3-D Object Model Synthesis in a Monocular Imaging System

A. Mark Earnshaw
Andrew K.C. Wong

Pattern Analysis & Machine Intelligence Lab
Department of Systems Design Engineering
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

Abstract

One common method of 3-D computer vision is stereopsis. Presented here is a single camera system which has been implemented using this approach. The *MAP* (Matching Algorithm for Points) program is able to synthesize representative models of unknown polyhedral objects (*i.e.* straight edges and planar faces) of simple to medium complexity. Edge data is first extracted from digitized images and then passed through a formatter to remove background noise. Corresponding points are matched between successive images and the resulting information is used to triangulate the (x, y, z) coordinates of the original object vertices. This then permits the synthesis of a wire-frame model depicting the item under consideration. Sample results are presented to demonstrate the capabilities of the system. The directions for future development are also briefly discussed.

Keywords: Stereopsis, Edge Detection, Point Matching, Triangulation, Model Synthesis

1 Introduction

A common method for equipping computers with the ability to obtain visual data in three dimensions is the technique of stereopsis. This approach uses the offsets between corresponding points in paired images captured from different locations to triangulate the original locations of observed features. If the camera positions are known, the ranging may be performed relatively easily and accurately.

This form of data acquisition has many potential applications, both in academic research and the industrial world. Coupled with an object recognition program, such a system would be capable of identifying different items in its environment. Other examples might include tasks such as automated parts inspection.

The PAMI Lab at the University of Waterloo has been studying the field of 3-D computer vision for some time. Previous work has concentrated on simple constellation matching between stereo images [7, 14] and on the recognition of predefined objects in real-world scenes [12]. Based on these investigations, emphasis is now being placed on the ability to generate object models from visual data without the need for any preacquired knowledge about the items.

Current research concentrating on the area of stereoscopic vision has resulted in a system, entitled *MAP* (Matching Algorithm for Points), capable of synthesizing models of medium complexity with reasonable accuracy. The immediate application of the program is the ongoing development of an autonomous workcell by providing an Intelledex 705S robot with a visual interface.

2 Background & Motivation

Over the years, a number of vision programs using stereopsis have been developed and implemented. Typically, the primary concerns appear to be the identification of corresponding features between paired images. Once these have been determined, the triangulation process to recover three-dimensional profiles is relatively simple. For example, Alvertos *et al* [1] performed detailed analyses for different possible stereo camera configurations. The features selected for matching tend to vary, but are generally individual pixels, lines, or line intersection points. The latter two represent higher level features and usually reduce the necessary processing time significantly due to their smaller numbers.

If a camera system has been accurately calibrated, it is possible to pair individual pixels between images to recover a range map. Kim and Aggarwal [8] matched edge points using local gradient patterns as a guide. A three-view approach implemented by Ito and Ishii [6] paired points between the traditional two views, but also used a third image to verify all hypothesized matches. Both Hoff and Ahuja [4] and Olsen [11] integrated the processes of feature matching and surface interpolation by using a surface smoothness constraint iteratively to aid in identifying correct and incorrect pairs.

Lines are a higher level feature which may also be used to recover three-dimensional structure. McIntosh and Mutch [10] presented one algorithm for accomplishing this task, although their range calculation method was not accurate for partially occluded lines. A graph theory approach was used by Horaud and Skordas [5] to identify the largest clique in a correspondence graph which represented all feasible line mappings. Even curved edge contours can be utilized as was illustrated by Sherman and Peleg [13] who used such attributes as shape similarity, equal contrasts, and relative contour ordering to facilitate the correspondence process.

The intersection points of linear edges represent yet another feature that may be matched for stereo purposes. Jenness [7] began some preliminary research into this area using constellation arrangements as an additional guiding attribute. However, the resulting system was not capable of constructing a complete object model.

There are a few problems with the existing approaches to stereo vision. Triangulation error is one primary area of concern. It is usually caused by the image discretization and/or imprecise feature localization. Ideally, the baseline between camera positions should be as large as possible to minimize this error. Further improvements result if the cameras are placed at right angles to each other [3]. However, this often causes the two views to appear radically different, thus complicating the feature correspondence process. In addition, only a portion of the object being studied will be visible from any single location. A number of features will be occluded and therefore only a partial reconstruction of the item may be performed.

A possible solution to both of these problems is the integration of information from multiple images. For instance, one approach would be to rotate a camera around an object, capturing images at successive 10° increments (this value was experimentally found to yield data whose similarity facilitated the feature matching process). Corresponding points may then be tracked through the image sequences. This provides more information for the triangulation procedure and would thus be expected to improve measurement accuracy. In addition, a full revolution of the object implies that all of its features should be visible in at least several of the images (assuming a relatively convex item) so the occlusion problem is also addressed.

3 System Overview

There are five major components in the *MAP* system as illustrated in **Figure 1**. Data enters the system through a CCD video camera. The resulting digitized images are then passed through an edge detector which extracts line features and identifies corner points by searching for intersecting lines. An optional image formatter separates important features from the raw edge data by removing

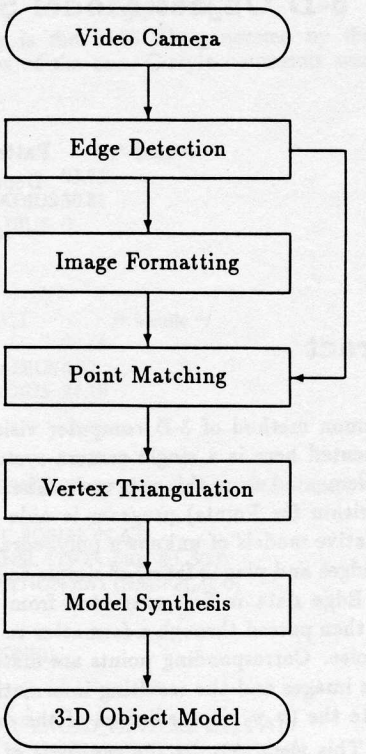


Figure 1: Flowchart of System Components

any identifiable background noise. Matching points are then tracked through successive images so that they can be correlated for triangulation purposes, this being the next stage of the program. Finally, the individual images are integrated to synthesize a model of the original object.

It should also be mentioned that the current system is programmable using a simple interpreted language. This allows the separate modules to be traversed in an order deemed to be most efficient for a specific application. For instance, it may be economical to format and match a given image while the camera is being moved to its next location. As a result, significant flexibility is attained without the requirement of extensive program reconfiguration.

4 Algorithm Outline

4.1 Edge Detection

There are four basic stages to the edge detection process. Gradient values representing possible edges are first extracted from a digitized image. These are linked together to form lines which are then extended to identify corner points. Finally, an optional routine recovers the camera's position and orientation from a visual reference pattern.

The direction of a gradient is defined as being perpendicular to the edge to which it belongs. Many edge detection schemes calculate gradients for several different directions, but the *MAP* system only computes horizontal and vertical values as follows:

$$H_{x,y} = [p_{x+2,y} + 2p_{x+1,y}] - [2p_{x-1,y} + p_{x-2,y}]$$
$$V_{x,y} = [p_{x,y+2} + 2p_{x,y+1}] - [2p_{x,y-1} + p_{x,y-2}]$$

where $H_{x,y}$ is the horizontal gradient at (x, y), $V_{x,y}$ is the corresponding vertical gradient, and $p_{i,j}$ is the gray level intensity at the specified coordinates. The 5×1 weighted mask was selected to identify edges primarily from the two adjacent pixels, although the outlying values are also used to smooth the gradient and reduce the effects of noise. To position the edge point, successive gradients that are higher than a specified magnitude are searched for local maxima or minima. These typically represent high contrast pixels and the best locations for situating the edge crossing.

It is now necessary to group collinear points together to form lines. A nucleus is first identified by locating two adjacent gradient values and extrapolating to predict the position of a third point. Once such a line has been successfully initialized, it is possible to track the entire edge by continuing to project the line at both ends and searching for gradient points in the vicinity of the predicted locations.

Slopes and intercepts are determined using standard linear regression formulas which minimize the sum of squared error terms. Since it is only necessary to maintain a few summation values for each edge, it is relatively simple to add new points by modifying the appropriate tabulations and then updating the line parameters. This capability is important for the edge tracking process which functions by incrementally adding points to an existing line.

The identified lines are then extended at each end to produce corner points. By first fitting the edges to the observed data and then locating the intersections, it is possible to position these points with subpixel accuracy. The task is accomplished by considering each line in turn and searching in the local vicinity of its two ends for any other nearby lines which may form potential intersections. Since multiple edges may not intersect exactly, the actual point coordinates are calculated to minimize the sum of squared perpendicular distances from each line. Before finalizing a corner point, various tests are made such as checking if two lines are collinear and should therefore be combined, or if the intersection falls in the middle of one line which should then be split in half.

It is also essential to know the camera's position and orientation for each image. This may be accomplished through the use of a visual reference pattern (as shown in **Figure 2**) which is included in the image. Since the pattern's dimensions and construction are known, it is not difficult to use its observed shape to derive a 3-D transformation (consisting of a rotation and a translation) giving the position of

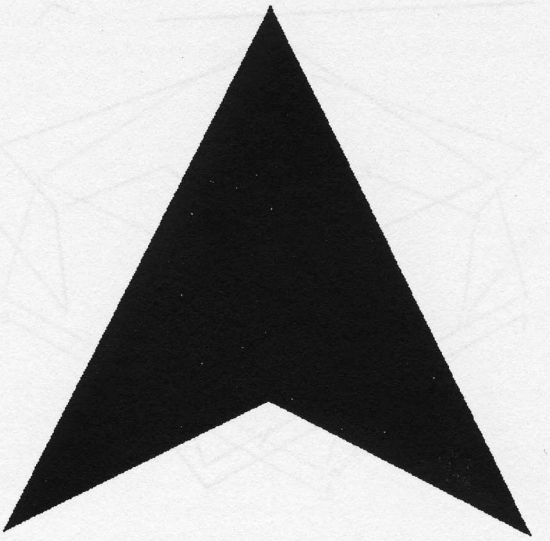


Figure 2: Visual Reference Pattern

the symbol relative to the camera [3, 9]. This may then be inverted to obtain the camera position and orientation in a common global coordinate system, assuming that the object and pattern remain fixed with respect to each other.

4.2 Image Formatting

Depending upon the quality of the edge data, it may not always be in a form immediately useful for matching purposes. As a result, it must sometimes be modified and formatted prior to being passed on to the third component of the system. Most of the background noise and other artifacts are removed in order to render a cleaner image. This has become less necessary though due to improvements in the edge detector. Formatting, if used, is accomplished by routines which perform functions such as filling in the missing sections of edges passing through low contrast areas, deleting lines which intersect the image boundaries, and removing isolated, unconnected edges.

A sample unformatted image consisting of the edge detector output may be found in **Figure 3**. The corresponding formatted image is shown in **Figure 4** and clearly represents a more useful form of information.

4.3 Point Matching

To recover the coordinates of the original object vertices, it is first necessary to match corresponding points between different views. Since successive images are normally captured at only 10° intervals, they are fairly similar and the problem becomes one of feature tracking which can be per-

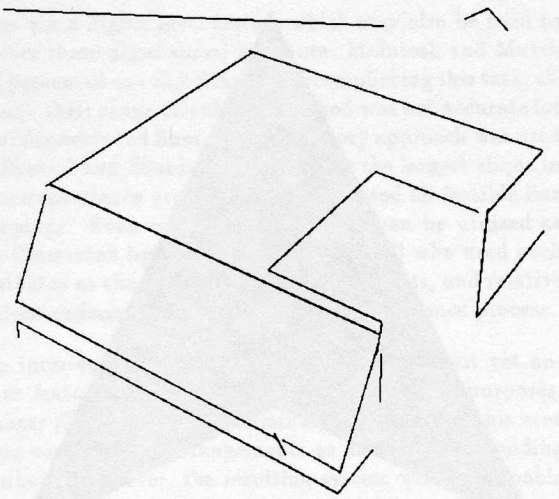


Figure 3: Sample Unformatted Image

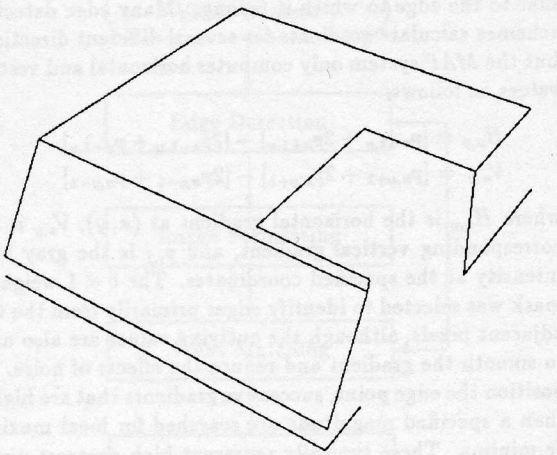


Figure 4: Sample Formatted Image

formed quickly and accurately. In addition, the use of a monocular system implies that only 36 images need to be matched over a full revolution. Therefore, the processing time is much less than for a stereo two-camera system performing the same task. Fewer images also reduce the probability of matching errors that might affect the final model.

The entire matching process is centred on a cost function which evaluates a possible point mapping on the basis of such factors as (x, y) coordinates, connections to already matched points, and lengths, angles and numbers of attached lines. For each specified pair, a cost figure is returned, with lower values indicating better matches than higher ones. If a match is considered to be completely unsuitable, it is assigned a very large penalty cost to remove it from further consideration. Since this routine is accessed thousands of times during the synthesis of a single model, it has been highly optimized through the use of techniques such as look-up tables for frequently calculated values.

When a mapping of image features is performed, the points are represented as vertices in a bipartite graph. Individual matchings are specified using the connecting arcs which are weighted with the appropriate cost values. A feasible image mapping is therefore composed of a set of graph edges such that no two are incident on the same vertex (*i.e.* a given point in one image is matched to no more than one point in the other image). If the graph were complete, a significant amount of processing time would be required to obtain an optimal image mapping. However, it is possible to apply certain heuristics to the problem which noticeably improve the throughput of the system.

The actual algorithm may be described as an iterative technique where a valid image mapping is obtained during each pass through a loop until a stable configuration is achieved. Each iteration consists of a matching cycle which finds the currently optimal set of feature correspondences and a correction cycle which compensates for any errors made during the former process. A maximum number of iterations is imposed to prevent infinite cycling (*i.e.* flipping between two different image mappings), although this has not yet manifested itself as a problem during testing.

During the matching cycle, the cost of pairing each point in the first image with each point in the second is nominally evaluated. Assuming about n points per image (n is typically between 15 and 20), this would be an $O(n^2)$ process and therefore computationally expensive. However, due to image similarities, potential matching candidates can be constrained according to vertical and horizontal position, and only a few possibilities thus result for each point. All of the valid potential matches are sorted into ascending order by cost value, thereby implying that the more suitable mappings will be placed near the start. These pairings are removed from the list on an individual basis and added to the current set of matchings. For each new addition to this group, any remaining matches involving either of the two points in question are dropped from further consideration. Upon the completion of this process, a set of possible feature mappings between the two images is obtained.

Following the generation of each preliminary image mapping, the correction cycle is invoked in order to locate and eliminate any errors. The set of matches is evaluated as a whole rather than on an individual basis to identify any mutual inconsistencies. This is primarily accomplished by examining the relative positioning and connectivity of paired points between the two images. The cost of each matching is recomputed, taking into consideration any information known about the local neighbours, and any infeasible correspondences are removed. The program then attempts to complete the total set of matches by searching

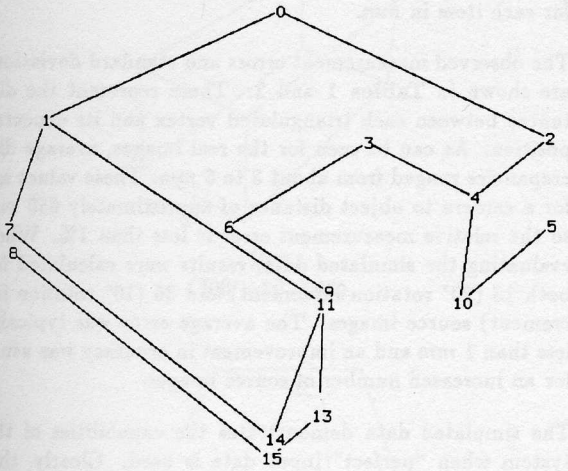


Figure 5: Sample Point Matching – Image 1

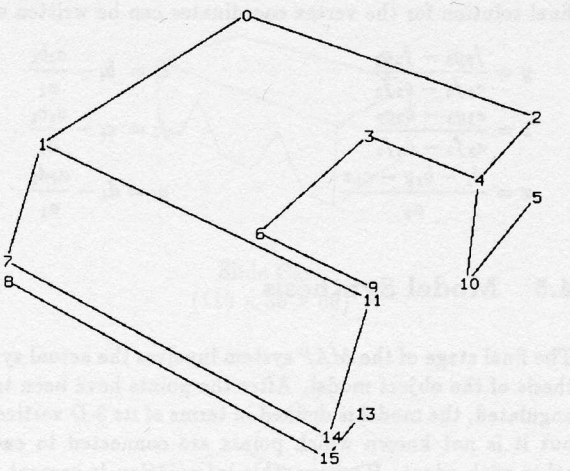


Figure 6: Sample Point Matching – Image 2

for valid pairings between the remaining leftover points.

A sample point matching as produced by the *MAP* system may be seen in **Figures 5** and **6**.

4.4 Vertex Triangulation

If the camera position and orientation are known for a given image and a pinhole camera model is assumed, an object point can be constrained to lie on a ray extending from the observed position on the CCD array and through the centre of the camera lens. If multiple rays can be determined for the same point (as matched between successive images), it is theoretically possible to compute their intersection point which will represent the original object vertex.

In reality, it is desirable to have at least several rays for a given object point to compensate for the inaccuracies of the image digitization and edge detection process. In addition, it is improbable that all of these lines will intersect at a common location. Thus, it is necessary to determine the best "point of intersection" as evaluated according to some error function. The *MAP* system uses the sum of squared perpendicular distances between the point and the lines and positions the object vertex to minimize this value.

For ease of calculation, rays are expressed in a parametric form such as:

$$(x_i, y_i, z_i) + k_i(u_i, v_i, w_i)$$

where (x_i, y_i, z_i) are the coordinates of a point known to be on the i^{th} ray (usually the lens centre), k_i is an unknown constant, and (u_i, v_i, w_i) is a normalized direction vector. An actual point on the line may be specified by calculating the appropriate value of k_i. If the x and y axes define the plane of the CCD array and the z axis represents the camera axis, then (u_i, v_i, w_i) may be expressed as the normalized form of (x_p, y_p, f) where (x_p, y_p) give the point's location on the image plane and f is the focal length. Note that this vector is represented in the camera's coordinate system and must therefore be translated into the global system by utilizing the known transformation.

Using the above definitions, the selected error function may be written as:

$$D = \sum_{i=1}^{n} \Big\{ [x - (x_i + k_i u_i)]^2 + [y - (y_i + k_i v_i)]^2 + [z - (z_i + k_i w_i)]^2 \Big\}$$

where (x, y, z) represents the unknown intersection point of the n lines and $(x_i + k_i u_i, y_i + k_i v_i, z_i + k_i w_i)$ gives the coordinates of the closest point on the i^{th} ray to this location. After setting the partial derivatives of D with respect to x, y and z equal to zero and performing some manipulation, the following system of equations may be obtained.

$$\left(n - \sum_{i=1}^{n} u_i^2\right) x + \left(-\sum_{i=1}^{n} u_i v_i\right) y + \left(-\sum_{i=1}^{n} u_i w_i\right) z = \sum_{i=1}^{n} \left[(1 - u_i^2) x_i - u_i v_i y_i - u_i w_i z_i\right]$$

$$\left(-\sum_{i=1}^{n} v_i u_i\right) x + \left(n - \sum_{i=1}^{n} v_i^2\right) y + \left(-\sum_{i=1}^{n} v_i w_i\right) z = \sum_{i=1}^{n} \left[-v_i u_i x_i + (1 - v_i^2) y_i - v_i w_i z_i\right]$$

$$\left(-\sum_{i=1}^{n} w_i u_i\right) x + \left(-\sum_{i=1}^{n} w_i v_i\right) y + \left(n - \sum_{i=1}^{n} w_i^2\right) z = \sum_{i=1}^{n} \left[-w_i u_i x_i - w_i v_i y_i + (1 - w_i^2) z_i\right]$$

Assuming these to have the form $a_i x + b_i y + c_i z = d_i$, the final solution for the vertex coordinates can be written as:

$$y = \frac{f_2 g_3 - f_3 g_2}{e_3 f_2 - e_2 f_3} \qquad e_i = b_i - \frac{a_i b_1}{a_1}$$

$$z = \frac{e_2 g_3 - e_3 g_2}{e_2 f_3 - e_3 f_2} \qquad f_i = c_i - \frac{a_i c_1}{a_1}$$

$$x = \frac{d_1 - b_1 y - c_1 z}{a_1} \qquad g_i = d_i - \frac{a_i d_1}{a_1}$$

4.5 Model Synthesis

The final stage of the *MAP* system involves the actual synthesis of the object model. After the points have been triangulated, the model is defined in terms of its 3-D vertices, but it is not known which points are connected to each other with edges. However, this information is present in the original source images and the mapping from image points to model points is known. When these vertex links are copied to the model, a count is kept of the number of source images in which a given edge appears. If this total is lower than a specified threshold value, the corresponding edge is assumed to be due to noise and/or errors in the point matching process and is deleted.

Point matches involving sections of partially occluded lines are also identified and eliminated at this time. Typically, these points are in the form of a T with the crossbar representing the edge of the occluding face and the remaining stroke the portion of the line which is still visible. Thus, occluded features may be identified by examining each sequence of successively matched points. If a fraction of these are of degree 3 (*i.e.* have three lines attached) and the remainder are not, the former points probably represent views in which the line was partially occluded and are therefore removed from further consideration. In such an instance, the coordinates of the model vertex are also recalculated from the updated information.

5 Results

The vision system has been tested with both real and simulated data using objects such as those shown in **Figures 7** and **8**. The former demonstrated that the program could function in real-world situations, and the latter were used to evaluate the theoretical accuracy of the reconstruction process. In both cases, camera positions and orientations were derived from images of the visual reference pattern. The real images were obtained from a camera mounted on a robot arm and represented a 220° revolution of the item, whereas the simulated data covered a full 360° rotation. These latter images were generated by shading faces with a common intensity and then smoothing the data by replacing each pixel with its 3×3 local average. The maximum dimensions in the (x, y, z) directions are also given for each item in *mm*.

The observed measurement errors and standard deviations are shown in **Tables 1** and **2**. These represent the distances between each triangulated vertex and its expected position. As can be seen for the real images, average discrepancies ranged from about 3 to 6 *mm*. These values are for a camera to object distance of approximately 650 *mm* so the relative measurement error is less than 1%. When evaluating the simulated data, results were calculated for both 18 (20° rotation increment) and 36 (10° rotation increment) source images. The average error was typically less than 1 *mm* and an improvement in accuracy was usual for an increased number of source images.

The simulated data demonstrates the capabilities of the system when "perfect" input data is used. Clearly, this will usually not be the case as illustrated by the larger measurement errors for real-world data. It is likely that these discrepancies can be attributed to such factors as slight errors in the camera positioning, imprecise edge localization, and physical camera attributes. In particular, no calibration procedure to measure the intrinsic camera parameters such as exact focal length and lens distortion has been integrated into the program as of yet.

Sample CPU execution times for the simulated images are shown in **Table 3**. These were obtained by averaging the

Object	# Points	$\bar{x}$	s
Box	8	3.84	2.25
Truncated Tetrahedron	6	4.00	1.41
Widget	12	3.04	2.03
Robot Cover Plate	12	6.27	2.54

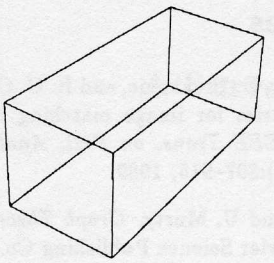
Table 1: Measurement Errors for Real Data (*mm*)

	18 Images		36 Images	
Object	$\bar{x}$	s	$\bar{x}$	s
Slide Case	0.111	0.058	0.091	0.047
Pyramid Box	0.159	0.060	0.215	0.115
Monitor	1.127	1.237	0.643	0.705
Rubik's Cube	0.184	0.096	0.115	0.065

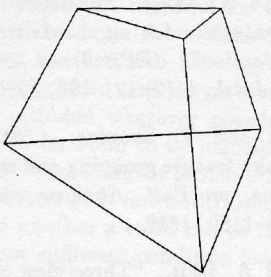
Table 2: Measurement Errors for Simulated Data (*mm*)

	Execution Times		
Object	Edge	Synth (18)	Synth (36)
Slide Case	1.41	0.89	1.64
Pyramid Box	1.42	0.80	1.54
Monitor	1.45	1.16	2.23
Rubik's Cube	1.65	5.11	8.28

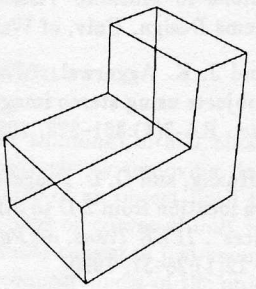
Table 3: CPU Execution Times for Simulated Data (s)



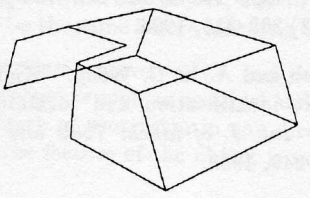
Box
(200 × 60 × 90)



Truncated Tetrahedron
(240 × 100 × 210)

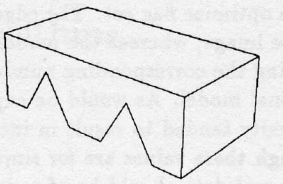


Widget
(100 × 100 × 150)

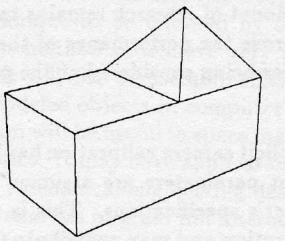


Robot Cover Plate
(320 × 70 × 240)

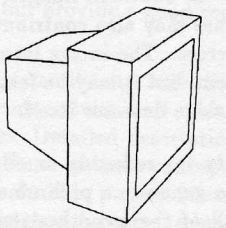
Figure 7: Real World Test Objects



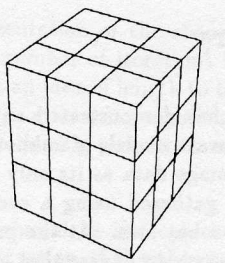
Slide Case
(110 × 30 × 60)



Pyramid Box
(300 × 175 × 150)



Monitor
(500 × 400 × 500)



Rubik's Cube
(120 × 120 × 120)

Figure 8: Simulated Test Objects

results for a large number of program iterations. All time values are for a Sun SPARC 330 executing *C* code compiled with the optimize flag set. The edge detection times are for a single image, whereas the model synthesis values are for matching the corresponding number of images and producing a final model. As would be expected, increased object complexity tended to result in increased execution times. Although these values are for simulated data, running times for real data should be of a similar magnitude.

6 Future Work

A significant amount of research remains to be carried out in order to improve the performance of the *MAP* system. Some possibilities being considered at the present time are discussed below.

Hitherto, no explicit camera calibration has been performed and the relevant parameters are assumed to conform to the manufacturer's specifications. This is not necessarily a realistic assumption and may contribute towards the observed measurement errors. Consequently, it would be useful to design or implement a method for carrying out some type of calibration to estimate these parameters.

The primary problem at present is the edge detector component. It is not always possible for the computer to fit accurate edges to the observed images. Incorrectly placed and/or missing lines may also contribute towards possible point matching errors. The image formatter partially addresses this problem, but it may be feasible to increase the robustness of the edge detector itself.

Another possibility for reducing or eliminating matching errors would be to generate a preliminary model and then use it to verify all of the hypothesized point matches by projecting it back onto the original images. Any identified errors would be corrected and the model revised to reflect these changes.

7 Summary

The *MAP* system has demonstrated an ability to generate reasonably accurate models of unknown polyhedral objects using video image data as its only input. These images are currently gathered using a camera mounted on an industrial-type robot arm. At the present time, some supervision is necessary to ensure that accurate and valid results are obtained. Work is now underway on improving the vision interface to develop a workcell environment where a robot can function autonomously without the need for operator intervention.

References

[1] N. Alvertos, D. Brzakovic, and R. C. Gonzalez. "Camera geometries for image matching in 3-D machine vision". *IEEE Trans. on Patt. Analysis and Mach. Intel.*, 11(9):897–915, 1989.

[2] J. Bondy and U. Murty. *Graph Theory with Applications.* Elsevier Science Publishing Co., Inc., 1976.

[3] A. M. Earnshaw. *3-D Object Synthesis in a Robotic Workcell.* Master's thesis, Dept. of Systems Design, Univ. of Waterloo, 1991.

[4] W. Hoff and N. Ahuja. "Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection". *IEEE Trans. on Patt. Analysis and Mach. Intel.*, 11(2):121–136, 1989.

[5] R. Horaud and T. Skordas. "Stereo correspondence through feature grouping and maximal cliques". *IEEE Trans. on Patt. Analysis and Mach. Intel.*, 11(11):1168–1180, 1989.

[6] M. Ito and A. Ishii. "Three-view stereo analysis". *IEEE Trans. on Patt. Analysis and Mach. Intel.*, PAMI-8(4):524–532, 1986.

[7] J. D. Jenness. *A Constellation Matching Algorithm with Applications to Machine Vision.* PhD thesis, Dept. of Systems Design, Univ. of Waterloo, 1987.

[8] Y. C. Kim and J. K. Aggarwal. "Positioning three-dimensional objects using stereo images". *IEEE J. of Rob. and Auto.*, RA-3(4):361–373, 1987.

[9] Y. Liu, T. S. Huang, and O. D. Faugeras. "Determination of camera location from 2-D to 3-D line and point correspondences". *IEEE Trans. on Patt. Analysis and Mach. Intel.*, 12(1):28–37, 1990.

[10] J. H. McIntosh and K. M. Mutch. "Matching straight lines". *Comp. Vision, Graphics, and Image Proc.*, 43:386–408, 1988.

[11] S. I. Olsen. "Stereo correspondence by surface reconstruction". *IEEE Trans. on Patt. Analysis and Mach. Intel.*, 12(3):309–315, 1990.

[12] K. D. Rueb and A. K. C. Wong. "Knowledge-based visual part identification and location in a robot workcell". *Int. J. of Mach. Tools and Manufacture*, 28(3):235–249, 1988.

[13] D. Sherman and S. Peleg. "Stereo by incremental matching of contours". *IEEE Trans. on Patt. Analysis and Mach. Intel.*, 12(11):1102–1106, 1990.

[14] A. K. C. Wong and R. Salay. "An algorithm for constellation matching". In *Proc. of the Eighth Int. Conf. on Patt. Rec.*, pages 546–554, 1986.