USING CHEBYSHEV POLYNOMIALS FOR INTERPRETING STRUCTURED LIGHT IMAGES

Michael O. Shneier, Wallace S. Rutkowski, and Tsai-Hong Hong
Industrial Systems Division
National Bureau of Standards
Gaithersburg, MD 20899

Abstract

A method is described of representing curved or line-like segments in images using Chebyshev polynomials. The advantages of the approximations include fast computation, minimum error over the sampling interval, and the reduction of the data to a concise description consisting of a few coefficients. An implementation is discussed that uses images obtained from a structured-light ranging system.

1. Introduction

Structured light sensors are often used in robot vision systems to provide fast, but sparse, range information. They operate by projecting a pattern of light onto a scene, and analyzing the image that results when the scene is viewed from a camera at a known position relative to the light source. Distance to illuminated points can then be computed using triangulation.

In practice, the complexity of the projected pattern is a large factor in determining the response time of a structured light system. At the National Bureau of Standards, we have chosen to project a parallel pair of planes of light, enabling us to determine the range and orientation of a surface (Albus *et al.*, 1982). More complex patterns require determination of the correspondence between the projected and imaged illumination (LeMoigne and Waxman, 1984), and for our purposes do not provide sufficient extra information to justify their cost. Simpler patterns, such as single planes or spots of light do not provide orientation information directly.

The applications for which the ranging system was developed require the ability to servo the robot in real time in such a way that it approaches an object at a known orientation relative to some surface on the object. To accomplish this, a camera and light source projector are mounted on the wrist of the robot, making the range and orientation measurements directly available for servoing. Our aim is to provide information rapidly enough (i.e., at frame rate) to be able to acquire randomly moving objects. This puts a heavy load on conventional algorithms for image analysis, even when the images are very simple.

To be able to extract the range and surface orientation information, it is necessary to identify the various lines in the image with the projected planes, and to pair up lines belonging to the same surface. This, together with problems due to noise, requires some preprocessing, which must be optimized if real-time response is to be attained.

Until recently, we had been performing generic operations on all images, including thresholding, noise-cleaning, and connected-components analysis. For floodlit images, this is justified, but for structured light images, most of the computations are wasted, and, in fact, further processing is necessary to identify the elongated components and to find their skeletons. While some of the processing can be performed by specialized hardware, the remaining processing is still too time-consuming. As a result, we explored alternate processing strategies, and implemented an algorithm that describes raw data consisting of curve segments in terms of Chebyshev polynomials.

2. Chebyshev Polynomials

The typical objects in a structured-light image are (thick) line and curve segments. These segments are noisy, and are sometimes connected to other segments which result from the illumination of different surfaces. (For example, a highlight on a surface may intersect several curve segments.) The information we wish to extract from the image includes the positions, extents, and shapes of all elongated components.

The positions of the curves provide range information, while the shape and extent information can help in recognizing the illuminated surfaces.

Because of the thickness of the lines (and the physical properties of the light projector), we represent each line by its skeleton (i.e., the set of midpoints oriented along the line). Due to noise, the points on the skeleton do not necessarily form a smooth curve. Rather than store and process all the individual points, we would like to find an expression that describes the shape of the curve. At the same time, we would like to minimize the effects of noise. This leads us to a least-squares technique. We use polynomial approximations to the curves because they are usually adequate to describe the segments that occur in practical cases. Chebyshev polynomials have a number of useful properties that make them ideal for this application (Ralston and Rabinowitz, (1978)).

First, the Chebyshev polynomials have the smallest maximum error in a neighborhood of any polynomial approximation of the same degree. Second, their behavior at the endpoints of intervals is better than that of other approximations. (This is important because the endpoints correspond to surface discontinuities which are important events.) Third, Chebyshev polynomials are orthogonal, which implies stability in the computations, and finally, they can be efficiently and incrementally computed ('on the fly').

Suppose we are given a set $\{x_i\}, i=0,1,\ldots,n$ of data points at which we have measured the values of a function $f(x)$ (in our case, the $(x_i, f(x_i))$ are the coordinates of the midpoints of the curves). We want to find an approximation to the curve of the form $a_0 p_0(x) + a_1 p_1(x) + \cdots + a_n p_n(x)$, for some n, where the $p_i(x)$ are Chebyshev polynomials. This will give us an approximation with the minimum error in the least-squares sense. To achieve this, we have two problems to solve. We have to find a set of polynomials, and then compute the coefficients that give the best fit to the data. These problems can be solved separately, because the orthogonal polynomials are functions only of the number of data points, and not of their corresponding values. We use the formulation given in Abromowitz and Stegun (1967).

We assume a set of evenly spaced sample points, which have been transformed to the range [0,N]. (This is not essential, see Ralston and Rabinowitz, (1978)). Let $x_m = m$ $(m=0,1,\ldots,N)$ be a set of points at which sample values are available. We can generate the set of orthogonal polynomials $p_0(x), p_1(x), \ldots$, using the following recurrence relations (Abromowitz and Stegun, 1967)

$$p_0(x)=1$$

$$p_1(x)=1-\frac{2x}{N}$$

$$(n+1)(N-n)p_{n+2}(x)=(2n+1)(N-2x)p_n(x)-n(N+n+1)p_{n-1}(x)$$

We note that the $p_i(x)$ can be precomputed for a range of N, and referenced at run-time. Alternatively, if the maximum degree (n) of the polynomial is prespecified, the orthogonal polynomials can be directly coded into the application program.

Having found the orthogonal polynomials, we now want to approximate the data by an expression $a_0 p_0(x) + a_1 p_1(x) + \cdots + a_n p_n(x)$ for some (usually small) n. (We also, in practice, want to gather terms in the resulting polynomial to give a result of the form $b_0 + b_1 x + \cdots + b_n x^n$).

We define

$$(p_n, p_m) = \sum_{x=0}^{N} p_n(x) p_m(x)$$

then

$$(p_n, p_n) = \sum_{x=0}^{N} p_n^2(x)$$

The (p_n, p_n) do not depend on the data values, but only on N, the number of samples, and they can thus also be precomputed and stored for use at run time. The (p_n, p_n) are given by the expression

$$(p_n, p_n) = \frac{(N+n+1)!(N-n)!}{(2n+1)(N!)^2}$$

The coefficients a_n are defined by

$$a_n = \frac{(f, p_n)}{(p_n, p_n)} = \frac{\sum_{x=0}^{N} p_n(x) f(x)}{(p_n, p_n)}$$

in which the only values that have to be computed from the data are the numerators of the expressions. This can clearly be done as the data are acquired, giving rise to a very efficient algorithm. An implementation of this technique is described in the next section.

3. Implementation

The implementation makes use of a limited number of terms in the approximations. For the typical data in our applications, quadratic approximation has proved sufficient. A number of image-processing problems have to be solved before the algorithm can be applied. The most important of these are the segmentation of the data into smooth curves and straight lines, and the selection of skeleton points. In principle, it would be possible to perform these operations in conjunction with the computation of the approximations, but this was not done in our implementation.

The line segments in the images are usually several pixels thick, and may be corrupted by noise (Figure 1). The segmentation process works on data that has been run-length coded by columns (bottom-to-top in the image). It requires that the runs in successive columns overlap their predecessors if they are to be considered part of the same curve. This is not a sufficient condition, however. Sometimes a curve will split into two, or two curves will merge. These cases are treated at the segmentation stage by forcing curves to end at junctions, and starting new curves for each branch. At a later stage in processing, smoothness criteria and domain knowledge can be used to selectively merge pieces of curves. Figure 2 shows an example of this situation.

A third possibility is that there might be a sharp change of curvature along a segment, for example, when two adjacent surfaces are illuminated. This case must be distinguished from the smooth curvature of, for example, a cylindrical surface. We use a curvature operator that examines a neighborhood in front of and behind each point to make a decision about whether or not to split the segment. The decision can be affected by high-level knowledge of the expected appearance of the image. Figure 3 shows examples of this process.

The output of the algorithm is a set of coefficients for each curve, and an index of the start and end columns in which the curve was found. An error term is also computed as an estimate of how well the data are approximated. Figure 4 shows some examples.

Our first implementation of this procedure splits the computations over two 16-bit processors. The first of these receives the run-length encoded data and performs the segmentation and skeleton. Its results are passed to a second processor, which takes each segment and computes its Chebyshev approximation. The reasons for splitting up the processing are partly to fit in with the general structure of our sensory system, and partly to separate those parts of the system that need a floating-point co-processor from those that do not. Including the time needed to take the pictures, the average cycle time of the algorithm is about 1/10th of a second, depending on the complexity of the image. The majority of the time is spent in the segmentation of the image, and we are confident that we could improve the time to frame rates by a combination of careful coding and sparser sampling. The Chebyshev approximations do not degrade significantly even when large numbers of the sample points are ignored. These improvements, together with the pipeline speedup possible using the two processors is expected to attain the necessary speed of processing.

4. Discussion

The approximation method is obviously not applicable only to structured-light images. Any curves or lines can be approximated in a similar fashion, although closed contours are more suited to approximations based on Fourier coefficients because of their cyclic nature.

The approximations are not without their disadvantages. One problem is that it is hard to decide, based on the coefficients, whether or not to merge two approximations, and the merging itself is not straightforward. For example, given a segment near the minimum of a parabola, and one further up one of the legs, it is not clear from the coefficients of their approximations that they belong to the same curve (one approximation will have a much larger x coefficient than the other). This is true even when the segments are close together. Even after it has been decided to merge the segments, the problem remains of how to compute a new, unified, approximation given only the coefficients of the separate approximations.

One approach is to decide when to merge by comparing the curvatures and intersection angles of opposing ends of segments, and then instantiating a small number of points on each curve, and computing a new approximation. This depends on intervals that are not evenly spaced, so it requires a slightly more complicated computation than that presented here, but it would probably not need to be done at frame rate.

The higher levels of our robot sensory system, into which the approximations are fed, deal with this problem in a different way. There are two main kinds of information we wish to extract from the structured light images. The first of these is range information, which can be found without merging curves. The second is information about the shapes and extents of surfaces, which requires at the least that the approximations be labeled as to which surface they fall upon.

Fortunately, we have other information about the world at our command. Of particular interest are predictions about what should appear in the image, derived from CAD models of expected objects and previous interactions with the world (Shneier *et al.*, 1984, Lumia, 1984). Given a prediction about a surface, it is possible to identify those image segments closest to that surface, and to compute an updated pose estimate for the corresponding object using the new image approximations. This is done in our system using a least-squares technique, described in Rutkowski *et al.*, (1984).

An avenue that we have not yet explored is the use of the prediction mechanism to generate expected coefficients for a surface fit. This would allow all curves on a surface to contribute to the same approximation, and would greatly simplify matching and recognition operations.

While the information available from the structured-light images includes the three-dimensional positions of each illuminated point, the approximations are computed in the image plane. This is because it is necessary to describe the curves before they can be associated with a particular plane of light and thus transformed to three dimensions, and because it is expensive to convert every point to three dimensions when only a few samples may be required. Also, the noise cleaning and smoothing performed on the two-dimensional data in the process of constructing the approximations greatly improve the accuracy of the extracted three-dimensional points.

Another useful property of the approximations is that they give a measure of the curvature of the associated segments (and thus of the underlying surface). This is useful for a preliminary sorting of the data before attempting to match it with the object models.

5. Conclusions

This paper has presented a method of describing curvilinear components of images using Chebyshev polynomials. In particular, it discussed an implementation that uses structured light images for supplying a robot manipulator with range and orientation information for servoing.

The advantages of the technique are that the approximations very concisely (and accurately) describe curved or straight segments, they smooth the data and deal with errors in an optimal way, and they have the potential for frame-rate processing speeds.

References

1. M. Abromowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* U. S. Government Printing Office, Washington D. C., 1967 (6th edition).

2. J. S. Albus, E. Kent, M. Nashman, P. Mansbach, L. Palombo, and M. Shneier, A 6-D vision system. Proc. SPIE Technical Symposium, Crystal City, Virginia, May 1982.

3. J. LeMoigne and A. M. Waxman, Projected light grids for short range navigation of autonomous robots. Proc. Seventh International Conference on Pattern Recognition, Montreal, Canada, July 1984.

4. R. Lumia, Representing solids for a real-time robot sensory system. To appear in Proc. Prolamat 1985, Paris, June 1985.

5. A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis.* McGraw-Hill, New York, 1978 (2nd edition).

6. W. S. Rutkowski, R. Benton, and E. W. Kent, Model-driven determination of object pose for a visually servoed robot. National Bureau of Standards Technical Report, 1984.

7. M. O. Shneier, R. Lumia, and E. W. Kent, Model-based strategies for high-level robot vision. National Bureau of Standards Technical Report, 1984, (to appear in Computer Vision, Graphics, and Image-Processing).

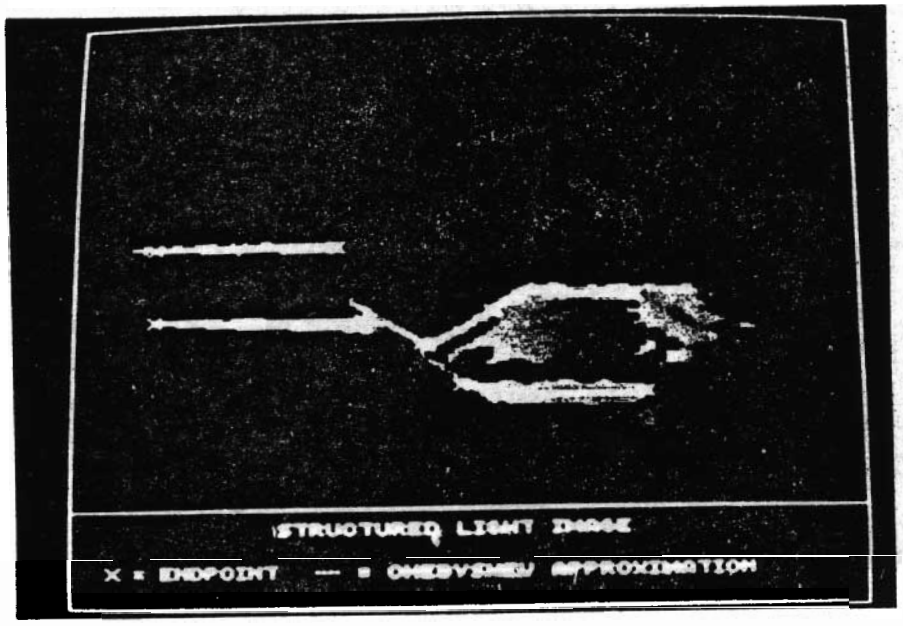


Figure 1. An image showing the kinds of noise that may be present in structured-light images. The dotted lines are the Chebyshev approximations.

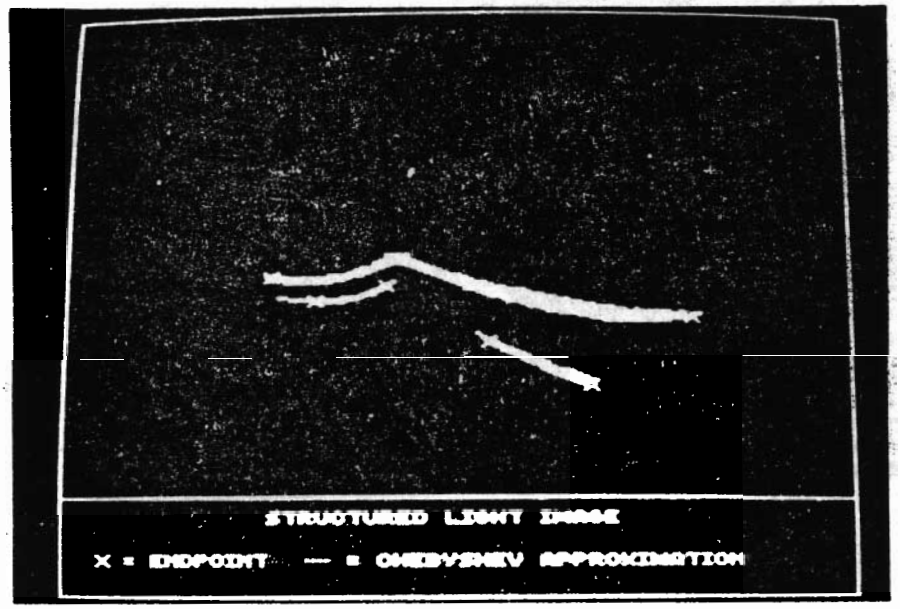


Figure 2. An example of the kinds of branching possible and the way in which the segmentation breaks up the components. The crosses show the endpoints of the Chebyshev approximations.

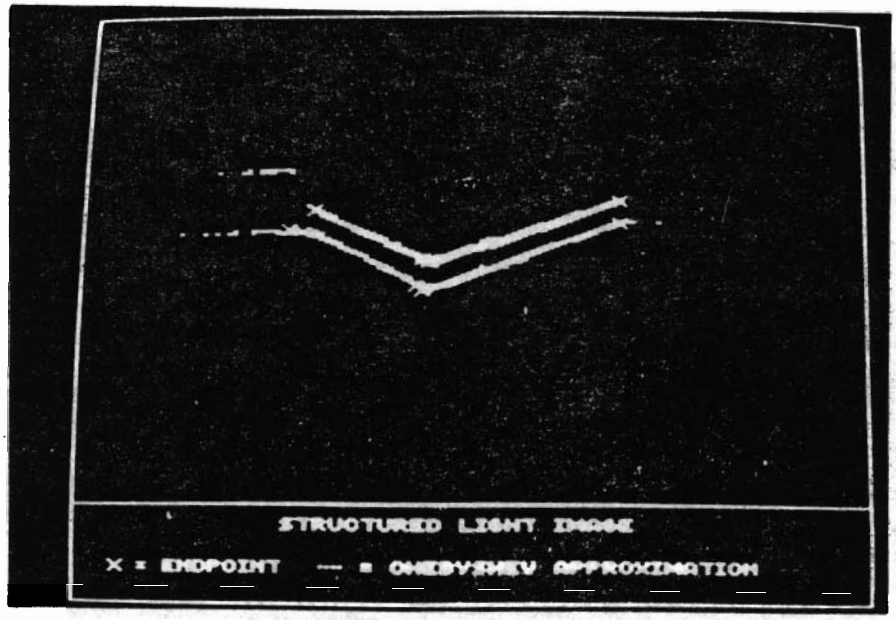
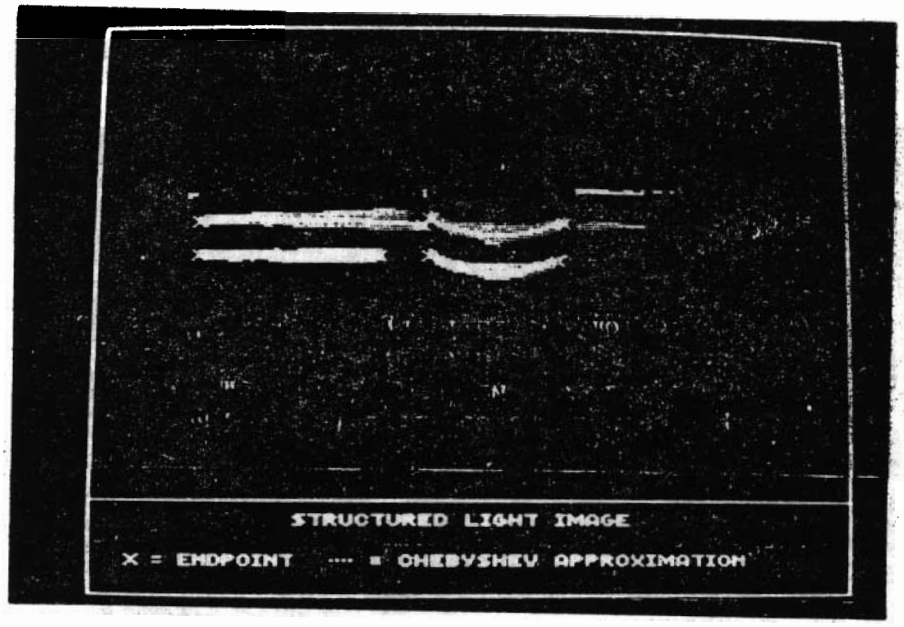


Figure 3. An example of how segments are split at junctions. The crosses show the endpoints of the Chebyshev approximations.



$0.0025x^2 - 0.0595x + 165.08$ error = 0.20 pixels
$-0.0049x^2 + 0.1465x + 181.86$ error = 0.46 pixels
$0.0617x^2 - 1.3798x + 164.80$ error = 0.27 pixels
$0.0561x^2 - 1.2880x + 183.20$ error = 0.40 pixels

Figure 4. Some examples of the approximations produced for a series of segments.