A Decision Tree Classifier for Object Recognition in Range Imagery

Michael Greenspan

Visual Information Technology
Institute for Information Technology
National Research Council of Canada

Department of Systems
and Computer Engineering
Carleton University

M50 Montreal Rd., Ottawa, Ontario, Canada, K1A 0R6
Michael.Greenspan@nrc.ca

Abstract

A method is presented for efficient and reliable object recognition within noisy, cluttered, and occluded range images. The method is based on efficiently matching a set of voxel templates generated by rotating a model exhaustively through its pose space. The template set is composed into a binary decision tree classifier, with the tree leaf nodes representing individual voxel templates of the model. The internal tree nodes represent the union of the templates of their descendant leaf nodes, and the union of all leaf node templates is the complete template set of the model oriented over its discrete pose space. Each internal node also references a single voxel which is common to its child node templates. Traversing the tree is equivalent to efficiently matching the large set of templates at a selected image location. Experiments have shown the process to be approximately 4 orders of magnitude more efficient than brute-force template matching. Results are presented in which recignition is achieved in less than 20 seconds within noisy and significantly occluded range images.

1 Introduction

Most object recognition methods involve a feature extraction preprocessing phase, followed by a matching phase which relates the extracted image features to those in a model database. The use of high level features effectively reduces the space of possibilities in the matching phase, and lends structure to the matching process. A reliance on feature extraction can, however, be problematic. The robustness of extracted features may depend upon extrinsic factors, such as lighting conditions in 2D intensity images, and sensor vantage in 3D range images [1]. Feature extraction can also be computationally expensive, and is often more so than the subsequent matching. Further, methods based on a particular feature set are restricted to the recognition of objects that contain some features from that set.

An alternative to feature extraction is template matching. An object to be recognized is represented as a template, which is a subimage described in low level image elements. Matching is performed by sliding the template over all possible image locations, and calculating a correlation metric.

In its straightforward implementation, template matching can be inefficient. For m template pixels and n image pixels, $m^2(n-m+1)^2$ operations are required to score the template at all image locations. Rosenfeld et al. [2, 3] reduce this expense by sequencing the order in which the elements are compared, thereby increasing the likelihood that an error threshold signifying a mismatch is exceeded prior to comparing all pairs. Other proposed methods to improve efficiency include multistage subtemplate matching [4], mutliresolution [5] and hierarchical structures [6, 7].

While the above cited methods focus on improving the efficiency of matching a single template against an image, other methods improve the efficiency of matching a set of templates. The best known of these is the Hough Transform [8] which is equivalent to matching a set of templates from a given class of objects [9]. Ramapriyan [10] organized a set of templates into a tree structure with leaf nodes corresponding to individual templates. At each intermediate node, the input template was

compared with a representative template comprised of the union of all descendant templates, and the value of the score determined which branch to follow. Experimental results on 2D images with a set of 36 templates showed an improvement of a factor of 4 over the brute-force method.

Our approach is based upon compiling a set of 3D templates, generated by rotating a model exhaustively through its pose space, into a binary decision tree classifier. A decision tree classifier is a pattern recognition structure which efficiently associates an input signal with one of a possibly large number of classes. In our case, the leaf nodes represent small sets of templates, and the intermediate nodes represent the union of all descendant sets. As the tree is traversed, a single template element, which is the intersection of all descendant node templates, is used to differentiate between the two children nodes. We have found this construct to be both reliable and many orders of magnitude more efficient than the brute-force approach.

The complete method is organized into 4 processing phases. In the first hypothesis generation phase, the tree is traversed for each (discrete) image point, and likely template matches (i.e. model poses) are identified at each location. The two subsequent verification phases rank the results first in the discrete and then the continuous domain. A final refinement phase improves the estimated pose of the match using a standard Iterative Closest Point (ICP) method [11]. A diagram of the main offline and online processing phases, and their associated data structures, is illustrated in Figure 1.

The remainder of this paper is organized as follows: in Section 2, 3D object recognition is defined as a constraint satisfaction problem, and the quantization of the problem domain is justified. Section 3 describes the hypothesis generation phase and addresses aspects of the decision tree construction and traversal. Section 4 describes the verification phases. Experimental results are presented in Section 5, and the paper concludes in Section 6 with a summary and a discussion of some future research issues.

2 Recognition as Constraint Satisfaction

Let a scene $\mathbf{S}$ consist of a set of surfaces, and let a range image $\mathbf{I}$ of $\mathbf{S}$ be defined as a set of n points $\{p_i\}_1^n$ which samples the surfaces in $\mathbf{S}$. The points are described in sensor frame coordinates, and in keeping with most range data acquisition technolo-

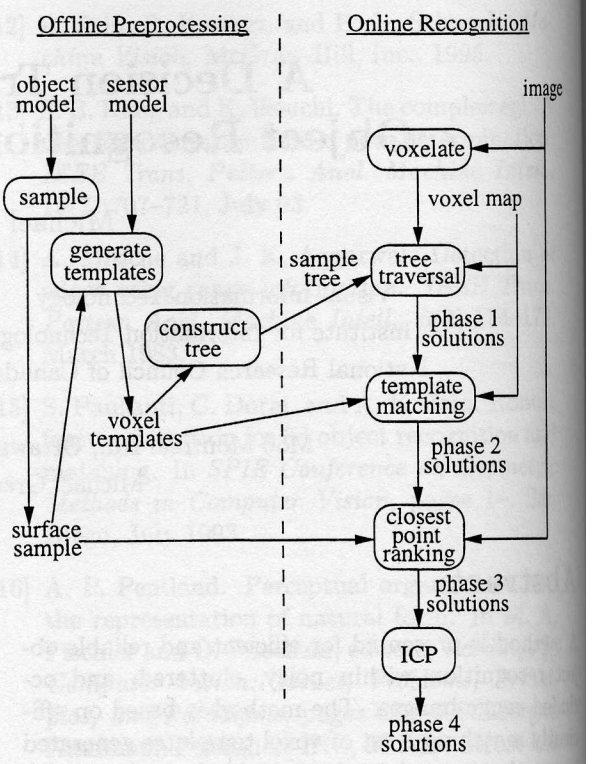

Figure 1: Process Diagram

gies, we assume that any $p_i \in \mathbf{I}$ correspond only to portions of the surfaces in $\mathbf{S}$ which are non-occluded with respect to the sensor vantage.

Let $\mathbf{M}$ be a surface model of a rigid object, which may be transformed homogeneously within $\mathbf{S}$ to any 6 dimensional pose $\vec{\theta}$. We denote an instance of $\mathbf{M}$ with pose $\vec{\theta}$ as $M|_{\vec{\theta}}$. The objective of object recognition is to determine all occurences of $M|_{\vec{\theta}}$ in the subspace of $\mathbf{S}$ that is imaged by I.

Let $\Gamma(M|_{\vec{\theta}}, \mathbf{S})$ be the set of surfaces defined by the projection of the non-occluded surfaces of $M|_{\vec{\theta}}$ onto the closest corresponding surfaces of $\mathbf{S}$. Thus, for every non-occluded point on the surface of the model in the given pose, there will exist a corresponding closest projected point in the scene;

$$p_m \in M|_{\vec{\theta}} \Longrightarrow \exists\, p_g \in \Gamma(M|_{\vec{\theta}}, \mathbf{S}) \quad (1)$$
$$\text{s.t. } p_g \in \mathbf{S} \text{ and } |p_m - p_g| \leq |p_m - p_s| \;\forall\, p_s \in \mathbf{S}$$

One way to measure the goodness of a match of $M|_{\vec{\theta}}$ in $\mathbf{S}$ is to consider the magnitude of the area of $\Gamma(M|_{\vec{\theta}}, \mathbf{S})$, and the magnitude of the volume bound by $M|_{\vec{\theta}}$ and $\Gamma(M|_{\vec{\theta}}, \mathbf{S})$. In the case of a perfect match, for example, the area of $\Gamma(M|_{\vec{\theta}}, \mathbf{S})$ would

equal that of $M|_{\vec{\theta}}$, and the volume between them would have a magnitude of zero.

Let $\mathcal{A}(\mathbf{S}_i)$ denote the magnitude of the area of a set of surfaces $\mathbf{S}_i$ and let $\mathcal{V}(\mathbf{S}_i, \mathbf{S}_j)$ denote the magnitude of the minimal volume bound by two surface sets. Object recognition can then be defined as a process of finding all values of $\vec{\theta}$ which satisfy the following constraints:

$$|\mathcal{A}(M|_{\vec{\theta}}) - \mathcal{A}(\Gamma(M|_{\vec{\theta}}, \mathbf{S}))| < \varepsilon_{\mathcal{A}} \quad (2)$$

$$\mathcal{V}(M|_{\vec{\theta}}, \Gamma(M|_{\vec{\theta}}, \mathbf{S})) < \varepsilon_{\mathcal{V}} \quad (3)$$

where $\varepsilon_{\mathcal{A}}$ and $\varepsilon_{\mathcal{V}}$ are thresholds determined by the model geometry, the accuracies of the sensor and model, and the noise and occlusion properties of the image and scene.

These constraints form a necessary but not sufficient condition for recognition. There may be arbitrary scene surfaces which satisfy the constraints, but are not part of the object of interest. There may also be other objects which are similar in shape to the object of interest, and satisfy the constraints in certain poses. Approximations of these constraints are the basis of ICP algorithms which are commonly used in pose refinement processes [11].

The exhaustive method of simply transforming the model through all values of $\vec{\theta}$ and estimating the constraint function values is not feasible, as the general 6 dimensional pose space is large. Rather, we must rely upon heuristic methods of searching this space.

2.1 Domain Quantization

Rather than operating in continuous space, the first two phases of the process map the problem into a coarse resolution discrete domain, with $\mathbf{I}$ represented as a voxel map, and each $M|_{\vec{\theta}}$ as a set of templates of the voxels that intersect with the surface of the model in the specified pose. The quantization of range images into voxels, and the related *octree* representation, is a technique that has been used to advantage in mesh generation [12, 13], image registration [14], gaze planning [15], and recently object recognition [16]. While many recognition methods quantize some mapping of the image signal [17, 18], to the author's knowledge, the only known method in which voxel template matching has been used directly is [19].

Each voxel in the range image voxel map takes on one of four values :

- **surface (S)** : the voxel intersects with the surface of the model ;

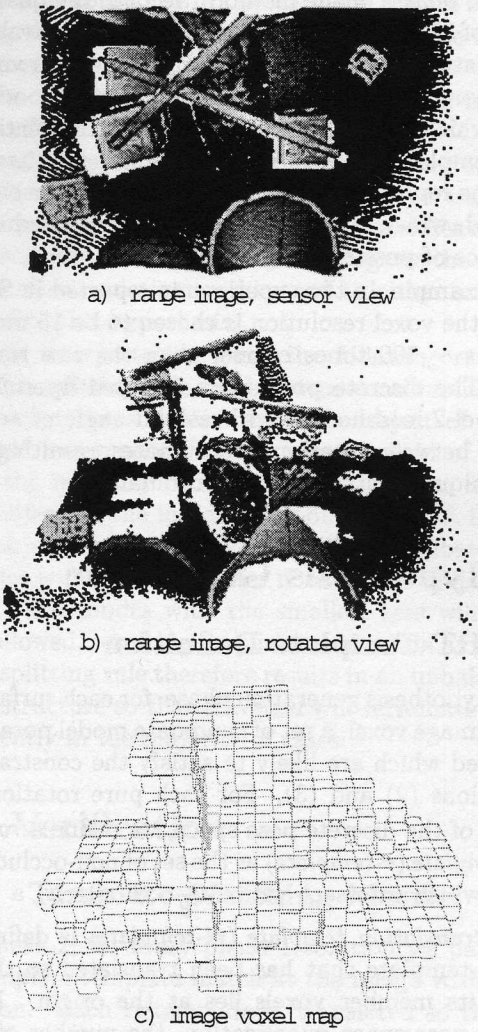

Figure 2: Voxelated Range Image

- **free (F)** : the voxel lies between the sensor head and a surface-valued voxel ;
- **occluded (O)** : the voxel lies behind a surface-valued voxel from the sensor head ;
- **unknown (U)** : the voxel lies outside of the sensing frustum.

An example of the surface-valued voxels of a voxelated range image is illustrated in Figure 2. It can be seen that the voxel resolution used in the experiments is coarser than is typically considered acceptable in image processing.

Mapping the problem into the discrete domain offers two advantages. First, it provides an efficient means to index a particular point in space to determine the presence or absence of surface-valued range

data, an operation which is used extensively in the first two phases of the method. Second, the quantization of 3D space into voxels imposes an equivalent quantization of the pose space : for any given voxel resolution, there is a continuous neighborhood of poses which map the model surfaces into identical voxel templates. The discrete pose space requires at most one representative pose value from each neighborhood, which effectively enumerates and reduces the space of possible model poses.

For example, in the experiments reported in Section 5, the voxel resolution is chosen to be 15 mm^3, which is $\sim$ 100 times coarser than the raw range data. The discrete pose space is based upon 1/2 of a level-2 icosahedron with a solid angle of $\sim$ 10 degrees between rotational coordinates, resulting in 5760 unique rotational pose coordinates.

3 Hypothesis Generation

3.1 RT-Template Definition

In the hypothesis generation phase, for each surface-valued image voxel, a set of candidate model poses is generated which are likely to satisfy the constraint inequations (2) and (3). For each pure rotational value $\vec{\theta}$ of the discrete pose space, we define a *rotation template* (*r-template*) as the set of non-occluded voxels which intersect with $M|_{\vec{\theta}}$. Similarly, a *rotation translation template* (*rt-template*) is defined as an r-template that has been translated so that one of its member voxels lies at the origin. For a given pose space quantization, the number of r-templates of a specific model will be equal to the number of unique rotation pose space coordinates, and the number of rt-templates will be equal to the number of r-templates multiplied by the average r-template size.

3.2 Template Matching As Decision Tree Classification

The objective of the Hypothesis Generation Phase is to find candidate matches of every rt-template against every image voxel. Due to the large number of rt-templates, the brute-force method of simply translating each rt-template to each surface-valued image voxel is too costly, even at a reduced resolution.

To improve efficiency, we compose the rt-template set into a binary decision tree classifier. In general, classification is a pattern recognition process whereby an input vector is determined to fall within some feature space region, or *class*. In cases where either the dimension of the input vector or the number of classes is large, efficiency can be improved with a multi-stage classifier, which considers only a small number of feature vector components at each stage, thereby reducing the set of possible classes that need be compared at each successive stage.

The root of a decision tree represents the entire set of classes, and each interior node represents a subset of the classes associated with its parent node. A leaf node represents either a single class, or a small set of classes. In addition to a class set, each non-terminal node has an associated feature set and decision rule. When traversing the tree, the decision rule is applied at each node to the associated feature subset of the input vector to determine which descendant branch to follow. The decision rule simultaneously measures the similarity of the feature subset with all remaining possible classes, and reduces their number accordingly. Decision trees have been used successfully in medical image processing [20] and OCR [21, 22] applications.

We call our decision tree a *sample tree*, because it is used to schedule the sampling of an image. Each node of the sample tree represents a set of rt-templates (the node set), and a relative voxel location (the node voxel). The root node set contains a reference to every rt-template. Descendant nodes partition their parent node set into two disjoint node sets, which are discriminated by the presence or absence of the parent node voxel. The node sets of all descendant nodes is the subset of rt-templates which best matches the node voxels encountered along the path to that node. A leaf node contains a small (possibly empty) node set, and no node voxel.

For example, for parent node P with node set N_p and relative voxel location V_p, all of the rt-templates referenced in the the TRUE child node set will contain V_p, and all of those referenced in the FALSE child node set will not.

3.3 Sample Tree Construction

The structure of a decision tree is primarily dependant upon which variable (in our case, which relative voxel) is interrogated at each tree node. Much of the literature on decision tree classifiers is concerned with methods of constructing trees which optimize certain properties, the most common objective being to minimize the expected traversal cost.

For even a modest number of classes, there are a huge number of possible tree structures, so it is infeasible to simply generate and compare all alter-

natives. There are construction methods which are known to minimize the expected testing cost, such as dynamic programming. All truly optimal methods, however, are computationally expensive, and have indeed been shown to be NP-complete [23].

Moret [24] has identified two classes of heuristic that are used in sub-optimal decision tree generation. In the *information heuristic*, a node variable is chosen that maximizes the amount of additional information that is gained at that stage along the path traversal. In the *splitting heuristic*, a variable is chosen which divides the remaining class set into 2 disjoint sets which are effectively discriminated by the node variable.

We have employed a splitting heuristic which, at any node along the traversed path, samples the point in space which is most likely to fall on the object. When constructing a tree node, this splitting rule selects the voxel which belongs to the largest number of rt-templates in the node set. This is similar to the heuristic used by Moret based upon variable activity.

The effect of this rule is that the difference between the cardinality of the two child node sets will be maximal, and the tree will be very unbalanced. This is contrary to most tree construction methods, which attempt to minimize the expected traversal cost by balancing the tree structure. A rational for favouring an unbalanced structure is presented in the following section.

3.4 Tree Traversal

Tree traversal proceeds by first selecting a surface-valued image voxel, called the *seed voxel*. The location of the seed voxel is the base which is used to relate all tree voxels to the image voxel map frame.

If the image data were perfect, then a simple depth-first traversal would arrive at the correct leaf node most efficiently. As each node is visited, the image voxel which is offset from the seed voxel by the node voxel location is queried. If this voxel is surface-valued data, then the traversal follows the *TRUE* path. If not, then the traversal follows the *FALSE* path.

In practise, image noise, occlusions, and quantization errors can cause a non-tentative search to follow an incorrect path. We therefore allow some limited backtracking to follow other likely paths, at a slight efficiency cost. The termination condition is satisfied when either a predefined number of solutions has been generated, or a predefined number of leafs has been visited.

The traversal is repeated for all possible image seeds. Whenever a leaf node is reached, the rt-templates contained in the leaf node set are stored, along with the seed voxel location, for use in the subsequent phase.

One way to interpret the tree traversal is as a form of scheduled subtemplate matching, where at each node the subtemplate consists of a single voxel. Another interpretation is that each tree traversal hypothesizes that the current seed voxel lies on the object, and then sequentially interrogates the image at locations relative to this seed to either support or refute this hypothesis.

This interpretation lends the following rational for the splitting rule used in tree construction: in all except the simplest cases, the object of interest will not monopolize the sensor's field of view, and the majority of image voxels will not fall on the object. If the initial seed point does not fall on the object, then as the path is followed, subsequent queries will likely encounter nonsurface-valued voxels. Those nodes with the smallest sets will then be followed, resulting in quick termination at a leaf. The splitting rule therefore results in an unbalanced tree structure so that traversal will terminate most efficiently in the majority of cases.

4 Verification

4.1 Template Matching Metric

In phase 2, for every phase 1 solution we perform a complete template match at the image voxel resolution. Each rt-template is translated so that it is aligned with its associated seed voxel. A simple correlation metric $\mathbf{r}$ is calculated by taking the ratio of the number of coincident surface voxels and the number of total rt-template voxels. If this measure exceeds a threshold value, then the result is passed to phase 3.

4.2 Closest Point Metric

In many cases, the highest ranked phase 2 solutions are correct. Sometimes however, due to the coarseness of the quantization, an incorrect solution may have a higher correlation measure than a correct solution. This will occur less frequently with a finer discrete resolution, but too fine a resolution will negate the benefits of quantization.

In phase 3, the phase 2 results are ranked by estimating the values of the constraint functions at the image data resolution. In preprocessing, two point sets are generated by sampling the surfaces of $\mathbf{M}$ at a coarse and fine resolution. These are called

the *coarse* and *fine surface samples*. The constraint function values are estimated by transforming the coarse surface sample in turn to the pose value of each phase 2 solution. For each point in the transformed surface sample, the identity of and separation distance to the closest image point is calculated. The area of overlap is estimated by counting the number of image points that are within a specified threshold distance, which is approximately equal to the voxel resolution. The volume of overlap is estimated by averaging the separation distances.

The closest point metric is subsequently estimated more accurately using the fine surface sample for the top 10 ranked solutions from the coarse surface sample estimate.

5 Experimental Results

The method has been implemented and tested on a hand-carved duck object, which was chosen specifically as it was free-form surface with no planes, edges, or other obvious features. Executing on a standard sequential computer (an SGI workstation with a MIPS R8000 CPU) the results compare favourably to other known methods, with recognition often achieved in less than 20 seconds.

The object was positioned arbitrarily in scenes of various degrees of clutter and occlusion. In some cases, the object was partially occluded by other objects, or partially outside of the sensor's field of view. Dense images were acquired using a laser range scanner based on the autosynchronous scanning method [25]. The scanner, which was mounted on the end effector of a robotic manipulator, was oriented at various angles and distances from the table top environment. The particular range scanner used was an early prototype which had fallen into disuse, and so the images were noisier than is achievable with current sensor versions.

Examples of three successful trials are illustrated in Figures 3 to 5. For each trial the range image is illustrated from the sensor's vantage, and two views are shown of the object superimposed onto the image at the recognized pose. The timing results of the various phases of these trials are tabulated in Table 1. Also included are: the number of solutions resultant from each phase; the phase 1 efficiency **E** (the ratio of the number of brute-force template element comparisons and the number of comparisons resulting from the tree traversal); the best phase 2 correlation measure **r** (as defined in Section 4); and an estimate of the volume constraint $\hat{\mathcal{V}}()$ (the average closest point separation).

6 Summary

A novel object recognition method has been presented, and early implementation results are very promising. Some of the benefits of this method are;

- **efficiency**: The method is fast. In experiments, the full 6 dimensional pose of objects were determined in $\approx$ 20 seconds. The time complexity of the tree traversal phase is related to the base 2 logarithm of the number of templates encoded in the tree.

- **free-form surfaces**: As there is no dependance upon a feature extraction preprocessing phase, the method is applicable to any shape of rigid object, including free-form surfaces.

- **fully automatic**: The input to the offline processing is either a CAD model or a cloud of points which sample the surface of the object, and the tree generation is fully automatic, requiring no user interaction other than the selection of a handful of parameters at the offset.

- **scalable**: The tree may be composed of a template set derived from a number of objects. Increasing the number of objects has a less than linear performance penalty on the tree traversal phase. A practical limitation to the number of objects simultaneously encoded into a single tree is likely to be the storage requirements, as the size of the tree increases exponentially with the number of templates.

- **parallelizable**: The process can be executed independantly accross seeds (seed parallelism) or piped accross phases (phase parallelism).

- **other sensing modes**: The method can be applied directly to 2D binary images (as in OCR) or volumetric data (as in [16]).

The continuing research will focus on revisiting some of the heuristics and algorithms used with the goal of improving upon the overall reliablility and efficiency of the method. It may be beneficial to add a hashing phase following the hypothesis generation phase, as in [18]. Additional tree construction and traversal heuristics will be investigated, and efficiency and reliability metrics will be thoroughly evaluated.

Acknowledgements

I would like to thank my thesis supervisors, Pierre Boulanger and Bernie Pagurek, for their support

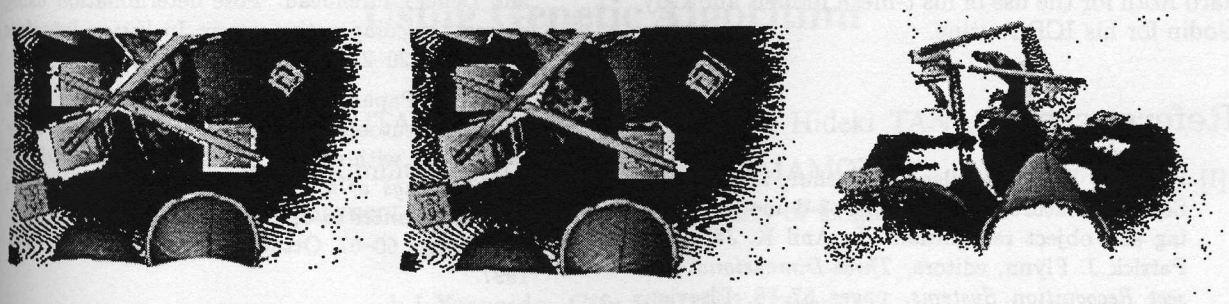

Figure 3: Image 1 : t = 10.5 seconds

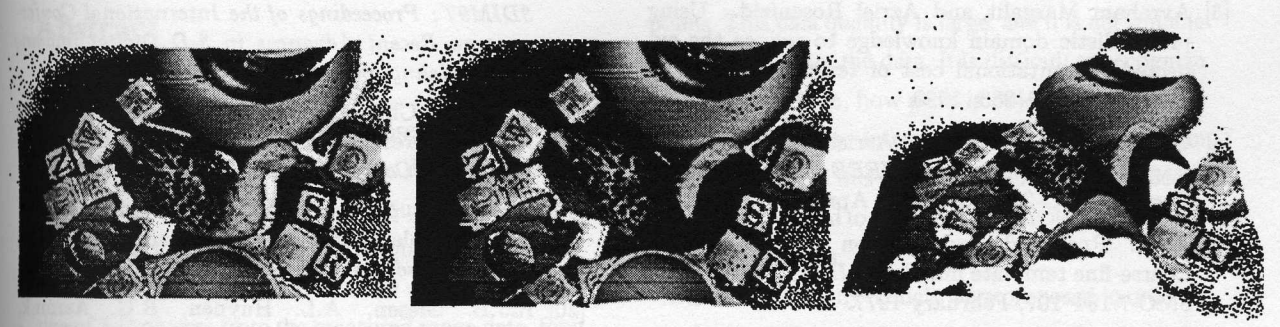

Figure 4: Image 2 : t = 14.0 seconds

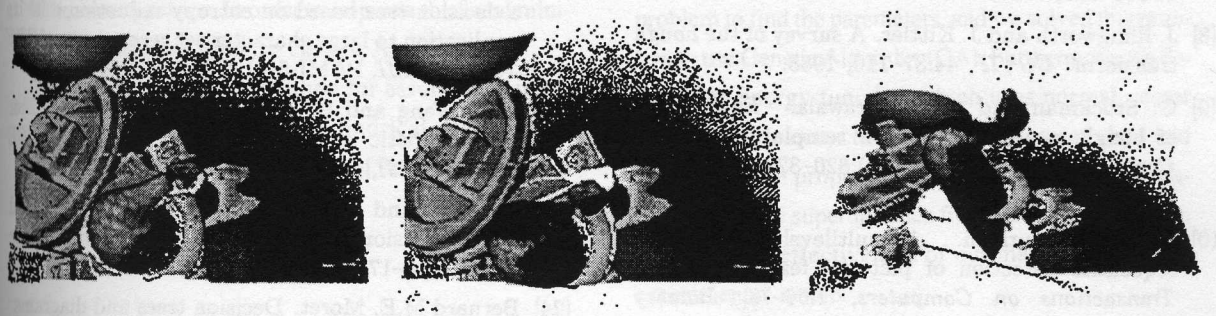

Figure 5: Image 3 : t = 21.9 seconds

	time (seconds)				*# of solutions*				
image	*total*	*phase 1*	*phase 2*	*phase 3*	*phase 1*	*phase 2*	E	r	$\hat{\mathcal{V}}()$
1	10.5	3.8	3.5	3.2	12660	78	17685	0.83	5.3e-5
2	14.0	3.8	3.9	6.3	14253	156	18022	0.94	1.6e-5
3	21.9	3.4	3.7	14.8	14024	335	17996	0.82	1.2e-4

Table 1: Test Results

and many helpful suggestions. Thanks also to Gerhard Roth for the use of his t-mesh models and Guy Godin for his ICP routine.

References

[1] Frank P. Ferrie, Shailendra Mathur, and Gilbert Soucy. Feature extraction for 3-d model building and object recognition. In Anil K. Jain and Patrick J. Flynn, editors, *Three-Dimensional Object Recognition Systems*, pages 57–88. Elsevier, 1993.

[2] Roger N. Nagel and Azriel Rosenfeld. Ordered search techniques in template matching. *Proceedings of the IEEE*, 60:242–244, February 1972.

[3] Avraham Margalit and Azriel Rosenfeld. Using probabilistic domain knowledge to reduce the expected computational cost of template matching. *CGVIP*, 51:219–233, 1990.

[4] Gordon J. Vanderbrug and Azriel Rosenfeld. Two-stage template matching. *IEEE Transactions on Computers*, C-26(4):384–393, April 1977.

[5] Azriel Rosenfeld and Gordon J. Vanderbrug. Coarse-fine template matching. *IEEE Trans. SMC*, SMC-7:104–107, February 1977.

[6] Robert Y. Wong and Ernest L. Hall. Sequential hierarchical scene matching. *IEEE Transactions on Computers*, C-27(4):359–366, April 1978.

[7] Steven L. Tanimoto. Template matching in pyramids. *Computer Graphics and Image Processing*, 16:356–369, 1981.

[8] J. Illingworth and J. Kittler. A survey of the hough transform. *CGVIP*, 44:87–116, 1988.

[9] G. Stockman and A.K. Agrawala. Equivalence of hough curve detection to template matching. *Comm. of the ACM*, 20(11):820–822, November 1977.

[10] H. K. Ramapriyan. A multilevel approach to sequential detection of pictorial features. *IEEE Transactions on Computers*, 1:66–78, January 1976.

[11] Paul J. Besl and Heil D. McKay. A method for registration of 3d shapes. *IEEE Trans. PAMI*, 14(2):239–256, February 1992.

[12] W.E. Lorenen and H.E. Cline. Marching cubes : A high resolution 3d surface reconstruction algorithm. In *Computer Graphics : Siggraph '87 Conference Proceedings*, volume 21, pages 163–169, July 1987.

[13] Gerhard Roth amd Eko Wibowoo. An efficient volumetric method for building closed triangular meshoes from 3-d image and point data. In *Graphics Interface '97*, pages 173–180, May 1997.

[14] Régis Houde, Jacques Tremblay, Frédéric Auclair, and Denis Laurendeau. Pose determination using octrees : A comparative survey. In *Vision Interface 95*, pages 220–227, 1995.

[15] Dimitri Papadopoulos-Orfanos and Francis Schmitt. Automatic 3-d digitization using a laser rangefinder with a small filed of view. In *3DIM97 : Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 60–67, Ottawa, Ontario, Canada, May 1997.

[16] Gill Barequet and Micha Sharir. Partial surface and volume matching in three dimensions. *IEEE Trans. PAMI*, 19(9):929–948, 1997.

[17] Andrew Edie Johnson and Martial Hebert. Surface registration by matching oriented points. In *3DIM97 : Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 121–128, May 1997.

[18] Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *Intl. Jour. Comp. Vis.*, 25(1):63–85, 1997.

[19] Timothy S. Newman and Anil K. Jain. Bidirectional template-matching for 3d cad-based inspection. *Proceedings of the SPIE*, 2183:257–265, 1994.

[20] R.J.B. Giesen, A.L. Huynen, R.G. Aarnink, J.J.M.C.H. de la Rosette, F.M.J. Debruyne, and H. Wijkstra. Construction and application of hierarchical decision tree for classification of ultrasonic prostate images. *Medical & Biological Engineering & Computing*, pages 105–109, March 1996.

[21] Q.R. Wang and C.Y. Suen. Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition. *IEEE Trans. PAMI*, PAMI-6(4):406–417, July 1984.

[22] Q.R. Wang and C.Y. Suen. Large tree classifier with heuristic search and global training. *IEEE Trans. PAMI*, PAMI-9(1):91–102, January 1987.

[23] L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is np-complete. *Comm. of the ACM*, 5:15–17, May 1976.

[24] Bernard M.E. Moret. Decision trees and diagrams. *ACM Comp. Surv.*, 14(4):593–623, December 1982.

[25] Marc Rioux. Laser range finder based on synchronized scanners. In Thierry Bosch and Marc Lescure, editors, *Analysis and Interpretation of Range Images*. Springer-Verlag, 1990. Also, Appl. Opt. 23(21): 3837-3844; 1984, NRC 24528.