Curvature Analysis for Three-Dimensional Tracking of Computer Vision Data

Adam Sanford

Cornell University

Lawrence Livermore National Laboratory
Livermore, California 94550

December 13, 1995

Prepared in partial fulfillment of the requirements of the Science and Engineering Research Semester under the direction of Dr. Shin-Yee Lu, Research Mentor, in the Lawrence Livermore National Laboratory.

* This research was supported in part by an appointment to the U.S. Department of Energy Science and Engineering Research Semester (hereinafter called SERS) program administered by LLNL under Contract W-7405-Eng-48 with Lawrence Livermore National Laboratory.

CONTENTS & THESIS

Thesis:

Markerless imaging systems could potentially track data points with an analysis of the local curvature of its acquired surfaces.

Contents:

I. ABSTRACT

The purpose of this project is to develop a method of tracking data points for computer vision systems using curvature analysis. This is of particular interest to fellow researchers at the Lab, who have developed a markerless video computer vision system and are in need of such a method to track data points. A three dimensional viewing program was created to analyze the geometry of surface patches. Virtual surfaces were plotted and processed by the program to determine the Mean and Gaussian Curvature Parameters for each point on the surface, thus defining each point's surface geometry type. Preliminary results indicate that coordinate independent curvature analysis shows great promise and could solve the tracking dilemma faced by those in the field of markerless imaging systems.

II. INTRODUCTION

A new computer vision technology, called CyberSight, has been developed at Lawrence Livermore National Laboratory. It is capable of acquiring three-dimensional range data of moving surfaces at video speed. It was developed by LLNL researchers Lu and Johnson, with whom I worked this past semester.

CyberSight shows great promise in many fields. It has applications in biomechanics, security, entertainment, and other fields. For biomechanical analyses, CyberSight could be used for studying virtually any motion, generating precise data which could be easily reduced to yield kinematic data. For security purposes, it could be adapted for facial recognition. Movie companies are interested in the possibilities of using "virtual actors" in their films, with CyberSight's fine detail of musculature and bony prominences.

However, these applications are not yet possible for CyberSight. As a markerless data acquisition system, data tracking methods must be developed to relate different frames from concatenated movies. Though tracking has been a dilemma in computer vision for quite some time, it is the subject that was worked on in this project. The goal is to lay a foundation for future efforts in computer vision tracking.

III. CURVATURE ANALYSIS

The specific possibility of tracking that was investigated was the analysis of local curvature. This seemed a logical area of inquiry, as CyberSight is capable of generating very fine[1] surfaces to analyze.

The crux of curvature analysis lies in the ability to acquire the first and second order partial derivatives of the surface at any point of inspection. This is due to the fact the surface geometry type is governed by two parameters, the Mean Curvature and Gaussian Curvature, which are defined by these partial derivatives as follows[2]:

Mean Curvature, H:

$$H = \frac{(1+z_y^2)z_{xx} + (1+z_x^2)z_{yy} - 2z_x z_y z_{xy}}{2(1+z_x^2+z_y^2)^2}$$

Gaussian Curvature, K:

$$K = \frac{z_{xx}z_{yy} - z_{xy}^2}{(1+z_x^2+z_y^2)^2}$$

Based on the sign of H and K, the surface geometry type is then defined:

$H < 0,\ K > 0$	Peak	$H < 0,\ K < 0$	Saddle Ridge
$H > 0,\ K > 0$	Pit	$H = 0,\ K < 0$	Minimal Surface
$H < 0,\ K = 0$	Ridge	$H > 0,\ K < 0$	Saddle Valley
$H = 0,\ K = 0$	Flat	$H > 0,\ K = 0$	Valley

[1] CyberSight generates surfaces at a spatial resolution of roughly one millimeter (mm).
[2] Li, page 233.

Curvature analysis is attractive in addressing the issue of tracking. H and K are invariant to translations and rotations[1], so objects of constant surface geometry undergoing these transformations in space can still be recorded with CyberSight and still yield the same geometry type if analyzed by curvature analysis routines. As no curvature analysis packages were readily available, the research was focused on creating such a package for insertion into CyberSight's existing image processing code for later use.

IV. METHODOLOGY

A. PointGrab as a visualization tool

CyberSight's final, processed images are roughly 500 X 500 pixels, and are viewed with a graphics package called VISION. This package requires a rather complex interface, with the user inputting Lisp commands to call C code. In addition, VISION displays a set image, and cannot view data from any viewpoint. Considering these hindrances with its rather slow processing time, and the choice to write stand alone software to replace VISION was made.

A graphics program was written to carry out the research. At first written to serve basic needs, more complex routines were added to the program as required. The program, called PointGrab, has been the primary visualization tool of this research throughout the semester.

---

[1] *Ibid.*, page 233.

PointGrab has many functions, including the following:

- View a three dimensional data set from any spherical viewpoint
- Transform and stretch data sets in space
- Plot a curvature map of a data set
- Yield percentage curvature compositions of a data set

B. Virtual shapes as a test of routines

CyberSight is not a perfect system -- that is, it is not free from error. From the frame-grabbing phase to each of the processing steps, error can be introduced to the data. With this in mind, testing any curvature analysis routines on real, processed images would not be prudent. Instead, virtual shapes were created as control surfaces for the testing of any curvature algorithms.

Virtual shapes are nothing more than computer generated shapes. Since they are not "real," they need not be processed (erroneously) by any image processing routines. Therefore, they are mathematically "perfect" up to 8 decimal places[1], and serve as a natural control for any attempts at curvature analysis.

Virtual surfaces were created with a separate program for each shape, using the following process:

1) An orthogonal XY grid was created. This was usually 25 X 25, based on memory constraints.

2) For each point, its corresponding (x, y) location was found.

3) Each point's X and Y location were inserted into the equation for the shape generated by the program, yielding each point's Z value.

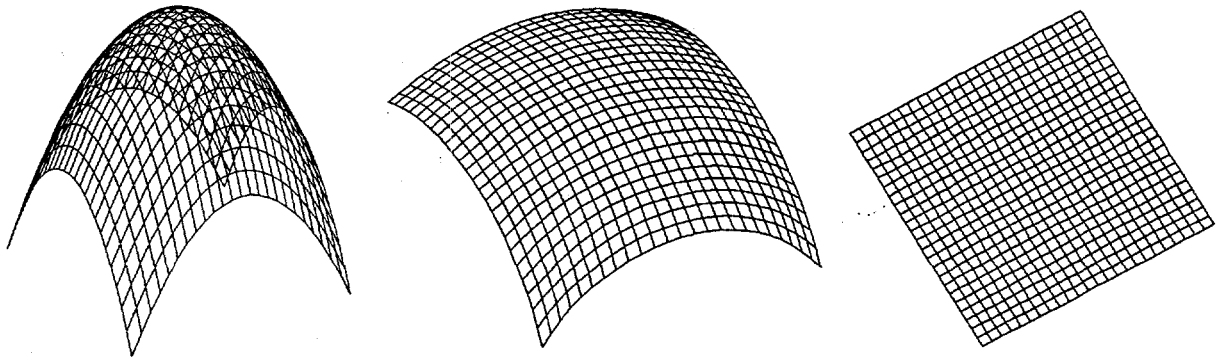
4) The data set was then input into a special file format that PointGrab could read, consisting of (x, y, z) points and line connections.

---

[1] The eight refers to the number of decimal places in a double precision floating point variable.

The goal was to generate as smooth a surface as possible, but aforementioned memory limitations prevented such surfaces from being generated for an entire shape. So a small surface patch of each shape was created for analysis.

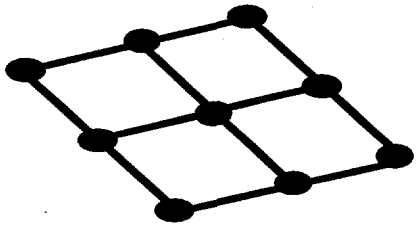
Virtual surfaces have been the data sets directly tested with the curvature algorithms this semester. Some of the shapes include a sphere, paraboloid, cylinder, cone, and plane. Some examples are below.



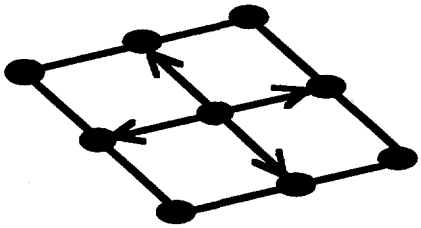
C. Surface normal generation

To achieve coordinate independent curvature analysis, a local coordinate system had to be defined for each point, with the new Z axis being each point's surface normal. Since the surface was a collection of points and not a continuous function where $z = f(x, y)$, the surface normal was not explicitly defined. So a method of generating each point's surface normal had to be devised.

As shown below, a 3 X 3 patch of points, centered about the point of interest, is generated for *every* point.

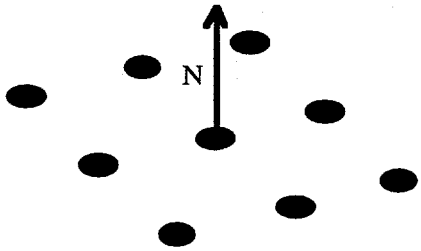


From each patch, four vectors are defined, originating from the center point and heading towards each of the four cardinal directions (one point excursions in the +/- X, Y directions) as shown.



Four independent surface normals were calculated by crossing the adjacent vectors in a circular order; that is, so that the cross products always pointed outward, or always pointed inward. Whichever direction that was chosen was set as the cross product convention for the surface normal generation of each patch.

The four surface normals were then averaged to yield the final (working) surface normal vector for each point, as shown below. This method is simple and was proven effective, barring any gross outliers in the data set.



D. Transformation about the surface normal

Once the surface normal had been calculated, it was necessary to make it the new Z axis. To do so, the data points in each patch needed to be transformed about their surface normals. However, general data transformations require three new axes to transform a system.

Since curvature is coordinate independent, the only requirement for analysis with local coordinate system is the surface normal as the Z axis. Therefore, the X and Y axes can be arbitrary, as long as they define an orthogonal coordinate basis. Here a decision was made: the new X axis was chosen as the direction from the center point (x, y) to the point one unit further down the old X axis (x+1, y). The Y axis naturally followed with Z cross X, and the orthogonal coordinate basis was determined.

From this new basis, the data in each patch is transformed with the following relation[1]:

$$^{N}q = {}^{R}T_{N}^{-1}\,{}^{R}q, \text{ where}$$

^{N}q represents the new (transformed) XYZ patch as [x′ y′ z′ 1] (row vector)

^{R}q represents the old (pre-transformed) XYZ patch as [x y z 1] (column vector)

$^{R}T_{N}^{-1}$ represents the inverse transformation, which is listed below:

[1] McKerrow, page 153.

$$\begin{bmatrix} x_x & x_y & x_z & -p_x x_x - p_y x_y - p_z x_z \\ y_x & y_y & y_z & -p_x y_x - p_y y_y - p_z y_z \\ z_x & z_y & z_z & -p_x z_x - p_y z_y - p_z z_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

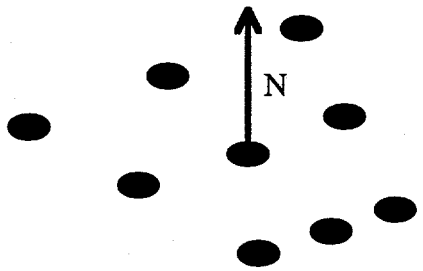
where

$$x = x_x i + x_y j + x_z k$$
$$y = y_x i + y_y j + y_z k$$
$$z = z_x i + z_y j + z_z k$$
$$p = p_x i + p_y j + p_z k$$

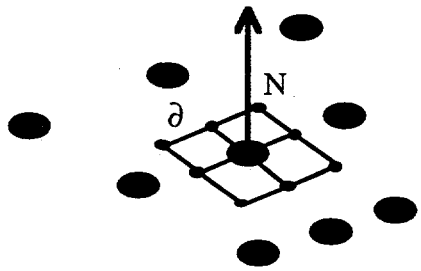
Above, x, y, z represent the new axes, and p represents the translation vector relating the old origin with the new origin (the patch center).

E. Reconstructing the orthogonal grid

Once the patch was transformed, it no longer retained the XY orthogonality that it once had. This was expected, as only a unit normal parallel to the old Z axis could retain orthogonal patch points. So each resulting patch was a bit distorted, as shown below:

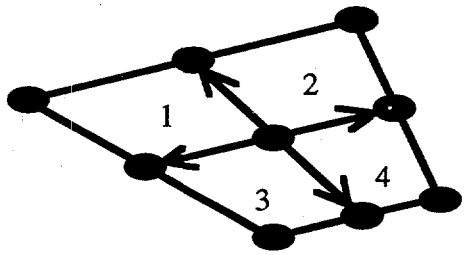


For H and K to be determined, a regular orthogonal grid needed to be reassembled in the new coordinate system. A smaller orthogonal patch, based on the new X and Y, was defined inside the old grid, with data points being separated by a distance ∂, as shown below:



The x and y values of the grid were predetermined +/- ∂ excursions about the center point, but the Z values were unknown. To find them, a planar interpolation was used, with the following procedure:

1) The four vectors are redefined, exactly like with the surface normal generation, but with the new transformed patch. This defines four quadrants, as shown.



2) The vectors were crossed, to generate four normals, just as before. However, the each normal was then used to define a plane for each quadrant[1]. The equation for each plane is of the form of $Ax + By + Cz = 0$.

3) An cross product routine determined which planes the projections of the new grid points fell on with respect to the Z axis.

4) The (x, y) values of each point on the grid were inserted into its corresponding plane equation, yielding the Z value of each point's projection on the plane.

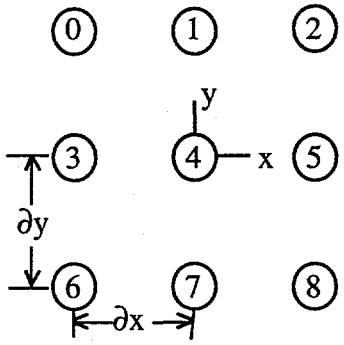
With the Z values determined, the new orthogonal data patch had been completed, and calculation of H and K could then commence.

[1] Thomas & Finney, page 734.

F. Acquiring H and K

After all of the data manipulation, acquiring H and K became a trivial task. Using grid point Z values, the partial derivatives were simply calculated. The newly constructed orthogonal grid made the calculus straightforward, as each data point was already separated an equal distance ∂.

Based on the new, transformed grid, shown below,



The following equations were used (z[n] equals the z value of point n):

$$z_x = \frac{(z[5]-z[4])}{\partial x} \qquad z_y = \frac{(z[1]-z[4])}{\partial y}$$

$$z_{xx} = \frac{\frac{(z[5]-z[4])}{\partial x} - \frac{(z[4]-z[3])}{\partial x}}{\partial x} \qquad z_{yy} = \frac{\frac{(z[1]-z[4])}{\partial y} - \frac{(z[4]-z[7])}{\partial y}}{\partial y}$$

$$z_{xy} = \frac{\frac{(z[2]-z[1])}{\partial x} - \frac{(z[5]-z[4])}{\partial x}}{\partial y}$$

As an aside, ∂x = ∂y = ∂ in the newly transformed grid, but the deltas have been left in their original format to clarify their usage in the equations.

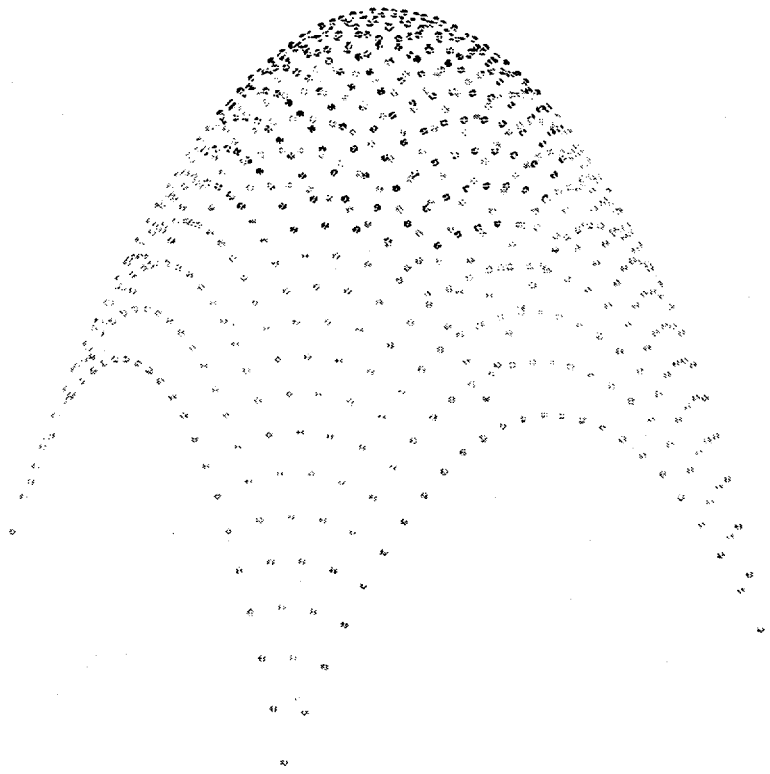
These equations were applied to each patch, calculating values for each of the partial derivatives necessary to evaluate H and K. H and K were than calculated, using the equations mentioned in the Curvature Analysis section.

G. Color Coding

After H and K were determined, the surface geometry type of each point was determined from the signs of H and K, as previously mentioned. What remained was a question of how the data could be identified graphically.

A color coding scheme was chosen to identify the point's curvature. This served to facilitate curvature identification (rather than by using special dots or shapes for points), and it made groupings of a set curvature type easy to pick out.

The actual carrying out of the color coding scheme was rather simple. A color was chosen and displayed for each geometry type, and a legend was displayed at the top of the graphics window. Percentage statistics of each curvature type present in a surface were also available in a text window.



For example, in the picture above, the curvature plot for a paraboloid is shown. The darker dots surrounding the top represent peak curvature, while the dots near the bottom represent ridge curvature. This plot is typical of curvature analyses carried out this semester.

V. RESULTS & CONCLUSIONS

The fuel for the pursuit of curvature tracking was the hope that curvature data could be truly coordinate independent and invariant with rotations and translations. After all of the aforementioned routines were programmed, compiled, and carried out, this indeed was the case.

Data was correctly marked for its curvature by PointGrab, which was not of any great import. But when the data was transformed arbitrarily in space, with various rotations and changes of origin, each point's curvature remained the same. This indeed did verify that curvature analysis is invariant to data transformations.

Current work is being done to insert PointGrab's algorithms for curvature analysis directly into CyberSight's image processing code. This would provide a means to apply curvature analysis to real data acquired by CyberSight, providing the real test for curvature tracking.

With all this said and done, a basic foundation has been laid for curvature tracking of computer vision data. Hopefully it can be put to good use towards the ultimate end of automatic tracking of curvature "landmarks" on scanned surfaces.

VI. ACKNOWLEDGMENTS

I would like to thank The University of California, the Department of Energy, and Lawrence Livermore National Laboratory for giving me the opportunity to take part in this research. I would also like to thank my mentor, Shin-Yee Lu, along with Robert Van Vorhis and Robert Johnson for their guidance and assistance during the semester.

In addition, I would like to thank Stanford Professor Eric Roberts for assisting me via e-mail with my questions regarding his C libraries that I used with PointGrab.

Last but not least, I would like to offer my thanks to fellow SERS student Marc Abramowitz, who showed great patience in bringing me, a mechanical engineering student, up to speed in the world of C and UNIX.

VII. BIBLIOGRAPHY

Ammeraal, Leendert. *Programming Principles in Computer Graphics*. John Wiley & Sons, New York: 1986.

Frobin, W., Hierholzer, E., Drerup, B. "Mathematical Representation and Shape Analysis of Irregular Body Surfaces." Pages 132-139, *SPIE BioStereometrics*, Vol. 361, 1982.

Hierholzer, E., Drerup, B. "Objective Determination of a Body-Fixed Coordinate System using Back Surface Data." Pages 262-265, *SPIE BioStereometrics*, Vol. 602, 1985.

Li, S.Z. "Toward 3D Vision from Range Images: An Optimization Framework and Parallel Networks." Pages 231-260, *Computer Vision Graphics & Image Processing: Image Understanding*, Vol. 55, No. 3, May 1992.

McKerrow, Phillip John. *Introduction to Robotics*. Addison-Wesley, Reading, Massachusetts: 1993.

Roberts, Eric S. *The Art and Science of C*. Addison-Wesley, Reading, Massachusetts: 1995.

Thomas, Jr., George B., Finney, Ross L. *Calculus & Analytic Geometry*. Addison-Wesley, Reading, Massachusetts: 1992.