the physical work. It is particularly challenging to see if we can provide him with views he has never seen before

Tools we can offer:

- Controlled views
- Impossible views
- Precise measurements
- Other environments
- Partial geometry
- On demand details

We can't guarantee that the model will provide dramatic new insights. All we can do is provide tools to the art historian. Our project is successful if he finds these tools useful.

- Controlled views

(from cat scan of scale model)

With the digital model the art historian can examine the varying appearance of the work along precisely defined, and repeatable, paths. This may provide insight into how the artist expected the work to be viewed, e.g. at what height was it to be displayed?

- Impossible views

impossible in museum (from cat scan scale model)

impossible view point (from July scan)

There are views that are impossible, because you would have to dangle from the ceiling of the museum, or would need to be inside the statue. Obviously, the statue was not designed to be viewed from impossible points of view. It may provide insight though into how the artist thought about the work as it progressed.

- Controlled + Impossible Views

Precise measurements

from February, geometry only scan

Distances are distorted by perspective position in a photograph. It is impossible to permanently mark points on the statue from which measurements are taken.
The digital model can be marked and precisely measured.
The art historian can make verifiable statements about the
proportions of the figures, showing exactly how he measured the width of the head, or the length of the forearm.

- Other environments

pre-1980 Photograph

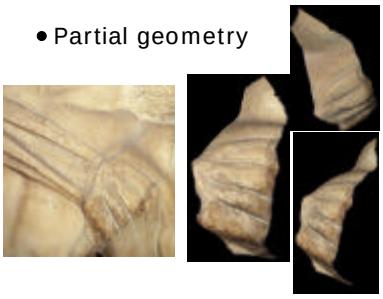
from cat scan
scale model

The statue has not always been in the museum, and was not intended for any of the locations where it actually has been displayed. Using the digital model we can place the statue in other environments, and view it under various types of lighting.

- Partial geometry

If just viewing the statue were enough, an image based representation would be enough. However to see the statue
after it was broken, we need to edit the three dimensional model.

- Partial geometry



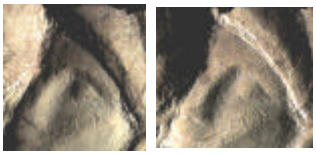
- On demand details

photograph

from July scan data

An interesting feature of this work is that it is not all smoothed and polished. Many of the original tool marks are visible. Different details are revealed depending on viewpoint and lighting. The detailed model allows interactive variation of lighting to get a sense of the small scale variations in geometry.

- On demand details

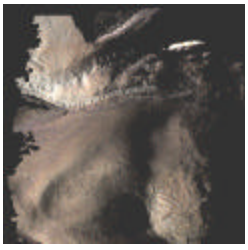


from July scan

This illustrates the variation in appearance under two different lighting conditions.

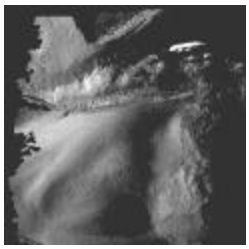
- On demand details

from July scan (texture map + bump map)



- On demand details

from July scan (bump map only)



With the digital model we can observe the statue as is, or we can separate out the effects of geometry and albedo
variations. The variations in geometry show the tool marks.
The variations in color and albedo show the effects of dirt and the application of coatings on the statue through the years.

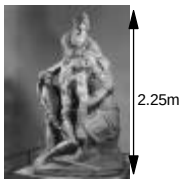
CASE STUDY:
Scanning Michelangelo's
Florentine Pieta`

Project planning
and hardware
selection

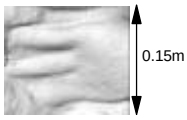
In this section we discuss some of the decisions that had to be made before going on site to collect data. We had to select and acquire hardware in a short period of time to stay on schedule for when Dr. Wasserman needed results. Our initial goal was to finish data acquisition by the middle of 1998.

Design Considerations: Length Scales

Examine on the
scale
of meters to study
proportion, design



Examine on the
scale
of millimeters to
study tool marks



The big technical challenge was the range of sizes needed. The statue is 2.25 m tall, and individual tool marks are one mm or less in size.
The detail is the Magdalene's hand on the back of the statue. You can see the small tool marks, particularly along the top of the top finger.

Tests with Cat Scanned Model

To get an idea of the effect of resolution, we had a small model from the museum cat scanned. Rendering this model
gave us an idea of what the real data would look like.

Design Considerations: Color

Dr. Wasserman does not have particular requirements for highly
accurate representation of the color of the statute. However, there are aspects of the statue and how it has been treated through the years that require capturing at least an estimate of the surface albedo. Here the back of the Magdalene is shown where the border between where a yellowish finish that was applied and where the statue was
left untreated is evident.

Limitations:

One scanner (budget)

Mobility (can not leave sitting in museum)

Size (to see into cavities)

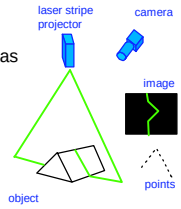
Time frame (book publication deadline)

Ideally, a number of different scanners would be used to capture different portions of the statue. However, we were constrained to invest in one device. We needed something small and mobil, because we would have to work in short periods in the museum after closing hours. We could not make any permanent modifications to the room in the museum . We needed to buy equipment essentially off the shelf, since our time frame did not allow for extensive fabrication of a custom scanner.

Scanning Technologies

- Laser range scanner
- Structured light
- Time-of-flight laser
- Multiple (video)cameras

3D-Scanners
ModelMaker



There are a variety of scanning products using different technologies currently available. Here we will discuss a few commercial products.
We mention specific brands and models for information only, and do not endorse any particular product.

Growing List of Commercial Scanners

Vitana -- http://www.vitana.com/
Hymarc -- http://www.hymarc.com/
3D Scanners -- http://www.3dscanners.com/
Perceptron -- http://www.perceptron.com/.
CyberOptics -- http://ww.cyberoptics.com/
Picza -- http://www.picza.com/
Cyra -- http://www.cyra.com/
Minolta -- http://www. minolta3d.com/
Digibotics -- http://www.digibotics.com/
Laser Design -- http://www.laserdesign.com/
Visual Interface -- http://www.visint.com/
In Harmony -- http://www.inharmonytech.com/

The list of commercial 3D scanners is growing. Some of the older companies emphasize scanning of parts for industrial inspection processes. Here, geometry only has been the main concern. Other scanners are oriented towards games and entertainment, and may have less accuracy,but include color. Some offer both accuracy and color acquisition, and sell scanning services to institutions such as museums as well as equipment and software.

Inexpensive Scanners:

Built for limited physical size:
desktop scanning

Many inexpensive scanners are built to scan small objects on the desktop, and are not readily modified to scan a large object.

Large Scale Scanners:
Expense, Accuracy, Lead Time, Availability

There are scanners for large (from human scale to building scale) objects. Some are very expensive. Some, while having excellent relative accuracy (on the order of mm's over scanning range of meters to a hundred meters) did not have the mm or less resolutions we needed. Large scale scanners are also frequently custom built, and we did not have the lead time.

Commercial scanners are developing quickly, and some fine scanners now on the market were simply not available in Dec. 97/Jan. 98 when we needed to make a decision.

Our Choice:

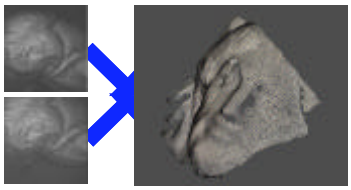
Virtuoso from Visual Interface

Availability
Cost
Size/Weight
Company Flexibility

The major factors for our choice were cost (<<$100,000), a camera could be sent to us immediately (there was one literally on the shelf), it had the mobility to allow us to view all parts of the statue, and, for reasonable fees, the company was able to be flexible at the time to make some modifications we needed.

Shape Capture

- Different viewpoints, same lighting pattern



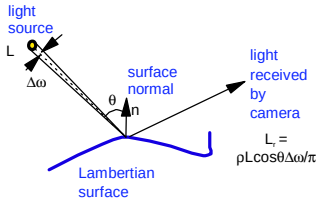
The basic principle that the camera uses is to project a pattern of stripes on the object, and then image the pattern with six cameras at calibrated positions. Using computer vision stereo algorithms, a 20 cm x 20 cm geometric mesh with 2 mm resolution can be obtained.

Major Problems:

Resolution and Color
2 mm not enough
Color images include lighting effects

Alignment
need to combine hundreds of meshes

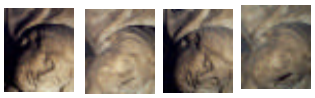
2mm is not enough:
Add Photometric Stereo



The basic principle of photometric stereo is to use one camera position and take pictures with lights in several calibrated positions. Using the relative radiances at a points for the different lights, a value for the surface normal can be computed. These normals can be computed at a resolution of 1mm or less using the color camera on the Virtuoso.

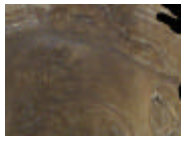
Details and Color Capture

- Same viewpoint, different lighting



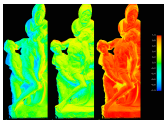
Fine details

"Flattened"color



Geometry + Bump Maps

Normal map computed from 5 color images



Original geometry & texture, different

Augmented with normal map

We tested the concept of Virtuoso + photometric in the lab with lights fixed to an optical table, and measuring a small model of the the statue obtained at the museum gift shop.

Shopping for Light Sources:

Need spatially even lighting

Lighting level compatible with camera (limits on exposure, apeture)

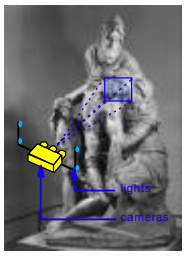
Controllable with camera software modificatoins

Low Weight

Version 1.0: Camera flashes

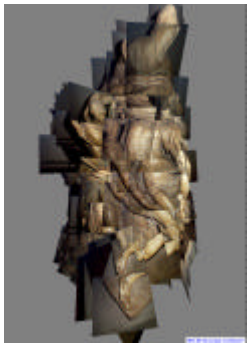
Finding appropriate lights and designing a rack to hold, and a triggering system coordinated with the Virtuoso camera took several weeks.

Final camera design



Alignment

Statue a composite of hundreds of meshes



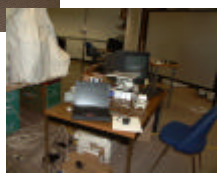
Because we would capture the goemetry in literally hundreds of pieces, we needed a strategy for how these pieces would be aligned to one another.

Rough alignment: Tracker on Camera

Fine alignment: Projected Laser Spots

To get a basic idea of where each mesh would go, we planned to use a tracker on the camera. We knew that this alone would not be enough, because small errors in camera location and orientation would result in large error in mesh location for a camera that is about 0.75 m from the statue. To get finer alignment, we used lasers to project spots on the statue as landmarks that could be used to align the meshes.

Practice



The scanning would not be automatic, but would involve manually moving the camera around the statue to cover all of the surface. A significant amount of practice is needed to learn to operate the camera efficiently.

CASE STUDY:
Scanning Michelangelo's
Florentine Pieta`

Working
on site

First trip, February 1998

Goals: scan basic geometry
for study of composition
and proportions

test equipment and identify
problems unique to the site
and the sculpture

photometric system not
yet designed

We were anxious to see if our scanning strategy would work in the museum. Would the surface finish of the actual piece be a problem? Was there enough space for what we wanted to do? Would the laser dots show up? Would the tracker work? How long would it take to set up and take down equipment each day? We wanted to gather some data early in the process.

Florence, Italy
Feb. 12-24, 1998

February was a good time to go for our first run-through, since the city is not as crowded with tourists. We learned that having a rental car was not useful because of restrictions on driving in the city. We needed to go out and buy various additional things -- such as saw horses to make a little collapsible desk to work on in the museum, but a taxi would have been fine. We also learned not to bother trying to shop Monday morning!

Taking delivery of scanning equipment at the Museum of the Opera del Duomo

A good reason for minimizing the equipment used is the difficulty of shipping from country to country. Special arrangements need to be made if equipment is not to be left or resold in the country.

Setting up lasers that project spots to facilitate aligning individual geometric patches

Camera was used with ladder and tripod, since piece is 1m to 3.25m above the floor.

The standard AC adapters on our computing equipment work for 100 to 240 volts and 50 -60 Hz. We only needed the simple adapters to go from US to Italian prongs on the plug. For the lasers however care had to be taken to correctly plug into the transformer.

Work was done at night when museum was closed. Equipment had to be disassembled and stored at the end of each session.

Jack Wasserman advised on what sections were of particular interest to capture in detail

It took a couple of days for us to become efficient at setting up and tearing down each night.

Lessons from February:

Tracker wouldn't work:
metal rail distorted magnetic field

Simpler is better:
limit computer connections,
no remote operation of camera,
just use Thinkpads in museum

Develop results immediately:
work on Intellistation during the day in hotel to get rough estimate of coverage

We spent a good deal of time one night trying to get the tracker to work -- since time was limited, we decided to work with out it rather than end up with no data while we tried to get it to work.

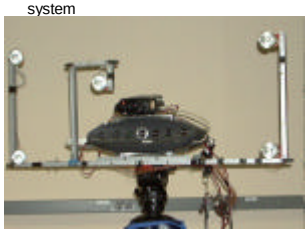
We originally had more elaborate plans to run the camera remotely from a laptop computer. Since each time this system failed there was a long recovery time, we also gave up on this idea.

Scanning is not "what you see is what you get" You need to spend hours each day processing and seeing what you got before scanning the next night.

Trip 2:
June/July

Goal:

Cover statue again, using camera + photometric system



We spent March, April and May processing February data and getting a rough alignment of the 500 or so meshes we brought back. We also spent considerable time designing and testing the photometric system.

The museum had no HVAC system. It had been terribly cold in February. By the end of May it was unbearably warm.

Major Problem:

Lighting System Failed

Camera flashes, working off "slave" on Virtuoso color camera, would not trigger reliably

The ideal lights we had found were camera flashes. However, no one sells flashes configured for our application. We depended on a "slave" that saw the color camera flash to trigger the flashes in the photometric system. The slave was not designed to be mounted directly in front of another flash and stopped triggering consistently. Also, when working in the museum we started having an electrical problem that would cause more than one flash to trigger at a time. After three valuable nights, we had taken practically no useful data.

New Lighting System
Built on Site

As a tourist you wouldn't notice it, but there are electrical and hardware stores in Florence, in the area of the Duomo. We went shopping for bulbs that could simply be turned on and off in turn, in place of flashes that had to be precisely synchronized with the camera. Fausto had come prepared with a soldering iron, and rebuilt the system with small, wide angle, halogen bulbs in place of the flashes.

On the second trip we were set up much better to process our data back at the hotel. We had an Intellistation that we could network with the ThinkPads we used in the museum.

Results from On Site:

Gigabytes of image data carried back on ThinkPad hard drives and a set of 1 gigabyte disks.

The work on site was a small fraction of the time we spent on this project. Planning the hardware, processing data and writing new software took far more time.

CASE STUDY:
Scanning
Michelangelo's
Florentine Pieta`

Software
Development
for
Model
Construction

Software Needed:

Alignment
Meshing
Photometric
Processing
Remapping

We could use the Virtuoso software to convert the sets of striped images into 20 cm x 20 cm meshes. We could also use the Virtuoso to get initial pairwise alignments. For the rest of the project however, all new software had to be developed.

Alignment:

Needed software customized for the particular data we had, with no initial data for camera positions.

Approach:

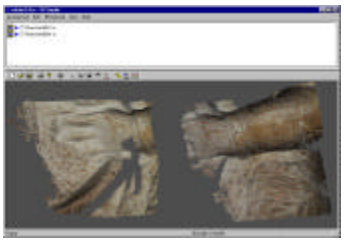
Tree of pairwise relationships

Matching laser dots

Marking overlapping patches

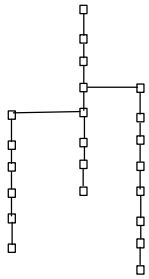
Iterative Closest Point

Pairwise Alignment in VI Studio



We kept notes as we took images to record the location of successive meshes. The Virtuoso comes with an interactive tool for aligning pairs of meshes by choosing 3 or more corresponding pairs of points on meshes.

Pairs of matrices form tree
of mesh relationships



Given alll the meshes related pairwise, we could build a tree of mesh connections. By successively applying transformations we could compute an approximate transformation for each mesh into a global coordinate system.

Improve "patches" by
matching
laser dots

Introduce artificial laser dots
to
connect patch to patch

During scanning we used lasers to project a grid of dots on the statue. For each camera position, we took a picture with the lasers on and all other lights off. These points could be identified on the meshes, and were used to automatically improve the initial crude registration.

Iterated Closest Point:

Method initially developed by Besl
for pair of meshes

For simultaneous registration

choose mesh N at random
freeze all other meshes
move N towards the closest points on frozen model

Refs. :
Iterative Closest Point:
Besl, Paul J. and Neil D. McKay, "A Method of Registration of 3-D Shapes" IEEE Trans. on PAMI Vol. 14, No. 2, pp. 239-256.
Simultaneous Registration:
Bergevin, Soucy, Gagnon, and Laurendeau, "Towards a General Multi-view Registration Technique," IEEE Trans. on PAMI, Vol. 18, No. 4, pp. 540-547.

Meshing

Problem:
Model larger than available meshing software can deal with.

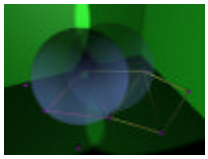
Can't fit data into memory.

Even if data fit, projected timing for meshing too long.

The Ball-Pivoting Algorithm

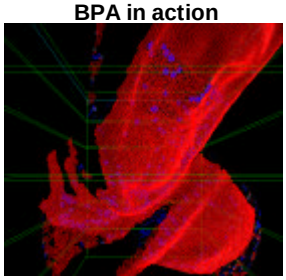
- Fast surface reconstruction from scans
  - Interpolating triangle mesh
  - Linear-time algorithm
- Robust
- Results
  - Real data: Pieta', Stanford repository
  - Generates 1M triangle mesh in 15 minutes on a PC
  - Out-of-core implementation

Ball-Pivoting



Starting with a seed triangle, we grow a triangle mesh. At each step we consider an edge at the boundary of the mesh. We "pivot" a ball of fixed radius around the edge until it hits a new data point. The edge and new point form a triangle that is added to the mesh.

BPA in action



The BPA computes a triangle mesh that interpolates the data points. Data can be loaded in slices, so that it is possible to process data sets that do not fit into memory.

We tested BPA on a variety of data sets. The sample data sets provided by Stanford University the Stanford 3D Scanning Repository at http://www.graphics.stanford.edu/ were very useful for testing real scanned data.

Photometric Processing:

Problem:
Computing colors and normals that are consistent with underlying geometry
and
with each other.

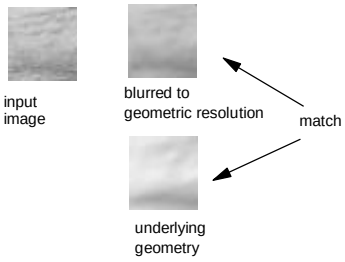
Problems with Photometric Data:

Lights not identical
Lights not isotropic
Temporal variations
Varying electrical power level
Short distances
Non-Lambertian Surfaces

The lights were particularly a problem since we had to use the new system built in Florence, rather than our original "ideal" system.

Approach:

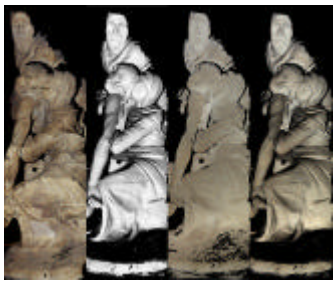
Use underlying geometry to adjust relative light levels in images



We were able to over come the uncertainties and variations in the new lighting system by making use of the 2mm resolution geometry in the course of computing the normals at a higher resolution.

Color Alignment

Do global alignment of "flattened" color images with a least squares solution equating the colors in laser dot locations in all images



Even after removing the effects of lighting from the colored images, there were discernible color variations mesh to mesh. Just blending meshes pairwise would give a splotchy effect. Instead, we formed a set of equations matching the colors at each pair of corresponding laser dots in overlapping meshes to solve for color adjustments globally.

Remapping

Impractical to use hundreds of texture and normals maps. Using full mesh, find new set of meshes that are each flat enough for a single texture.

CASE STUDY:
Scanning
Michelangelo's
Florentine Pieta`

Presenting
the Model
to the User

It is not enough to just build a 3-D model, it needs to be accessible to the user. The geometry alone is far too much data to work with interactively on a high end PC, let alone all of the color and normals maps.

Products:

Edited Mesh
Without pieces removed by
Michelangelo

Figures separated

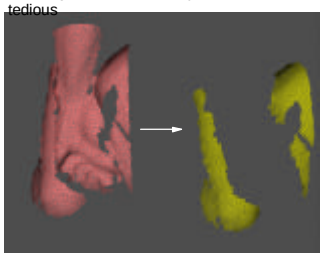
Interactive Viewer

We had two tasks for preparing the data for Dr. Wasserman. One was editing the model to show it with and without the breaks, and with the individual figures separated. The other was making a form of the model that was useful for examining the data interactively.

Mesh
Editing

Need to edit many times to provide
intermediate results

Working at mesh level very tedious



We made many generations of the model as we improved the alignment and meshing. We wanted to provide intermediate results to Dr. Wasserman, but we didn't want to re-edit the very detailed mesh every time we generated a new model.

Edit Once At Texture Level
Points included/excluded by
texture map color

We identified the broken pieces and individual figures by coloring the associated texture maps. In this manner, each point had an identifier for whether it was on a broken piece, and which figure it belonged to. Points could then be selected automatically to form edited versions of every generation of the model.

Interactive
Viewer

Problems:

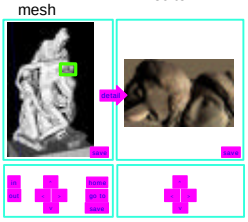
Model too large for interaction

Need simple controls for position and lighting

The model needed to be simplified for interactivity. A new viewer was needed specific to Dr. Wasserman's needs which demanded more functionality than a standard VRML browser, but less complexity than an engineering-oriented geometry viewer.

3d viewer for simplified mesh

detailed light editor



The result is a combination geometric and image-based viewer. The user browses the simplified textured 3D model, selects a region of interest, and generates a detailed view. The detailed view is computed by a back-end process, derived from the full database, to a resolution higher than the simplified geometry. The detail is displayed in a light editor.

Building the Light Editor

Working in Java

-- Importing Images

-- Performance

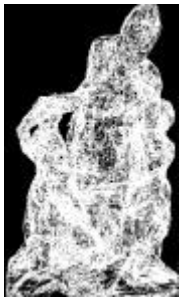
The light editor is based on observation of how Dr. Wasserman examined the actual statue with a flashlight. It is designed to facilitate examining very small surface features, such as the tool marks.

The light editor image is a standard RGB image plus a normal at each pixel.

The light editor was build in Java using standard AWT components. Image generation is implemented with ImageProducers and ImageConsumers.

Simplifying the Model

Simplification methods not designed for models that don't fit into memory!



The problem of simplifying the large model was solved by a trick originally proposed for parallelizing simplification:

Cut the model into pieces along a regular grid

Simplify each piece, leaving the boundary unchanged

At this point most of the model is simplified, but not the boundaries of the pieces.

Shift the grid by one-half grid cell in each dimension cut again.

Simplify each piece again, levaing the boundary unchanged as before.

Past the pieces together again

Now the entire model is simplified except for any boundary in the original model.

Simplify the entire model one last time to reduce any real boundaries.

All simplifications were performed using Andre` Gue`ziec's method with tolerances between 4 and 16 mm. With tolerance 8 mm at each stage, the simplified model has 81 K triangles, reduced from a version of the statue that had 6 million triangles.

The Ball-Pivoting Algorithm for Surface Reconstruction

Fausto Bernardini Joshua Mittleman Holly Rushmeier Cláudio Silva Gabriel Taubin

IBM T. J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Abstract

The Ball-Pivoting Algorithm (BPA) computes a triangle mesh interpolating a given point cloud. Typically the points are surface samples acquired with multiple range scans of an object. The principle of the BPA is very simple: Three points form a triangle if a ball of a user-specified radius ρ touches them without containing any other point. Starting with a seed triangle, the ball pivots around an edge (*i.e.* it revolves around the edge while keeping in contact with the edge's endpoints) until it touches another point, forming another triangle. The process continues until all reachable edges have been tried, and then starts from another seed triangle, until all points have been considered. We applied the BPA to datasets of millions of points representing actual scans of complex 3D objects. The relatively small amount of memory required by the BPA, its time efficiency, and the quality of the results obtained compare favorably with existing techniques.

Figure 1: Section of Michelangelo's Florentine Pietà, sample points and reconstruction with textures.

1 Introduction

Advances in 3D data-acquisition hardware have facilitated the more widespread use of scanning to document the geometry of physical objects for archival purposes or as a step in new product design.

A typical 3D data acquisition pipeline consists of the following steps (adapted from [17]):

Scanning: Acquisition of surface samples with a measurement device, such as a laser range scanner or a stereographic system.

Data registration: Alignment of several scans into a single coordinate system.

Data integration: Interpolation of the measured samples, or points derived from the measured samples, with a surface representation, usually a triangle mesh.

Model conversion: Mesh decimation/optimization, fitting with higher-order representations etc.

This paper focuses on the data integration phase. We present a new method for finding a triangle mesh that interpolates an unorganized set of points. Figure 1 shows data points from a small collection of scans and a triangle mesh obtained with our method (the technique used to generate the textures is described in [18]).

The method makes two mild assumptions about the samples that are valid for a wide range of acquisition techniques: that the samples are distributed over the entire surface with a spatial frequency greater than or equal to some application-specified minimum value, and that an estimate of the surface normal is available for each measured sample.

Main contributions:

- The method is conceptually simple. Starting with a seed triangle, it pivots a ball around each edge on the current mesh boundary until a new point is hit by the ball. The edge and point define a new triangle, which is added to the mesh, and the algorithm considers a new boundary edge for pivoting.
- The output mesh is a manifold subset of an alpha-shape [13] of the point set. Some of the nice properties

of alpha-shapes can also be proved for our reconstruction.

- The Ball Pivoting Algorithm (BPA for short) is efficient in terms of execution time and storage requirements. It exhibited linear time performance on datasets consisting of millions of input samples. It has been implemented in a form that does not require all of the input data to be loaded into memory simultaneously. The resulting triangle mesh is incrementally saved to external storage during its computation, and does not use any additional memory.

- The BPA proved robust enough to handle the noise present in real scanned 3D data. It was tested on several large scanned datasets, and in particular was used to create models of Michelangelo's Florentine Pietà [1] from hundreds of scans acquired with a structured light sensor (Visual Interface's Virtuoso ShapeCamera). The BPA allowed us to process this data in less than one hour on an off-the-shelf PC.

The rest of the paper is structured as follows: In section 2 we define the problem and discuss related work. In section 3 we discuss the concepts underlying the Ball-Pivoting Algorithm, and in section 4 describe the algorithm in detail. We present results in section 5, and discuss open problems and future work in section 6.

2 Background

Recent years have seen a proliferation of scanning equipment and algorithms for synthesizing models from scanned data. We refer the reader to two recent reviews of research in the field [6, 16]. In this section we focus on the role interpolating meshing schemes can play in scanning objects, and why they have not been used in practical scanning systems.

2.1 Interpolating Meshes in Scanning Systems

We define the scanning problem: Given an object, find a continuous representation of the object surface that captures features of a length scale $2d$ or larger. The value of d is dictated by the application. Capturing features of scale $2d$ requires sampling the surface with a spatial resolution of d or less. The surface may consist of large areas that can be well approximated by much sparser meshes; however in the absence of a priori information we need to begin with a sampling resolution of d or less to guarantee that no feature is missed.

We consider acquisition systems that produce sets of range images, i.e. arrays of depths, each of which covers a subset of the full surface. Because they are height fields with regular sampling, individual range images are easily meshed. The individual meshes can be used to compute an estimated surface normal for each sample point.

A perfect acquisition system would return samples lying exactly on the object surface. Any physical measurement system introduces some error. However, if a system returns samples with an error that is orders of magnitude smaller than the minimum feature size, the samples can be regarded as lying on the object surface. A surface can be reconstructed by finding an interpolating mesh without additional operations on the measured data.

Most scanning systems still need to account for acquisition error. There are two sources of error – error in registration, and error along the sensor line of sight. Estimates of true surface points are derived using samples from redundant scans. Interpolating meshes are formed for these derived, or consensus, surface points. Methods for computing consensus points are usually constrained to structures that facilitate the construction of the mesh. Two classes of methods have been used successfully for large datasets. Both methods assume negligible registration error, and compute estimates to correct for errors along sensor lines of sight. One class of method is volumetric, such as that introduced by Curless and Levoy [10]. In volumetric methods, individual aligned meshes are used to compute a signed distance function on a volume grid encompassing the object. Consensus points are computed as the points on the grid where the distance function is zero. The structure of the volume then facilitates the construction of a mesh using the marching cubes algorithm [14].

Another class of methods stitches together disjoint height field meshes, such as the technique of Soucy and Laurendeau [19]. Disjoint meshes are defined by finding areas of overlap of the acquired meshes. Consensus points are computed on these disjoint meshes using data projected along sensor lines of sight. The disjoint meshes of consensus points are then stitched together to form a single mesh. Turk and Levoy developed a similar method [21], which first stitches (or zippers) the disjoint meshes, and then computes consensus points.

In the volumetric approach, a general mesh interpolation technique could be used in place of the marching cubes. In the mesh-joining approaches, a general interpolation technique could be used in place of tracking all the edges to be joined. Most importantly, with a general technique, any method for computing consensus points could be used, including those that do not impose additional structure on the data and which do not treat registration and sensor line-of-sight error separately. For example, it has been demonstrated that reducing error in individual meshes before alignment can reduce registration error [11]. A method that moves samples within the known error bounds to conform the meshes to one another as they are aligned could potentially reduce registration errors.

Finally, it may be desirable to find an interpolating mesh from measured data even if it contains uncompensated error. The preliminary mesh could be smoothed, cleaned, and decimated for use in planning functions. A mesh interpolating

measured points could also be used as a starting point for computing consensus points.

2.2 State of the Art for Interpolating Meshes

Existing interpolating techniques fall into two categories – sculpting-based [4, 2, 6] and region-growing [7, 15]. In sculpting-based methods, a volume tetrahedralization is computed from the data points, typically the 3D Delaunay triangulation. Tetrahedra are then removed from the convex hull to extract the original shape. Region-growing methods start with a seed triangle, consider a new point and join it to the existing region boundary, and continue until all points have been considered.

The strength of sculpting-based approaches is that they often provide theoretical guarantees for the quality of the resulting surface, e.g. that the topology is correct, and that the surface converges to the true surface as the sampling density increases (see e.g. [5, 3]). However, computing the required 3D Delaunay triangulation can be prohibitevly expensive in terms of time and memory required, and can lead to numerical instability when dealing with datasets of millions of points. The goal of the BPA is to retain the strengths of previous interpolating techniques in a method that exhibits linear time complexity and robustness on real scanned data.

3 Surface Reconstruction and Ball-Pivoting

The main concept underlying the Ball-Pivoting Algorithm is quite simple. Let the manifold M be the surface of a three-dimensional object and S be a point-sampling of M. Let us assume that sample points are spaced by a distance $d < 2R$, where R is the smallest radius of curvature of the surface (see figure 3 for a 2D example). We start by placing a ρ-ball (a ball of radius ρ), $\rho > d/2$, in contact with three sample points. Keeping the ball in contact with two of these initial points, we "pivot" the ball until it touches another point, as illustrated in figure 2 (more details are given is section 4.3). We pivot around each edge of the current mesh boundary. Triplets of points that the ball contacts form new triangles. The set of triangles formed while the ball "walks" on the surface constitutes the interpolating mesh.

The BPA is closely related to alpha-shapes [12, 13]. In fact every triangle T computed by the ρ-ball walk obviously has an empty smallest open ball b_T whose radius is less than ρ (see [13], page 50). Thus, the BPA computes a subset of the 2-faces of the ρ-shape of S. These faces are also a subset of the 2-skeleton of the three-dimensional Delaunay triangulation of the point set. Alpha shapes are an effective tool for computing the "shape" of a point set. The surface reconstructed by the BPA retains some of the qualities of alpha-shapes: It has provable reconstruction guarantees under certain sampling assumptions, and an intuitively simple

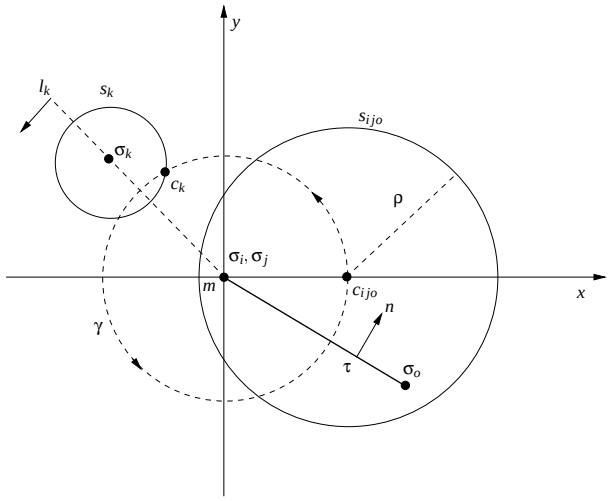
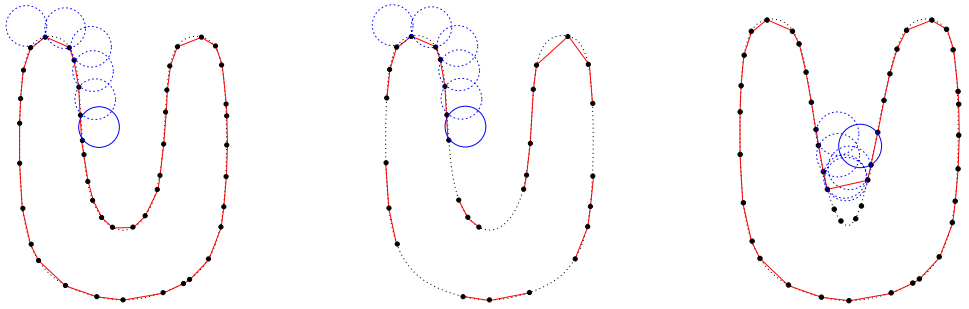


Figure 2: Ball pivoting operation. See section 4.3 for further details. The pivoting ball is in contact with the three vertices of triangle $\tau = (\sigma_i, \sigma_j, \sigma_o)$, whose normal is n. The pivoting edge $e_{(i,j)}$ lies on the z axis (perpendicular to the page and pointing towards the viewer), with its midpoint m at the origin. The circle s_{ijo} is the intersection of the pivoting ball with $z = 0$. The coordinate frame is such that the center c_{ijo} of the ball lies on the positive x axis. During pivoting, the ρ-ball stays in contact with the two edge endpoints σ_i, σ_j, and its center describes a circular trajectory γ with center in m and radius $||c_{ijo} - m||$. In its pivoting motion, the ball hits a new data point σ_k. Let s_k be the intersection of a ρ-sphere centered at σ_k with $z = 0$. The center c_k of the pivoting ball when it touches σ_k is the intersection of γ with s_k lying on the negative halfplane of oriented line l_k.



(a) (b) (c)

Figure 3: The Ball Pivoting Algorithm in 2D. (a) A circle of radius ρ pivots from sample point to sample point, connecting them with edges. (b) When the sampling density is too low, some of the edges will not be created, leaving holes. (c) When the curvature of the manifold is larger than $1/\rho$, some of the sample points will not be reached by the pivoting ball, and features will be missed.

geometric meaning.

However, the 2-skeleton of an alpha-shape computed from a noisy sampling of a smooth manifold can contain multiple non-manifold connections. It is non-trivial to filter out unwanted components. Also, in their original formulation, alpha-shapes are computed by extracting a subset of the 3D Delaunay triangulation of the point set, a data structure that is not easily computed for datasets of millions of points. With the assumptions on the input stated in the introduction, the BPA efficiently and robustly computes a manifold subset of an alpha-shape that is well suited for this application.

In [5], sufficient conditions on the sampling density of a curve in the plane were derived which guarantee that the alpha-shape reconstruction is homeomorphic to the original manifold and that it lies within a bounded distance. The thorem can be easily extended to surfaces: Suppose that for the smooth manifold M the sampling S satisfies the following properties:

1. The intersection of any ball of radius ρ with the manifold is a topological disk.
2. Any ball of radius ρ centered on the manifold contains at least one sample point in its interior.

The first condition guarantees that the radius of curvature of the manifold is larger than ρ, and that the ρ-ball can also pass through cavities and other concave features without multiple contacts with the surface. The second condition tells us that the sampling is dense enough that the ball can walk on the sample points without leaving holes (see figure 3 for 2D examples). The BPA then produces a homeomorphic approximation T of the smooth manifold M. We can also define a homeomorphism $h : T \mapsto M$ such that the distance $||p - h(p)|| < \rho$.

In practice, we must often deal with less-than-ideal samplings. What is the behavior of the algorithm in these cases? Let us consider the case of real scanned data. Typical problems are missing points, non-uniform density, imperfectly-aligned, overlapping range scans, and scanner line-of-sight error.[1]

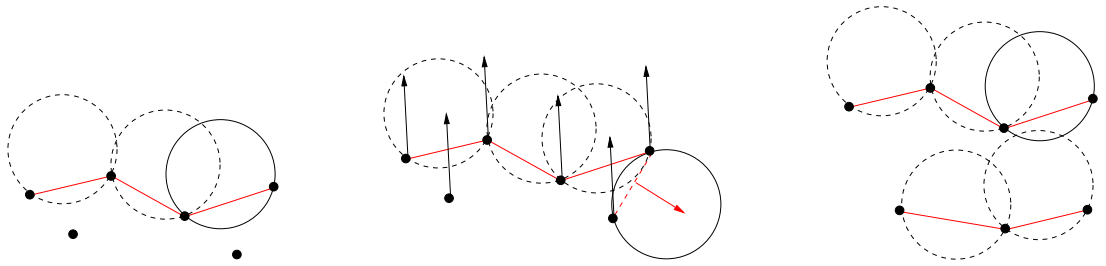
The BPA is designed to process the output of an accurate registration/conformance algorithm (see section 2), and does not attempt to average out noise or residual registration errors. Nonetheless, the BPA is robust in the presence of imperfect data.

We augment the data points with approximate surface normals computed from the range maps to disambiguate cases that occur when dealing with missing or noisy data. For example, if parts of the surface have not been scanned, there will be holes larger than ρ in the sampling. It is then impossible to distinguish an interior and an exterior region with respect to the sampling. We use surface normals (for which we assume outward orientation) to decide surface orientation. For example, when choosing a seed triangle we check that the surface normals at the three vertices are consistently oriented.

Areas of density higher than ρ present no problem. The pivoting ball will still "walk" on the points, forming small triangles. If the data is noise-free and ρ is smaller than the local curvature, all points will be interpolated. More likely, points are affected by noise, and some of those lying below the surface will not be touched by the ball and will not be part of the reconstructed mesh (see figure 4(a)).

Missing points create holes that cannot be filled by the pivoting ball. Any postprocessing hole-filling algorithm could be employed; in particular, BPA could be applied a second time, with a larger ball, on the subset of points on remaining boundary edges of the output mesh. However, we do need to handle possible ambiguities that missing data can introduce. When pivoting around a boundary edge, the ball can touch an unused point lying close to the surface. Again we use sur-

[1] Some types of scanners also produce "outliers", points that lie far from the actual surface. These outliers occur more frequently at the boundaries of range images, or in the presence of sharp discontinuities. Outlier removal is best done with device-dependent preprocessing. The scanner used to acquire the data presented in the results section is not affected by this problem.



(a) (b) (c)

Figure 4: Ball pivoting in the presence of noisy data. (a) Surface samples lying "below" surface level are not touched by the pivoting ball and remain isolated (and are discarded by the algorithm). (b) Due to missing data, the ball pivots around an edge until it touches a sample that belongs to a different part of the surface. By checking that triangle and data point normals are consistently oriented, we avoid generating a triangle in this case. (c) Noisy samples form two layers, distant enough to allow the ρ ball to "walk" on both layers. A spurious small component is created. Our seed selection strategy avoids the creation of a large number of these small components. Remaining ones can be removed with a simple postprocessing step. In all cases, the BPA outputs an orientable, triangulated manifold.

face normals to decide whether the point touched is valid or not (see figure 4(b)).

The presence of misaligned overlapping range scans can lead to poor results if the registration error is similar to the pivoting ball size. Undesired small connected components lying close to the main surface will be formed, and the main surface affected by high frequency noise (see figure 4(c). Our seed selection strategy however avoids creating a large number of such small components). A simple postprocessing that removes small components and surface smoothing [20] can significantly improve the result in these cases, at least esthetically.

Regardless of the defects in the data, the BPA is guaranteed to build an orientable manifold. Notice that the BPA will always try to build the largest possible connected manifold from a given seed triangle.

4 The Ball-Pivoting Algorithm

The BPA follows the advancing-front paradigm to incrementally build an interpolating triangulation. BPA takes as input a list of surface-sample data points σ_i, each associated with a normal n_i (and other optional attributes, such as texture coordinates), and a ball radius ρ. The basic algorithm works by finding a *seed triangle* (*i.e.*, three data points $(\sigma_i, \sigma_j, \sigma_k)$ such that a ball of radius ρ touching them contains no other data point), and adding one triangle at a time by performing the ball pivoting operation explained in section 3.

The *front* F is represented as a collection of linked lists of edges, and is initially composed of a single loop containing the three edges defined by the first seed triangle. Each edge $e_{(i,j)}$ of the front, is represented by its two endpoints (σ_i, σ_j), the opposite vertex σ_o, the center c_{ijo} of the ball that touches all three points, and links to the previous and next edge along in the same loop of the front. An edge can be *active*,

Algorithm *BPA*(S, ρ)

```
1.  while (true)

2.      while (e_(i,j) = get_active_edge(F))
3.          if (σ_k = ball_pivot(e_(i,j)) &&
                    (not_used(σ_k) || on_front(σ_k)))
4.              output_triangle(σ_i, σ_k, σ_j)
5.              join(e_(i,j), σ_k)
6.              if (∃e_(k,i)) glue(e_(i,k), e_(k,i))
7.              if (∃e_(j,k)) glue(e_(k,j), e_(j,k))
8.          else
9.              mark_as_boundary(e_(i,j))

10.     if ((σ_i, σ_j, σ_k) = find_seed_triangle())
11.         output_triangle(σ_i, σ_j, σ_k)
12.         insert_edge(e_(i,j), F)
13.         insert_edge(e_(j,k), F)
14.         insert_edge(e_(k,i), F)
15.     else
16.         return
```

Figure 5: Skeleton of the BPA algorithm. Several necessary error tests have been left out for readability, such as edge orientation checks. The edges in the front F are generally accessed by keeping a queue of active edges. The *join* operation adds two active edges to the front. The *glue* operation deletes two edges from the front, and changes the topology of the front by breaking a single loop into two, or combining two loops into one. See text for details. The *find_seed_triangle* function returns a ρ^2-exposed triangle, which is used to initialize the front.

boundary or*frozen*. An *active* edge is one that will be used for pivoting. If it is not possible to pivot from an edge, it is marked as *boundary*. The *frozen* state is explained below, in the context of our out-of-core extensions. Keeping all this information with each edge makes it simpler to pivot the ball around it. The reason the front is a collection of linked lists, instead of a single one, is that as the ball pivots along an edge, depending on whether it touches a newly encountered data point or a previously used one, the front changes topology. BPA handles all cases with two simple topological operators *join* and *glue*, which ensure that at any time the front is a collection of linked lists.

The basic BPA algorithm is shown in figure 5. Below we detail the functions and data structures used. In particular, we later describe a simple modification necessary to the basic algorithm to support efficient out-of-core execution. This allows BPA to triangulate large datasets with minimal memory usage.

4.1 Spatial queries

Both *ball_pivot* and *find_seed_triangle* (lines 3 and 10 in figure 5) require efficient lookup of the subset of points contained in a small spatial neighborhood. We implemented this spatial query using a regular grid of cubic cells, or voxels. Each voxel has sides of size $\delta = 2\rho$. Data points are stored in a list, and the list is organized using bucket-sort so that points lying in the same voxel form a contiguous sublist. Each voxel stores a pointer to the beginning of its sublist of points (to the next sublist if the voxel is empty). An extra voxel at the end of the grid stores a NULL pointer. To visit all points in a voxel it is sufficient to traverse the list from the node pointed to by the voxel to the one pointed to by the next voxel.

Given a point p we can easily find the voxel V it lies in by dividing its coordinates by δ. We usually need to look up all points within 2ρ distance from p, which are a subset of all points contained in the 27 voxels adjacent to V (including V itself).

The grid allows constant-time access to the points. Its size would be prohibitive if we processed a large dataset in one step; but an out-of-core implementation, described in section 4.5, can process the data in manageable chunks. Memory usage can be further reduced, at the expense of a slower access, using more compact representations, such as a sparse matrix data structure or a hash table that uses the (i, j, k) voxel indices as keys.

4.2 Seed selection

Given data satisfying the conditions of the reconstruction theorem of section 3, one seed per connected component is enough to reconstruct the entire manifold (function *find_seed_triangle* at line 10 in figure 5). A simple way to find a valid seed is to:

- Pick any point σ not yet used by the reconstructed triangulation.
- Consider all pairs of points σ_a, σ_b in its neighborhood in order of distance from σ.
- Build potential seed triangles $\sigma, \sigma_a, \sigma_b$.
- Check that the triangle normal is consistent with the vertex normals, *i.e.* pointing outward.
- Test that a ρ-ball with center in the outward halfspace touches all three vertices and contains no other data point.
- Stop when a valid seed triangle has been found.

In the presence of noisy, incomplete data, it is important to select an efficient seed-searching strategy. Given a valid seed, the algorithm builds the largest possible connected component containing the seed. Noisy points lying at a distance slightly larger than 2ρ from the reconstructed triangulation could form other potential seed triangles, leading to the construction of small sets of triangles lying close to the main surface (see figure 4(c)). These small components are an artifact of the noise present in the data, and are usually undesired. While they are easy to eliminate by post-filtering the data, a significant amount of computational resources are wasted in constructing them.

We can however observe the following: If we limit ourselves to considering only one data point per voxel as a candidate vertex for a seed triangle, we cannot miss components spanning a volume larger than a few voxels. Also, for a given voxel, consider the average normal n of points within it. This normal approximates the surface normal in that region. Since we want our ball to walk "on" the surface, it is convenient to first consider points whose projection onto n is large and positive.

We therefore simply keep a list of non-empty voxels. We search these voxels for valid seed triangles, and when one is found, we start building a triangulation using pivoting operations. When no more pivoting is possible, we continue the search for a seed triangle from where we had stopped, skipping all voxels containing a point that is now part of the triangulation. When no more seeds can be found, the algorithm stops.

4.3 Ball Pivoting

A pivoting operation (line 3 in figure 5) starts with a triangle $\tau = (\sigma_i, \sigma_j, \sigma_o)$ and a ball of radius ρ that touches its three vertices. Without loss of generality, assume edge $e_{(i,j)}$ is the pivoting edge. The ball in its initial position (let c_{ijo} be its center) does not contain any data point, either because τ is a seed triangle, or because τ was computed by a previous pivoting operation. The pivoting is in principle a continuous motion of the ball, during which the ball stays in contact with the two endpoints of $e_{(i,j)}$, as illustrated in figure 2. Because

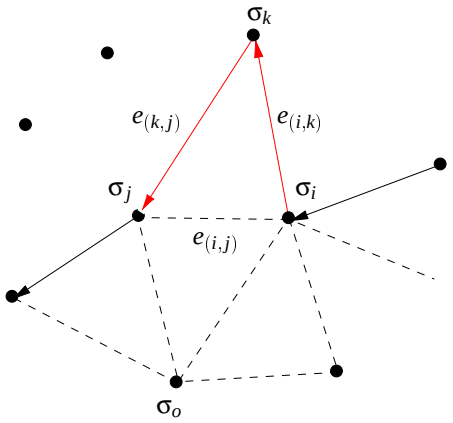


Figure 6: A *join* operation simply adds a new triangle, removing edge $e_{(i,j)}$ from the front and adding the two new edges $e_{(i,k)}$ and $e_{(k,j)}$.

of this contact, the motion is constrained as follows: The center of the ball describes a circle γ which lies on the plane perpendicular to $e_{(i,j)}$ and through its midpoint $m = \frac{1}{2}(\sigma_j + \sigma_i)$. The center of this circular trajectory is m and its radius is $||c_{ijo} - m||$. During this motion, the ball may hit another point σ_k. If no point is hit, then the edge is a boundary edge. Otherwise, the triangle $(\sigma_i, \sigma_k, \sigma_j)$ is a new valid triangle, and the ball in its final position does not contain any other point, thus being a valid starting ball for the next pivoting operation.

In practice we find σ_k as follows. We consider all points in a 2ρ-neighborhood of m. For each such point σ_x, we compute the center c_x of the ball touching σ_i, σ_j and σ_x, if such a ball exists. Each c_x lies on the circular trajectory γ around m, and can be computed by intersecting a ρ-sphere centered at σ_x with the circle γ. Of these points c_x we select the one that is first along the trajectory γ. We report the first point hit and the corresponding ball center. Trivial rejection tests can be added to speed up finding the first hit-point.

4.4 The *join* and *glue* operations

These two operations generate triangles while adding and removing edges from the front loops (lines 5-7 in figure 5).

The simpler operation is the *join*, which is used when the ball pivots around edge $e_{(i,j)}$, touching a *not_used* vertex σ_k (*i.e.*, σ_k is a vertex that is not yet part of the mesh). In this case, we output the triangle $(\sigma_i, \sigma_k, \sigma_j)$, and locally modify the front by removing $e_{(i,j)}$ and adding the two edges $e_{(i,k)}$ and $e_{(k,j)}$ (see figure 6).

When σ_k is already part of the mesh, one of two cases can arise:

1. σ_k is an internal mesh vertex, (*i.e.*, no front edge uses σ_k). The corresponding triangle cannot be generated, since it would create a non-manifold vertex. In this case, $e_{(i,j)}$ is simply marked as a boundary edge;

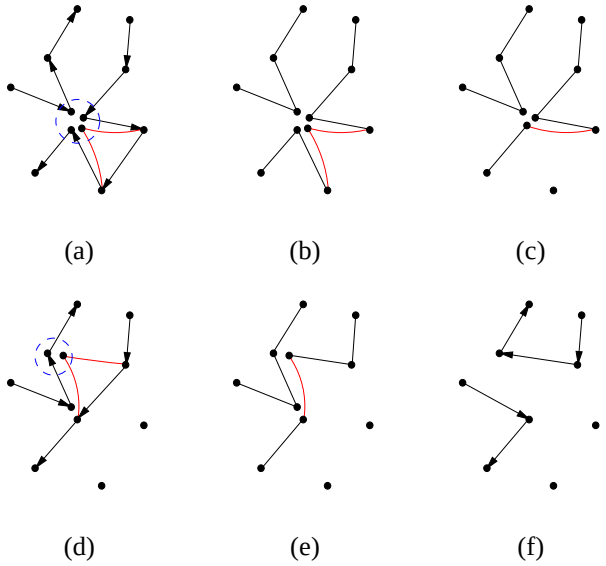


Figure 8: Example of a sequence of *join* and *glue* operations. (a) A new triangle is to be added to the existing front. The four front vertices inside the dashed circle all represent a single data point. (b) A *join* removes an edge and creates two new front edges, coincident with existing ones. (c), (d) Two *glue* operations remove coincident edge pairs. (d) also shows the next triangle added. (e) Only one of the edges created by this *join* is coincident with an existing edge. (f) One *glue* removes the duplicate pair.

2. σ_k belongs to the front. After checking edge orientation to avoid creating a non-orientable manifold, we apply a *join* operation, and output the new mesh triangle $(\sigma_i, \sigma_k, \sigma_j)$. The *join* could potentially create (one or two) pairs of coincident edges (with opposite orientation), which are removed by the *glue* operation.

The *glue* operation removes from the front pairs of coincident edges, with opposite orientation (coincident edges with the same orientation are never created by the algortihm). For example, when edge $e_{(i,k)}$ is added to the front by a *join* operation (the same applies to $e_{(k,j)}$), if edge $e_{(k,i)}$ is on the front, *glue* will remove the pair of edges $e_{(i,k)}, e_{(k,i)}$ and adjust the front accordingly. Four cases are possible, as illustrated in figure 7.

An sequence of *join* and *glue* operations is illustrated in figure 8.

4.5 Out-of-core extensions

Being able to use a personal computer to triangulate high-resolution scans allows inexpensive on-site processing of data. Due to their locality of reference, advancing-front algorithms are suited to very simple out-of-core extensions.

We employed a simple data-slicing scheme to extend the algorithm shown in figure 5. The basic idea is to cache the portion of the dataset currently being used for pivoting, to

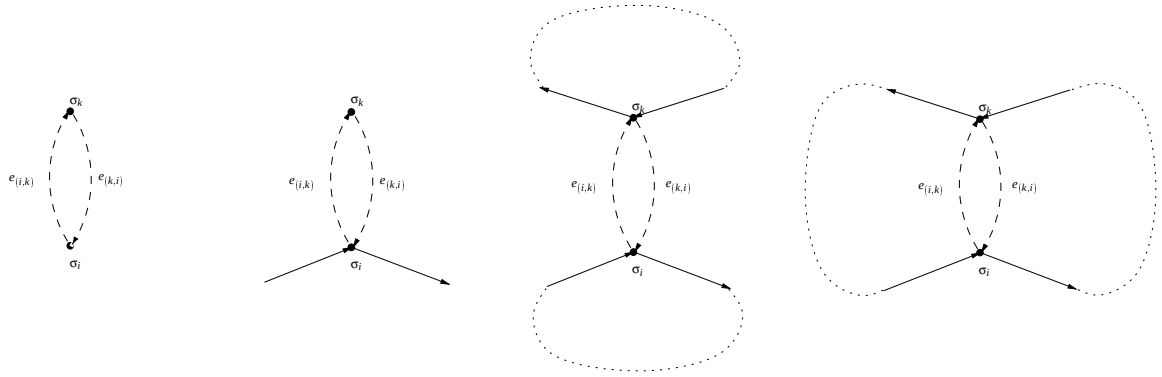


Figure 7: A *glue* operation is applied when *join* creates an edge identical to an existing edge, but with opposite orientation. The two coincident edges are removed, and the front adjusted accordingly. There are four possible cases: (a) The two edges form a loop. The loop is deleted from the front. (b) Two edges belong to the same loop and are adjacent. The edges are removed and the loop shortened. (c) The edges are not adjacent, and they belong to the same loop. The loop is split into two. (d) The edges are not adjacent and belong to two different loops. The loops are merged into a single loop.

dump data no longer being used, and to load data as it is needed. In our case, we use two axis-aligned planes π_0 and π_1 to define the active region of work for pivoting. We initially place π_0 in such a way that no data points lie "below" it, and π_1 above π_0 at some user-specified distance. As each edge is created, we test if its endpoints are "above" π_1; in this case, we mark the edge *frozen*. When all edges remaining in the queue are *frozen*, we simply shift π_0 and π_1 "upwards", and update all *frozen* into *active* edges, and so on. A subset of data points is loaded and discarded from memory when the corresponding bounding box enters and exits the active slice. Scans can easily be preprocessed to break them up into smaller meshes, so that the memory load remains low.

The only change required in the algorithm to implement this refinement is an outer loop to move the active slice, and the addition of the instructions to unfreeze edges between lines 1–2 of figure 5.

4.6 Remarks

The BPA algorithm was implemented in C++ using the Standard Template Library. The whole code is less than 4000 lines, including the out-of-core extensions.

The algorithm is linear in the number of data points and uses linear storage, under the assumption that the data density is bounded. This assumption is appropriate for scanned data, which is collected by equipment with a known sample spacing. Even if several scans overlap, the total number of points in any region will be bounded by a known constant.

Most steps are simple $O(1)$ state checks or updates to queues, linked lists, and the like. With bounded density, a point need only be related to a constant number of neighbors. So, for example, a point can only be contained in a constant number of loops in the advancing front. The two operations *ball_pivot* and *find_seed_triangle* are more complex.

Each *ball_pivot* operates on a different mesh edge, so the number of pivots is $O(n)$. A single pivot requires identifying all points in a 2ρ neighborhood. A list of these points can be collected from 27 voxels surrounding the candidate point in our grid. With bounded density, this list has constant size B. We perform a few algebraic computations on each point in the list and select the minimum result, all $O(1)$ operations on a list of size $O(1)$.

Each *find_seed_triangle* picks unused points one at a time and tests whether any incident triangle is a valid seed. No point is considered more than once, so this test is performed only $O(n)$ times. To test a candidate point, we gather the same point-list discussed above, and consider pairs of points until we either find a seed triangle or reject the candidate. Testing one of these triangles may require classifying every nearby point against a sphere touching the three vertices, in the worst case, $O(B^3) = O(1)$ steps. In practice, we limit the number of candidate points and triangles tested by the heuristics discussed in section 4.2.

An in-core implementation of the BPA uses $O(n+L)$ memory, where L is the number of cells in the voxel grid. The $O(n)$ term includes the data, the advancing front (which can only include each mesh edge once), and the candidate edge queue. Our out-of-core implementation uses $O(m+L')$ memory, where m is the number of data points in the largest slice and L' is the size of the smaller grid covering a single slice. Since the user can control the size of slices, memory requirements can be tailored to the available hardware. The voxel grid can be more compactly represented as a sparse matrix, with a small (constant) increase in access time.

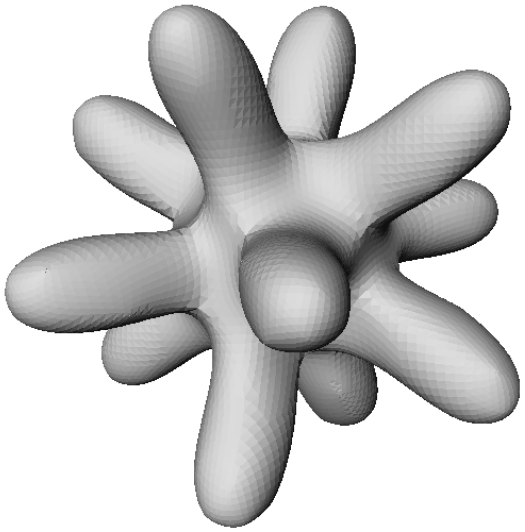


Figure 9: Results. “Clean” data computed from an analytical surface.

5 Experimental Results

Our experiments for this paper were all conducted on PCs running Microsoft Windows NT 4.0 or RedHat Linux 5.1. We primarily report timings on one 450MHz Pentium II Xeon processor of an IBM IntelliStation Z Pro running Windows NT.

In our experiments we used several datasets: “clean” dataset (*i.e.*, points from analytical surface, see figure 9); the datasets from the Stanford scanning database (see figure 10(a)-(c)); and a very large dataset we acquired ourselves (and the main motivation of this work), a model of Michelangelo’s Florentine Pietà [1] (see figure 10(d)).

To allow flexible input of multiple scans and out-of-core execution, our program reads its input in four parts: a list of meshes to be triangulated; and for each mesh, a transformation matrix, a post-transform bounding box (used to quickly estimate the mesh position for assignment to a slice), and the actual mesh, which is loaded only when needed.

5.1 Experiments

Table 1 summarizes our results. The “clean” dataset is a collection of points from an analytical surface.

The Stanford Bunny, Dragon and Buddha datasets are multiple laser range scans of small objects. The scanner used to acquire the data was a CyberWare 3030MS.

These data required some minor preprocessing. We used the Stanford `vrip` program to generate a triangulated mesh from each range data file. We also removed the plane carvers, large planes of triangles used for hole-filling by algorithms described in [10]. This change was made only for esthetic reasons; BPA has no problem handling the full input.

In order to confirm the effectiveness of our out-of-core capabilities, we modified the Stanford Dragon, which requires in excess of 350MB to process, by subdividing each range mesh into several pieces, multiplying the original 71 meshes to over 7500. A similar preprocessing was also applied to the Buddha dataset. We note that such decompositions can be performed efficiently for arbitrarily large range scans (which do not necessarily need to fit in memory) by the techniques described in [8].

The Pietà data has undergone extensive preprocessing during and after scanning and registration that is out of the scope of this paper. The data is large enough that it cannot be processed in-core, and is only processed in slices.

6 Conclusions

In this paper, we introduced the Ball-Pivoting Algorithm, an advancing-front algorithm to incrementally build an interpolating triangulation of a given point cloud. BPA has several desirable properties:

- **Intuitive:** BPA triangulates a set of points by “rolling” a ρ-ball on the point cloud. The user chooses only a single parameter.
- **Flexible, efficient, and robust:** Our test datasets ranged from small synthetic data to large real-world scans. We have shown that our implementation of BPA works on datasets of millions of points representing actual scans of complex 3D objects. It does so efficiently, and on off-the-shelf PCs.
- **Strong theoretical foundation:** BPA is related to alpha-shapes [12, 13], and given sufficiently-dense sampling, it is guaranteed to reconstruct a surface homeomorphic to and within a bounded distance from the original manifold.

There are some avenues for further work. It would be interesting to evaluate whether BPA can be used to triangulate surfaces sampled with particle systems. This possibility was left as an open problem in [22], and further developed in [9] in the context of isosurface generation.

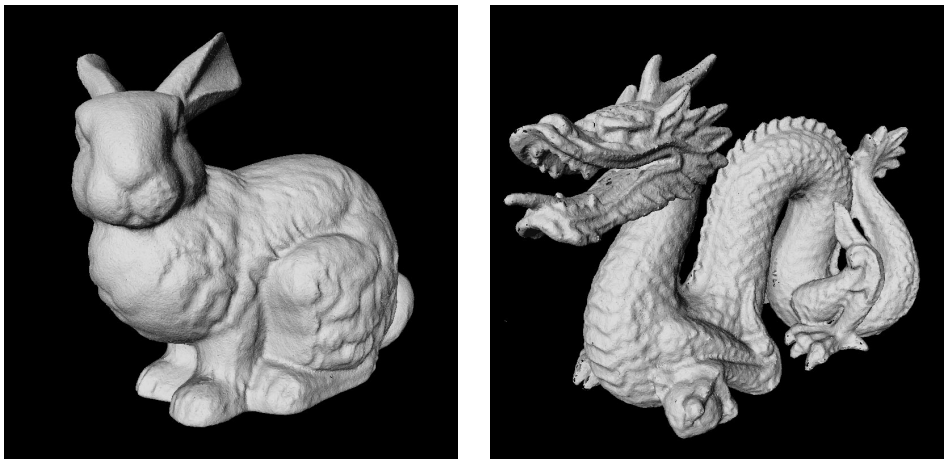
By using weighted points [12], we might be able to generate triangulations of adaptive samplings. The sampling density could be changed depending on local surface properties, and point weights accordingly assigned or computed. An extension of our algorithm along the lines of the weighted generalization of alpha-shapes [12] should be able to generate a more compact, adaptive, interpolating triangulation.

We have done some initial experiments in using a smoothing algorithm adapted from [20] to pre-process the data and to compute consensus points from multiple scans to be used as input to the BPA. Our preliminary results are encouraging.

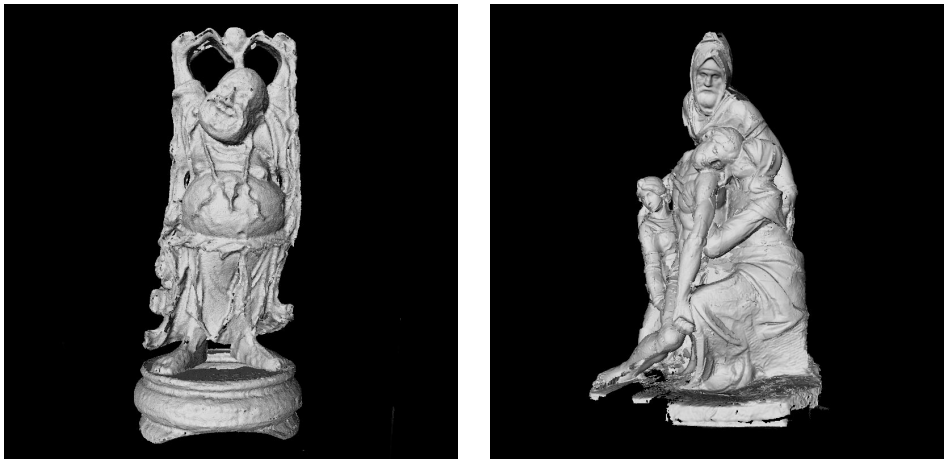
Acknowledgments. Thanks to the Stanford University Computer Graphics Laboratory, for making some of the range data used in this paper publicly available.

Dataset	# Pts	# Scans	ρ	# Slices	# Triangles	Mem. Usage	I/O Time	CPU Time
Clean	11K	1	4	-	22K	4	1.2secs	1.8secs
Bunny	361K	10	0.7	-	452K	92	26secs	66secs
Dragon	2.1M	71	0.7	-	641K	320	1.5	13
Buddha	3.3M	58	0.7	-	706K	405	2.5	14
Out of core								
Dragon	2.3M	7530	0.7	23	641K	70	2	32
Buddha	3.5M	3743	0.7	24	706K	110	3	36
Pietà	8.6M	555	3	24	5.4M	260	20	43

Table 1: Summary of results. *# of Pts* and *# of Scans* are the original number of data points and range images respectively. ρ is the radius of the pivoting ball, in *mm*. *# Slices* is the number of slices into which the data is partitioned for out-of-core processing. *# of Triangles* is the number of triangles created by BPA. *Mem. Usage* is the maximum amount of memory used at any time during mesh generation, in MB. *I/O Time* is the time spent reading and parsing the files; it also includes the time to write the output triangles. *CPU Time* is the time spent computing the triangulation. All times are in minutes, except where otherwise stated. All tests were performed on a 450MHz Pentium II Xeon.



(a) (b)



(c) (d)

Figure 10: Results. (a) Stanford bunny. (b) Stanford dragon. (c) Stanford Buddha. (d) Preliminary reconstruction of Michelangelo's Florentine Pietà.

References

[1] J. Abouaf. The Florentine Pietà: Can visualization solve the 450-year-old mistery? *IEEE Computer Graphics & Applications*, 19(1):6–10, February 1999.

[2] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proc. SIGGRAPH '98*, Computer Graphics Proceedings, Annual Conference Series, pages 415–412, July 1998.

[3] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *Proc. 14th Annual ACM Sympos. Comput. Geom.*, pages 39–48, 1998.

[4] C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 95, pages 109–118, 1995.

[5] F. Bernardini and C. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. In *Proc. of the Ninth Canadian Conference on Computational Geometry*, pages 193–198, August 1997. Updated online version available at www.qucis.queensu.ca/cccg97.

[6] F. Bernardini, C. Bajaj, J. Chen, and D. Schikore. Automatic reconstruction of 3D CAD models from digital scans. *International Journal of Computational Geometry and Applications*, 1999. To appear.

[7] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, 1984.

[8] Yi-Jen Chiang, Cláudio T. Silva, and William Schroeder. Interactive out-of-core isosurface extraction. In *Proc. IEEE Visualization ́98*, pages 167–174, November 1998.

[9] Patricia Crossno and Edward Angel. Isosurface extraction using particle systems. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization ́97*, pages 495–498. IEEE, November 1997.

[10] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 96, pages 303–312, 1996.

[11] C. Dorai, G. Wang, A. K. Jain, and C. Mercer. Registration and integration of multiple object views for 3D model construction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):83–89, January 1998.

[12] H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1992.

[13] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, January 1994.

[14] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Comput. Graph.*, 21(4):163–170, 1987.

[15] R. Mencl. A graph-based approach to surface reconstruction. *Computer Graphics Forum*, 14(3):445–456, 1995. Proc. of EUROGRAPHICS '95.

[16] R. Mencl and H. Müller. Interpolation and approximation of surfaces from three-dimensional scattered data points. In *Proceeding of Eurographics '98*, State of the Art Reports. Eurographics, 1998.

[17] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle. Robust meshes from multiple range maps. In *Intl. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, pages 205–211. IEEE Computer Society Press, May 1997.

[18] H. Rushmeier and F. Bernardini. Computing consistent normals and colors from photometric data. Submitted for publication, 1999.

[19] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):344–358, April 1995.

[20] Gabriel Taubin. A signal processing approach to fair surface design. In *Proc. of SIGGRAPH '95*, Computer Graphics Proceedings, Annual Conference Series, pages 351–358. ACM SIGGRAPH, August 1995.

[21] G. Turk and M. Levoy. Zippered polygonal meshes from range images. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 94, pages 311–318, 1994.

[22] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 269–278. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

Computing Consistent Normals and Colors from Photometric Data

H. Rushmeier and F. Bernardini*

IBM Thomas J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532

Abstract

We present a method for computing normals and colors from multiple sets of photometric data, that are consistent with each other and an underlying lower resolution mesh. Normals are computed by locally adjusting the light source intensities using data from the underlying mesh. Colors are derived from the photometric calculations, and are adjusted by a global color registration analogous to global geometric registration.

1 Introduction

We consider the problem of scanning objects that are large relative to the smallest geometric level of detail to be represented. Three dimensional models of these objects are obtained by combining the results of hundreds of individual scans. We have developed a hybrid multiview/photometric method for scanning such objects. A multiview light striping system is used to obtain a base geometric model. A photometric system is used to obtain normals and colors at a higher spatial resolution. In this paper we present a solution for computing these colors and normals so that the results are consistent over the object, and with the underlying base geometry.

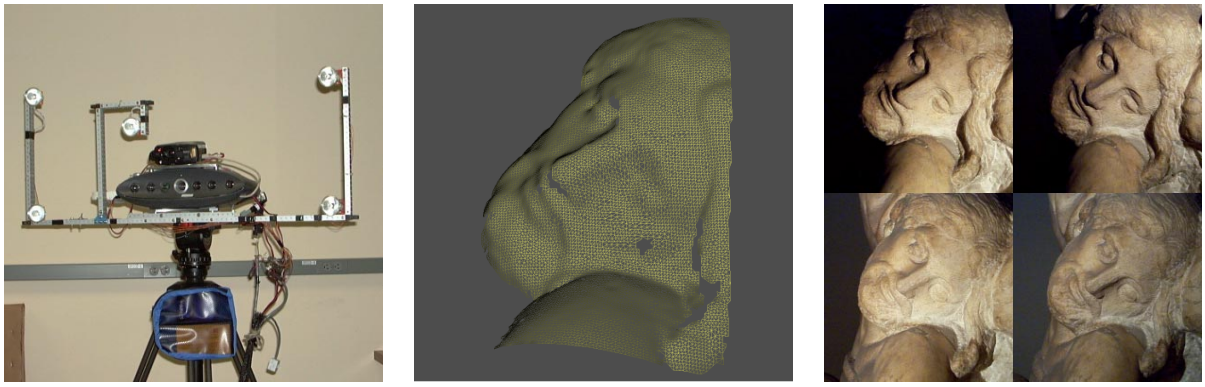
The motivation for this work is a project to acquire a three dimensional model of Michelangelo's Florentine Pietà [1]. The model is being used by art historian Jack Wasserman as part of a comprehensive study of the piece. The requirements are to obtain a model that can be manipulated to study overall problems of composition and form, and can also be examined in close detail to study tool marks and fine repairs. While there are not any substantial variations in hue on the piece, there are many variations in surface reflectance as a result of repairs, applied coatings, and wear on the sculpture. The subtle variations in the tone and color do not represent the artist's original intent, but are useful to document for studying the treatment of the piece since it was originally created in the sixteenth century.

*{holly,fausto}@watson.ibm.com

Our project goals also included developing a relatively inexpensive scanning system in a short (i.e. months) time frame. Rather than have a custom laser scanning system fabricated, we developed a system using readily available cameras and lights. Because measurements needed to be made in many short sessions in the museum environment, our system had to rely as little as possible on high precision positioning and control of electrical power levels. Work in a similar environment has been described in [3].

Our system consisted of the Virtuoso multiview shape camera from Visual Interface [11], augmented with an in house lighting system consisting of five halogen lights (see Figure 1). The system, described in more detail in [9], allows the capture of five images with each of the five lights turned on in sequence, registered to the geometry acquired by the Virtuoso shape camera. We obtained several hundred overlapping meshes covering the sculpture. Each geometric mesh has a resolution of approximately 2mm, and an estimated sub-millimiter accuracy. A set of normals and colors are computed from the five photometric images at a resolution of between 0.5 and 1 mm. An example of the set of data acquired from a single camera pose is illustrated in Figure 1.

Variations in lighting and positioning result in slight but discernible variations in the photometric results from mesh to mesh. In the rest of this paper we discuss how we correct these variations. The normals maps are made consistent with the underlying mesh by locally adjusting the light source intensities used in the calculations, using data from the underlying mesh. By using a consistent global underlying mesh computed by registering and remeshing the underlying base meshes, the normals maps are consistent with one another, as well as with the global mesh. We also compute corrected red, green and blue maps from the photometric data. Chromatic varations from map to map are corrected with a global color registration analogous to the global geometric registration used to obtain the global base mesh.



(a) (b) (c)

Figure 1: (a) The scanning system used for data acquisition. The Virtuoso multiview shape camera is augmented with a custom-made lighting system. Five color images, registered to the captured geometry, are taken with the five light sources turned on in sequence. (b) Triangle mesh computed by the Virtuoso software from the six striped images. (c) Four of the five photometric color pictures captured by our system.

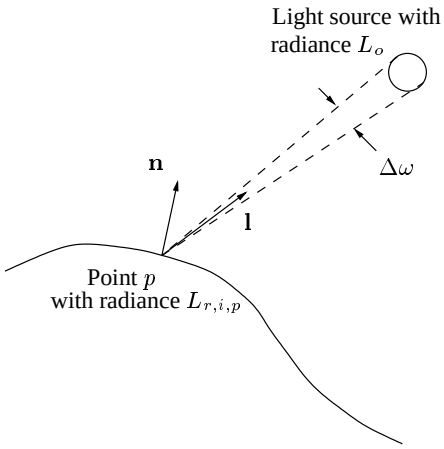


Figure 2: Geometry of light reflectance used in Eq. 1

2 Photometric Stereo – Calculations and Problems

The photometric stereo method for computing three dimensional shapes was originally developed by Woodham [7]. The essence of the method is to take a series of images from the same camera view, each with lighting from a different, known, direction. Assuming an approximately Lambertian bidirectional reflectance distribution function (BRDF) and identical light sources, the relative reflectance and normal at each pixel location can be computed from three images. Specifically, the following set of equations are solved:

$$\frac{\rho_p}{\pi} L_o \Delta\omega \begin{bmatrix} l_{1,x} & l_{2,x} & l_{3,x} \\ l_{1,y} & l_{2,y} & l_{3,y} \\ l_{1,z} & l_{2,z} & l_{3,z} \end{bmatrix} \begin{bmatrix} n_{p,x} \\ n_{p,y} \\ n_{p,z} \end{bmatrix} = \begin{bmatrix} \alpha L_{r,1,p} \\ \alpha L_{r,2,p} \\ \alpha L_{r,3,p} \end{bmatrix} \quad (1)$$

where ρ_p/π is the BRDF at pixel p, L_o is the light source radiance, $L_{r,i,p}$ is the radiance reflected from the point visible through pixel p in image i, $\Delta\omega$ is the solid angle subtended by the light source, $\mathbf{n}_p$ is the surface normal at p and $\mathbf{l}_i$ is the direction vector to the (infinitely distant) i-th light source (see Figure 2). The constant α accounts for the camera scaling of the reflected radiance to a value from 0 to 255, after the images are adjusted if necessary for gamma values other than 1. The equations can be solved directly for $(\rho_p L_o \Delta\omega/\alpha\pi)\mathbf{n}_p$. Since the magnitude of $\mathbf{n}_p$ is one, from this result we can obtain the normal vector $\mathbf{n}_p$ and a value $\rho_{rel,p} = \rho_p L_o \Delta\omega/\alpha\pi$, that is the reflectance at pixel p relative to the other pixels in the image.

Two difficulties with this approach are the presence of a highly specular component in the BRDF of the surface being measured, and shadows. Following the approach used by other researchers (E.g. see [6]) we obtain extra images from additional light source positions. For each pixel location, pixels with very high values are not used to exclude specular reflections, and pixels with low values are not used to exclude shadows. We use relative value only rather than color to distinguish possible specular reflection [10], since there are spectrally flat regions in the statue for which such methods would fail. Pixel locations

where fewer than three images have pixel values that fall in the valid range are not used in the computations that follow.

In our system we used a system of five lights, i.e. two redundant reflectance measurements per pixel location. This number was used because it allowed a set of lights with a significant physical separation while keeping to an overall manageable size for mounting on a tripod.

In a controlled environment, the term L_o can be made the same for each source for all points on the scanned target by using controlled, identical, isotropic light sources. The term $\Delta\omega$ can be made constant for all points and all sources by placing the sources at a distance from the target that is large relative to the sizes of the light sources and the target.

In an inexpensive system used in an environment such as the museum, any number of variations in light source radiances and position can occur:

- The sources may not be identical. The radiance from individual bulbs of the same type will vary slightly, particularly for inexpensive off-the-shelf bulbs.
- The sources may not be isotropic. In our case we used halogen bulbs for their brightness and ease of use. The emitted radiance is approximately uniform in a 30 degree cone around the direction that the source is aimed. There are small variations across the scanned area, and small disturbances caused slight changes in the source direction.
- The sources may vary over time. The emittance radiance varies with the age of the bulb. It also varies with each use as a function of time since the bulb was turned on.
- The source radiance varies with the electrical power level. A regulated power supply is required to maintain a consistent source level.
- The distance from the light sources to the area being scanned varies across the piece. Requiring the distance to the light sources to be large relative to the scanned area would result in very small areas being scanned in each shot to obtain the spatial resolution required.
- As well as overall strength, the light source spectrum varies bulb to bulb, with time and with level of power input.

All of these factors can be treated by high precision characterization and control of the lighting system, and by restricting the spatial extent of data acquired for each set of photometric images. However, such precision adds greatly to the cost of the system, and to the time required to acquire the data for the full sculpture. Instead, we deal with these problems by exploiting the existence of an underlying lower resolution mesh, and by performing a global registration of results.

3 Revised Calculation of Normals

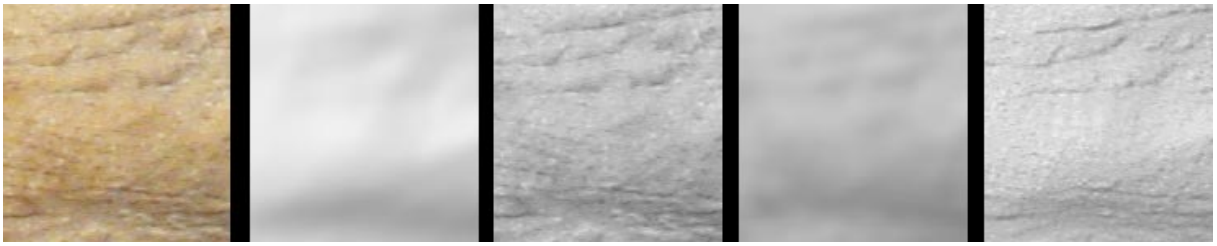
Because of the variations from an ideal photometric setting, we need a method to solve the system in Eq. 1 expressed with variable light source radiances and solid angles. We assume that the radius d of the light sources is the same for all the sources, and that d is small (i.e. less that 10 per cent) relative to the distance $r_{i,p}$ from surface point p to each light source i so that the solid angle $\Delta\omega_i$ for each source can be approximated by $\pi d^2/r_{i,p}^2$. We denote the radiance of light source i at pixel location p with $L_{o,i,p}$. Eq. 1 can be rewritten as

$$\rho_p d^2 \begin{bmatrix} l_{1,p,x} & l_{2,p,x} & l_{3,p,x} \\ l_{1,p,y} & l_{2,p,y} & l_{3,p,y} \\ l_{1,p,z} & l_{2,p,z} & l_{3,p,z} \end{bmatrix} \begin{bmatrix} n_{p,x} \\ n_{p,y} \\ n_{p,z} \end{bmatrix} = \begin{bmatrix} \alpha L_{r,1,p} {r_{1,p}}^2 / L_{o,1,p} \\ \alpha L_{r,2,p} {r_{2,p}}^2 / L_{o,2,p} \\ \alpha L_{r,3,p} {r_{3,p}}^2 / L_{o,3,p} \end{bmatrix} \quad (2)$$

All of the quantities in the above equation except for α and d vary from pixel to pixel across the image.

With our hybrid system, we obtain a geometric mesh at a spatial resolution of approximately 2mm registered with each set of five photometric images obtained. The individual meshes (Figure 1) are registered to one another using a two pass method that first uses a set of laser dots projected onto the statue for an initial alignment, and then a form of Iterative Closest Point (see e.g. [4]) for a final more precise registration. The meshes are combined into one single mesh using the Ball-Pivoting Algorithm described in [5]. For each set of photometric images we have in the end a consistent underlying mesh, and a camera transformation that gives the camera parameters for the photometric image capture in terms of the global coordinate system. Given these data, we can easily compute the following for each pixel of the photometric images:

- The distances $r_{i,p}$ from the visible surface point p to each of the light sources i. While there is some error in this distance approximation, the error is well under one percent over the typical 0.75 meter distance from the surface to the camera.
- The direction $\mathbf{l}_{i,p}$ from the visible surface point to each light source.
- An approximate surface normal $\mathbf{n}'_p$ at the visible point computed from the lower resolution surface.



(a) (b) (c) (d) (e)

Figure 3: Intermediate images used in computing normals. (a) A color image of a small section of the statue. (b) The underlying geometry computed for this section, with a reflectance of one, lit from the front. (c) A grey scale version of the section in one of the photometric images (the one with the lighting from directly in front) and (d) a grey scale image after averaging. (e) The final surface normals (as lit from the front) computed from converting all of the photometric light images into the forms shown in (c) and (d).

We can immediately use the underlying values of $\mathbf{l}_{i,p}$ and $r_{i,p}$ as good estimates for the values in Eq. 2. We need to find estimates for the relative values of $L_{o,i,p}$ that account for the temporal, directional and bulb to bulb variations of sources. To do this, we equate the relative reflected radiances from the underlying surface illuminated by ideal sources, to the relative image radiances in the neighborhood of the pixel.

Consider ideal light sources with uniform radiances L_u in the same positions as the physical light sources. If these ideal sources illuminated the underlying base surface coated with a Lambertian reflectance of 1, the reflected radiances would be:

$$\tilde{L}_{r,i,p} = (L_u d^2 / r_{i,p}^2)\ \mathbf{l}_{i,p} \cdot \mathbf{n}'_p \tag{3}$$

For the i-th photometric image, the reflected radiance in the neighborhood around pixel p is

$$\alpha \overline{L}_{r,i,p} = \sum_{q}^{Q} \alpha L_{r,i,q} / |Q| \tag{4}$$

where we consider a neighborhood of pixels Q that approximately represents the area on the surface of a disk with a radius equal to the base geometry resolution (i.e. for our system 2 mm). Notice that from the images, we do not have $L_{r,i,p}$ directly, but the quantity $\alpha L_{r,i,p}$ as recorded by the camera. The values $\alpha L_{r,i,q}$ are the result of recording the reflection from the true physical sources:

$$\alpha L_{r,i,q} = \rho_q \alpha L_{o,i,q} d^2 / r_{i,q}^2\ \mathbf{l}_{i,q} \cdot \mathbf{n}_q \tag{5}$$

We make the assumption that although ρ_p, $L_{o,i,p}$, $r_{i,p}$, and $\mathbf{l}_{i,p}$ vary pixel by pixel over the entire image, they vary relatively little in the small neighborhood around pixel p. We also assume that the average over Q of the dot products $\mathbf{l}_{i,q} \cdot \mathbf{n}_q$ is approximately equal to the dot product of $\mathbf{l}_p$ and the underlying surface normal $\mathbf{n}'_p$. Note that these assumptions are the same basic assumptions made in all photometric stereo calculations – with the exception that we now make the assumptions in a (small) region of pixels rather than for just the surface represented by a single pixel. With these assumptions we have:

$$\alpha \overline{L}_{r,i,p} \simeq \rho_p \alpha L_{o,i,p} d^2 / r_{i,p}^2 \mathbf{l}_{i,p} \cdot \mathbf{n}'_p \tag{6}$$

We now form the ratio of the radiance reflected from the underlying surface given in Eq. 3 to the average recorded radiance in the pixel neighborhood given in Eq. 6:

$$\tilde{L}_{r,i,p} / \alpha \overline{L}_{r,i,p} \simeq L_u / \rho_p \alpha L_{o,i,p} \tag{7}$$

We can use Eq. 7 to express the unknown source radiances $L_{o,i,p}$ in terms of quantities we can compute, or which are the same for all light source directions:

$$L_{o,i,p} \simeq \frac{L_u \alpha \overline{L}_{r,i,p}}{\rho_p \alpha \tilde{L}_{r,i,p}} \tag{8}$$

Using Eq. 8 in the right hand side of Eq. 2, and simplifying terms results in:

$$\frac{L_u}{\alpha} d^2 \begin{bmatrix} l_{1,p,x} & l_{2,p,x} & l_{3,p,x} \\ l_{1,p,y} & l_{2,p,y} & l_{3,p,y} \\ l_{1,p,z} & l_{2,p,z} & l_{3,p,z} \end{bmatrix} \begin{bmatrix} n_{p,x} \\ n_{p,y} \\ n_{p,z} \end{bmatrix} = \begin{bmatrix} \alpha L_{r,1,p} {r_{1,p}}^2 \tilde{L}_{r,1,p} / \alpha \overline{L}_{r,1,p} \\ \alpha L_{r,2,p} {r_{2,p}}^2 \tilde{L}_{r,2,p} / \alpha \overline{L}_{r,2,p} \\ \alpha L_{r,3,p} {r_{3,p}}^2 \tilde{L}_{r,3,p} / \alpha \overline{L}_{r,3,p} \end{bmatrix} \tag{9}$$

We have left in the values of α since the quantities αL_r are the values actually available from the recorded images. Eq. 9 can be solved for a vector in the direction of the surface normal, and normalization gives the result $\mathbf{n}_p$. The effect of the surface reflectance ρ_p is in both the individual pixel values and the averaged pixel values. As a result the

reflectance term is canceled out of the equation, and the length of the vector in the direction of the surface normal no longer has a useful physical interpretation.

The new calculation process is illustrated in Figure 3.

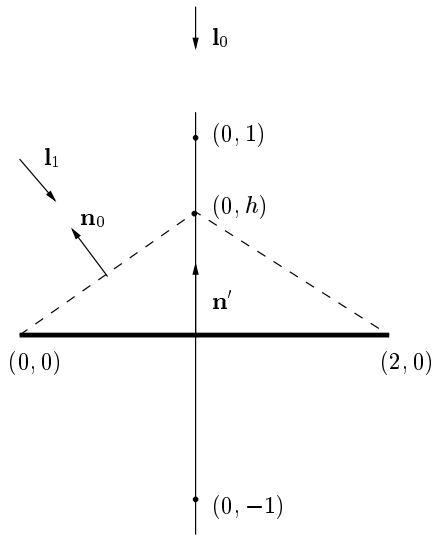
The accuracy of the resulting normals depends on the validity of the assumptions made in the approximation in Eq. 6. The small size of the neighborhood used for averaging relative to the distance to the camera (2mm versus 750mm), justifies the assumptions that $L_{o,i,p}$, $r_{i,p}$ and $\mathbf{l}_{i,p}$ are uniform. The reflectance is also reasonably assumed to be uniform in a small neighborhood. Exceptions to this are abrupt changes due to a change in surface coating. However, changes in reflectance cause a consistent change in all of the photometric images and can be detected. That is, a sharp increase in reflectance results in an increase in reflected radiance for all light source directions. The main assumption is the equivalence of the average of the dot product of the light direction and the normals on the high resolution surface and the dot product of the light direction with the normal of the underlying lower resolution surface.

To evaluate the assumption of equating the average and single dot products we consider the simple two dimensional system shown in Figure 4. We consider a horizontal surface, of width 2, that is approximated by the low resolution surface connecting the points $(0,0)$ and $(2,0)$. The surface is illuminated by two lights: $\mathbf{l}_0$ in direction $(0,1)$, and $\mathbf{l}_1$ in direction $(-\sqrt{2}/2, \sqrt{2}/2)$. We apply our calculation method to a higher resolution surface that consists of two facets formed by a displacement h in the center of the surface. We examine the accuracy of the result we obtain for the normal of the left facet as a function of h.

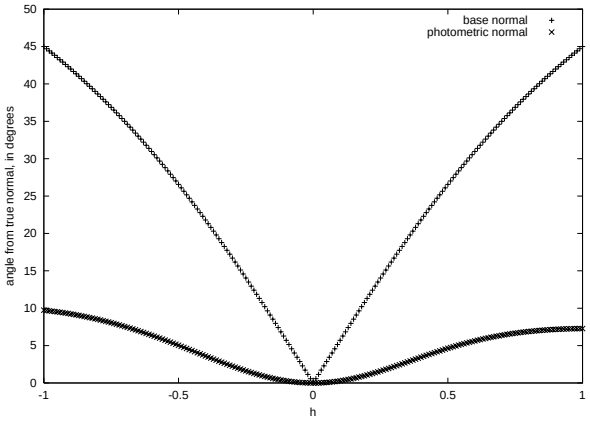
Specifically, we compute the following quantities:

- The underlying surface normal $\mathbf{n}' = (0,1)$ for all h.
- The true facets normals $\mathbf{n}_0 = (-h,1)/\sqrt{1+h^2}$ and $\mathbf{n}_1 = (h,1)/\sqrt{1+h^2}$
- The reflected light for each facet for each light, $L_{r,i,j} = \mathbf{l}_i \cdot \mathbf{n}_j$
- The approximate surface normal from Eq. 9, using $\overline{L}_{r,i} = L_{r,i,0} + L_{r,i,1}$ and $\tilde{L}_{r,i} = \mathbf{l}_i \cdot \mathbf{n}'$ and setting the constant factor to one.

The results of these calculations are shown in Figure 4. The plot compares the angle between the estimated and true normal from using the underlying surface normal, and from using the approximate normal from the photometric method. The non-linearity of the dot products with normal vectors accounts for the errors in the photometric method, and for the asymmetry between negative and positive values for h. Although the error in the photometric method



(a)



(b)

Figure 4: (a) The two dimensional system used to evaluate the validity of assumptions made in computing the normals. (b) Results of applying our method for normals computation.

increases with the absolute value of h, the photometric estimate of the normal is in all the cases a significant improvement over using the underlying normal.

The above calculations do not account for interreflections that would occur when h is negative. In our application we are measuring a surface with low reflectance and so have relatively little indirect illumination from interreflections. For surfaces with high reflectance and steep slopes in the high resolution surface there would be additional error in the computed normals.

4 Color Calculations

Given an estimate of the surface normal at each pixel, we can compute versions of the photometric input images with the shading effects due to the directional lighting removed. By taking a weighted average of these corrected images, we obtain an estimate of reflectances in the red, green and blue color channels. Unlike scans with polychromatic laser [2], we can not obtain the absolute value of the reflectance in each channel, or a precise spectral definition of each channel. However, using some additional spot measurements of spectral reflectance we can adjust the corrected images to approximate absolute reflectance in defined spectral ranges.

Denoting the wavelength of light as λ and using the subscript C to denote the color channel that is either red, green or blue, the recorded color value $\alpha L_{r,i,p,C}$ of each pixel p in each photometric image i is given by:

$$\alpha L_{r,i,p,C} = \int_0^\infty \rho(\lambda) L_{o,i,p}(\lambda)(d^2/r_{i,p}^2)\, \mathbf{l}_{i,p} \cdot \mathbf{n}_p S_C(\lambda)\, d\lambda \tag{10}$$

where $S_C(\lambda)$ is the spectral sensitivity of the camera in channel C.

Using the normals from the previous section, we compute an approximate relative reflectance ρ_{rel} for each channel from:

$$\rho_{rel,i,p,C} = \alpha L_{r,i,p,C} {r_{i,p}}^2 / \mathbf{l}_{i,p} \cdot \mathbf{n}_p \tag{11}$$

This reflectance is relative to other pixels in the same channel in the same image. As in the case of the calculation of normals, we do not use high or low pixel values to avoid using data in areas of specular reflection and shadow. Unlike the normals calculations, we do not need a pixel estimate from three images to estimate the relative reflectance. Because we can afford to use less of the image, we avoid including the effects of directionally varying light source intensity by only using pixels that fall in a small cone around the direction the light source is aimed.

The five relative reflectance images computed with Eq. 11 are adjusted relative to one another to compensate for variations in light source strength. The pixels for which all five of the images have non-zero values are found, and the averages for these overlapping pixels are computed. The overall level of the images is adjusted by making all of the average values equal.

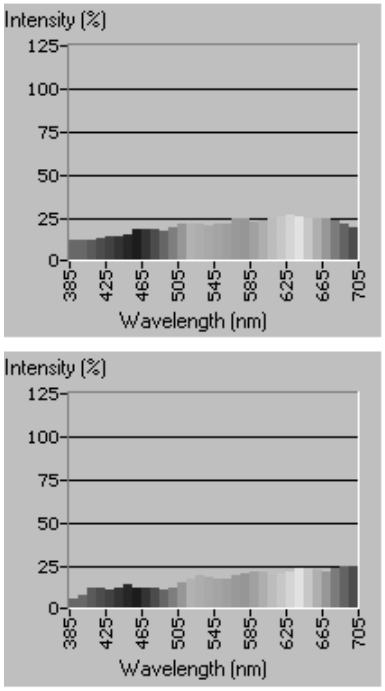


Figure 5: Typical reflectance spectra measured on the statue.

Once the images levels are adjusted, the five images are combined with a weighted average to produce a single corrected RGB image for the camera position. The weight assigned to each pixel in each image increases with distance to the nearest black edge in the image. This weighting is used to avoid small but sudden changes at edges where each photometric image stops contributing to the combined average.

Clearly the relative reflectances computed from Eq. 11 are not scaled to the actual percentage of visible light that would be reflected. The images also still contain the effects of the light source spectra and the camera spectral sensitivity. Furthermore, the red, green and blue channels are not defined in terms of wavelengths. To deal with these problems, we made separate spot measurements of spectral reflectance using the Colortron color ruler [8]. Two typical spectra measured on the sculpture are shown in Figure 5. The spectral reflectances are simple functions in this case, and we can readily define spectral bounds for red, green, and blue for which the spectral reflectances vary little from the average for the range. That is we can define $\lambda_{C,min}$

and $\lambda_{C,max}$ and compute

$$\rho_{ave,C} = \int_{\lambda_{C,min}}^{\lambda_{C,max}} \rho(\lambda)/(\lambda_{C,max} - \lambda_{C,min})\, d\lambda \quad (12)$$

By taking the ratio of $\rho_{ave,C}$ and $\rho_{rel,C}$ at the location in the image where the spot measurement was taken, we can estimate the scaling for light source and camera spectral effects in the image. Using the scaling for each channel in the image, we can obtain at least approximate values for the absolute values of reflectance in clearly defined spectral ranges. We are still performing tests to better quantify the quality of the estimates we can obtain.

Once we have a few images that have been adjusted to approximate reflectances in the three color channels, we can adjust all of the images for which $\rho_{rel,C}$ has been computed by requiring the colors in matching points in overlapping images to be the same. As in the case of geometric registration, it is inadequate to simply perform these adjustments pairwise, since small errors accumulate. Instead we do a simultaneous, global registration.

Forming equations for the color registration is facilitated by the availability of the points used in the initial geometric alignment. One result of the initial alignment is a list of points in overlapping images that represent the same location on the base geometry. We compute a color for each of these points by taking the average of a small neighborhood of pixels in the corrected image. We use a neighborhood rather than a point sample since very small alignment errors (e.g. of the size of one pixel projected onto the surface) can produce a very large color error when there are abrupt changes in colors between pixels. Let $\beta_{C,k}$ be the level adjustment for the k-th corrected photometric image in the C channel. Each matching pair of points gives an equation of the following form for images m and n:

$$\rho_{rel,C,m}\beta_{C,m} - \rho_{rel,C,n}\beta_{C,n} = 0 \quad (13)$$

There are far many more matching points than corrected images, so a least squares solution is used to compute the values of β. In Eq. 13 no level has been set for the system. We could set the value of β for the image (or images) that have been adjusted with the Colortron measurements to one. However, we want the errors in the results to be distributed evenly over the model, rather than being skewed by distance from a particular image. To accomplish this, the following equation is added to the set:

$$\sum_{k}^{N} \beta_{C,k} = N \quad (14)$$

where N is the total number of images. Using N on the right hand side of Eq. 14 produces values of β around 1 (i.e. from about 0.6 to 1.4 in our tests). After all of the images have been adjusted to one another, global correction values for red, green and blue can be made to match the Colortron measurements.

5 Results and Conclusions

The results of using our new method for computing colors and normals are shown Figures 6 and 7. Figure 6 shows a small 20cm-wide section of the statue. The top row of images shows the low resolution geometry lit from below, directly in front and from the right. The lower row of images shows the same section with the normals computed using the photometric method lit from the same directions. Figure 6 demonstrates that the only difference between the original base surface and the photometric results are high spatial frequency details. The appearance of the details varies a great deal with lighting direction, and so could not be represented with a simple texture map.

Figure 7 demonstrates that the normals and colors computed from various camera positions are consistent with one another. Pictures (a) and (b) in Figure 7 show 72 meshes with their default texture maps, used to reconstruct a 2.25m tall section of the statue. The patchwork effect is due to the varying lighting conditions as the camera is moved around the statue. Image (c) shows the model with the computed photometric normals. The seams between the individual meshes are not visible. The next image shows the color reflectances computed from the 72 camera viewpoints. This is the result of the global color registration only, the final adjustments to true spectral quantities have not been made.[1] Image (e) shows a rendering of the model with photometric normals and colors.

We have presented a method for successfully computing consistent normals and colors from photometric data from many different camera viewpoints. We are continuing to refine our color calcualtions to better represent the actual spectral characteristics of the measured surface. Our method produces many overlapping normals and color maps for each area on the surface. We are exploring methods for selecting and weighting the best maps to use in each area.

References

[1] J. Abouaf. The Florentine Pietà: Can visualization solve the 450-year-old mistery? *IEEE Computer Graphics and Applications*, 19(1):6–10, February 1999.

[2] R. Baribeau, M. Rioux, and G. Godin. Color reflectance modeling using a polychromatic laser range sensor. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1992.

[1] Note that after true spectral values are computed, color accurate images will have to be computed taking into account the color display characteristics under which the images are being viewed to accurately represent the statue.

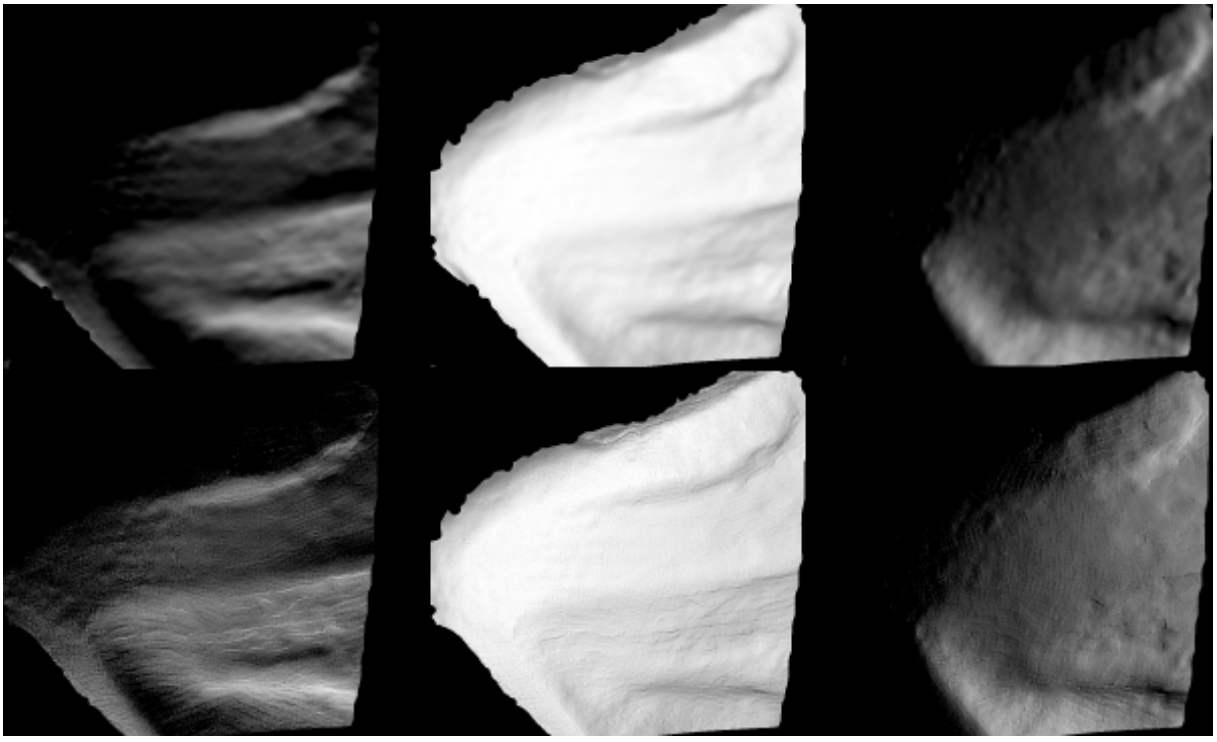
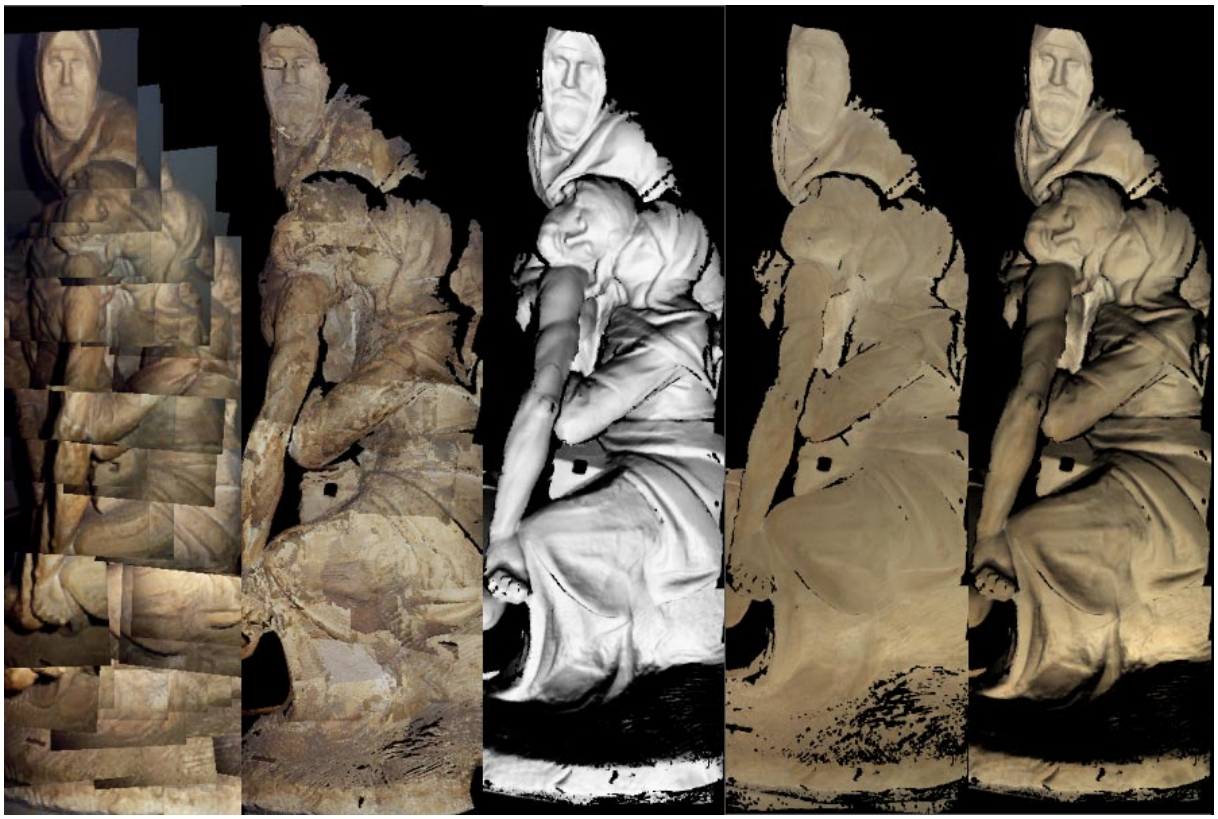


Figure 6: A comparison of the low resolution model (top row) and the same model augmented by the normals computed with our method. The three columns from left to right represent the same section of the statue lit from below, directly in front and from the right.



(a) (b) (c) (d) (e)

Figure 7: Results of our method applied to the Pietà data. From left to right: (a) A montage of sample photometric images of the statue, showing the wide range of variation in illumination conditions. (b) Inconsistencies are evident in the patchwork effect resulting from stitching together the meshes and textures. (c) Photometric normals applied to the base mesh. (d) Color reflectance computed with our method. (e) A rendering of the combined normals and color maps.

[3] J.-A. Beraldin, F. Blais, L. Cournoyer, M. Rioux, F. Bernier, and N. Harrison. Portable digital 3-D imaging system for remote sites. In *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, pages 488–493, 1998.

[4] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multi-view registration technique. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(5):540–547, May 1996.

[5] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. Submitted for publication, 1999.

[6] R. Epstein, A. Yuille, and P. Belhumeur. Learning object representations from lighting variations. In *ECCV 96 International Workshop*, pages 179–199, April 1996.

[7] B. K. P. Horn and M. J. Brooks. *Shape from Shading*. MIT Press, 1989.

[8] Light Source, Inc. Colortron. *http://www.ls.com/colortron.html*.

[9] H. Rushmeier, F. Bernardini, J. Mittleman, and G. Taubin. Acquiring input for rendering at appropriate levels of detail: Digitizing a Pietà. In *Proceedings of the Ninth Eurographics Rendering Workshop*, pages 81–92, June 1998.

[10] Y. Sato, M. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 379–388, August 1997.

[11] Visual Interface, Inc. Virtuoso. *http://www.visint.com/*.

Annotated Bibliography

ARTICLES RELATED TO DIGITIZING THE FLORENTINE PIETA‘

Abouf, Jeffrey, “The Florentine Pieta‘: Can Visualization Solve the 450-Year-Old Mystery”, **IEEE Computer Graphics and Applications, January/February 1999,** pp. 6-10, a description of the project written for the Applications department of this journal by an independent author. While we supplied the information for the article, the author does supply an outside point of view on the project.

Rushmeier, Holly, Fausto Bernardini, Joshua Mittleman and Gabriel Taubin, “Acquiring Input for Rendering at Appropriate Levels of Detail:Digitizing a Pieta”, **Rendering Techniques ’98, Proceedings of the 9th Eurographics Rendering Workshop**, Drettakis and Max, Eds., Springer, Vienna, 1998, pp. 81-92, a description of the basic strategy of combining the multi-view and photometric stereo techniques. The emphasis is on the idea that if normals maps are going to be used to represent and render the model efficiently, it makes sense to acquire them directly rather than deriving them from dense point samples.

Gueziec, A. “Surface simplification inside a tolerance volume,”, **IBM Research Report, RC20440**, 1997. This simplification technique was developed before our project started. This technique has been essential in allowing us to view intermediate results on the statue.

REFERENCES WE HAVE FOUND USEFUL

Proceedings on Recent Advances in 3-D Digital Imaging and Modeling, Gerhard Roth and Marc Rioux, Eds, IEEE Computer Society Press, 1997. These proceedings give an excellent snapshot of work in the area of scanning, from sensors, to processing and applications. Besides the content of the papers, it is an excellent guide to research groups that are active in this area so that you can check their web pages to see what is going on currently. A second 3DIM conferences will be held in October 1999.

Computer Vision: Three-Dimensional Data from Images Reinhard Klette, Karsten Schluens, Andreas Koschan, Springer, 1998. This is a good, basic, introduction to the var-

ious methods developed over the years in computer vision for acquiring three dimensional data. Stereo, shape from shading, and structured light are discussed, as are the basics of cameras and the reflection of light.

Besl, Paul J. and Neil McKay, "A Method of Registration of 3-D Shapes" **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Vol. 14, NO. 2, pp. 239-256. The iterative closest point method is described here. Many algorithms for registration are variations of one sort or another of this basic method.

Bergevin, Soucy, Gagnon, and Laurendeau, "Towards a General Multi-view Registration Technique," **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Vol. 18, No. 4, pp. 540-547. This is basically the method we used to enforce global, rather than just pairwise alignment.

Mencl, R. and Mueller, H. "Interpolation and Approximation of Surfaces from Three Dimensional Scattered Data Points" **STAR Report, Eurographics 1998 Conference**. A recent state-of-the-art report on building meshes from scattered points.

Bernardini F., C. Bajaj, J. Chen and D. Schikore, "Automatic reconstruction of 3D CAD models from digital scans", **Interational Journal of Computational Geometry and Applications**, to appear 1999. This article has an extensive survey of reconstruction techniques.

Ghodsi, M. and J.-R. Sack "A Coarse Grained Parallel Solution to Terrain Simplification," **Proceedings of the Tenth Canadian Conference on Computational Geoemtry** McGill Univeristy, Montreal, Quebec, Canada, August 9-12, 1998. This short (two page) describes the basic strategy of splitting and recombining the geometry that we used in simplifying the model.

ONLINE RESOURCES

The Visual Information Technology group at the National Research Council of Canada are the leaders in scanning technology – in hardware, software and applications. They have excellent web pages about their work and links to other work at:
http://www.vit.iit.nrc.ca/

Marc Levoy's group at Stanford has an extensive project scanning many of Michelangelo's works. They maintain pages detailing many facets of their work at, including fabulously detailed results for the massive statue of David at:
http://www.graphics.stanford.edu/projects/mich/

CNUCE, Pisa, Italy. The Visual Computing Group has an ongoing project to scan cultural

heritage pieces, part of a broader effort to use computer technology to help in documenting and preserving art. contact: R. Scopigno
http://vcg.iei.pi.cnr.it

The Geometric Modeling Group at the University of Dortmund, Germany. Research in automatic reconstruction from scans. contact: Heinrich Mueller
http://ls7-www.informatik.uni-dortmund.de

IGD Darmstadt, Germany, Department Cognitive Computing Medical Imaging. Examples of reconstruction of statues. contact: P.J. Neugebauer
http://www.igd.fhg.de