Approximate K-D Tree Search for Efficient ICP

Michael Greenspan[1,2] Mike Yurick[2]

michael.greenspan@ece.queensu.ca mike@biomed.queensu.ca

[1]Dept. of Electrical and Computer Engineering
[2]School of Computing
Queen's University, Kingston, Ontario, Canada, K7L 3N6

Abstract

*A method is presented that uses an Approximate Nearest Neighbor method for determining correspondences within the Iterative Closest Point Algorithm. The method is based upon the k-d tree. The standard k-d tree uses a tentative backtracking search to identify nearest neighbors. In contrast, the Approximate k-d tree (*Ak-d tree*) applies a depth-first nontentative search to the k-d tree structure. This search improves runtime efficiency, with the tradeoff of reducing the accuracy of the determined correspondences.*

This approximate search is applied to early iterations of the Iterative Closest Point Algorithm, transitioning to the standard k-d tree for the final iterations after the change in the mean square error of the correspondences becomes sufficiently small. The method benefits both from the improved time performance of the approximate search in early iterations as well as the full accuracy of the complete search in later iterations.

Experimental results indicate that the time efficiency of Ak-d tree is superior to the k-d tree and Elias for moderately large point sets. The change in the shape of the minimum potential well space is subtle, and the convergence properties are often identical. In those cases where the global minimum was not achieved, the difference in final mse *was very small. In one trial, Ak-d tree converged faster to a better minimum with a smaller* mse*, which indicates that the use of approximate methods may be beneficial in the presence of outliers.*

Keywords: *nearest neighbor, k-d tree, iterative closest point, registration, range image, computational geometry*

1 Introduction

The Iterative Closest Point Algorithm (*ICP*) [1] is a method for bringing into registration two partially overlapping but slightly misaligned range images. ICP has proven to be very useful in the processing of range data, and a number of variations on the basic method have been proposed [2]. These include alternate transformation calculations [3], the use of intensity information and curvature [4], and outlier selection filters.

At each iteration, corresponding points between the two images are determined by a Nearest Neighbor (*NN*) method. Determining these correspondences accounts for the majority of the runtime expense of ICP. In their original paper Besl and McKay [1] suggested the use of the k-d tree method. They also indicated that other NN methods may be suitable and there have been some prior explorations into the use of alternatives. Blais and Levine [5] used the inverse calibration parameters of the acquisition sensor to project the points from one image onto the reference frame of the other image. While this method was efficient, it required a normal estimate for each point and did not necessarily identify the true nearest neighbor correspondences. Simon et al. [6] used a caching technique similar to Voronoi methods to exploit the spatial coherency between neighboring points. Greenspan and Godin [7] developed a method that preprocessed spherical neighborhoods of each point and tracked their evolution across iterations.

In this paper we explore the use of an Approximate Nearest Neighbor (*ANN*) method for improving the runtime efficiency of ICP. True NN methods return the point $\vec{p}_c$ from a set $\mathbf{P}$ that is the minimum distance from the query point $\vec{q}$;

$$\|\vec{q} - \vec{p}_c\| \leq \|\vec{q} - \vec{p}_i\| \ \ \forall \ \ \vec{p}_i \in \mathbf{P} \tag{1}$$

Alternately, ANN methods return an approximation $\hat{\vec{p}}_c$ of $\vec{p}_c$ whose distance from $\vec{q}$ is within some bound $(1+\epsilon)$ of the distance between $\vec{q}$ and $\vec{p}_c$;

$$\|\vec{q} - \hat{\vec{p}}_c\| \leq (1+\epsilon)\|\vec{q} - \vec{p}_c\|, \epsilon \geq 0 \tag{2}$$

The main benefit of ANN methods is improved time

Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM 2003)
0-7695-1991-1/03 $17.00 © 2003 IEEE

performance over their exact couterparts. This has been of particular interest for high dimensional spaces where true NN methods can become computationally expensive.

This paper continues in Section 2 with a description of the basic k-d tree, followed by the introduction of a modified search of the k-d tree that we shall call the *Approximate k-d tree* (*Ak-d tree*) in Section 3. In Section 4, Ak-d tree is applied to ICP and its runtime performance is compared to the k-d tree and Elias methods. Tests on more general data sets are presented in Section 5, and the paper concludes with a summary and a discussion of future research directions in Section 6.

2 k-d Tree

The k-d tree was discovered by Bentley [8] and is likely the most well-utilized NN method. The k-d tree is a binary search tree in which each node represents a partition of the k-dimensional space. The root node represents the entire space, and the leaf nodes represent subspaces containing mutually exclusive small subsets of $\mathbf{P}$. At any node, only one of the k dimensions is used as a discriminator, or *key*, to partition the space. When the tree is searched, the single scalar value of the key of $\vec{q}$ is compared against the node key value, and the appropriate branch is followed. When a leaf node is encountered, all of the B points resident in the leaf's bin region are tested against $\vec{q}$, and the closest bin point $\vec{p}_b$ is determined.

It happens that the true nearest neighbor may lie in a different bin than $\vec{q}$. This will occur when the distance between $\vec{q}$ and a boundary of its bin region is less than the distance between $\vec{q}$ and the closest bin point $\vec{p}_b$. Conversely, $\vec{p}_b$ is guaranteed to be the true nearest neighbor if the sphere centered at $\vec{q}$ with radius $\|\vec{q}-\vec{p}_b\|$ is completely contained within the bin region. This is known as the Ball-Within-Bounds (**BWB**) test.

If the **BWB** test fails, then $\vec{p}_b$ may not be the true nearest neighbor, and it is necessary to backtrack up the tree and test points contained in alternate paths. The Bounds-Overlap-Ball (**BOB**) test determines whether or not the sphere centered at $\vec{q}$ intersects with some region, which may therefore contain the true nearest neighbor. All points contained in all bin regions that pass the **BOB** test must be considered during backtracking. If a new nearest neighbor is encountered, then the sphere radius is adjusted downward, the **BWB** test is repeated, and the backtracking resumes if necessary.

The performance of the k-d tree search is $O(\log n)$, and the average number of distance computations is minimal when the bin size is smallest, i.e., $B = 1$. Bentley pointed out that the average runtime cost is minimized for a slightly larger bin size [8]. This is due to the greater cost of backtracking as compared to a simple linear traversal of a small list. (This property is also observed in other elementary logarithmic algorithms, such as Binary Search and Quicksort.) In practise it is typical to set the bin size to a default value of $B = 10$ or 20.

3 Approximate k-d Tree Search

The idea behind the Ak-d tree is to return as an approximate nearest neighbor $\hat{\vec{p}}_c$ the closest point $\vec{p}_b$ in the bin region where $\vec{q}$ lies. This value is determined from the initial depth-first search, and so the expense of backtracking is avoided. It is also no longer necessary to execute the **BWB** test, which further improves runtime efficiency.

The tradeoff for this enhanced efficiency is that, in cases where the true nearest neighbor does not lie in the same bin as $\vec{q}$, only an approximate result will be returned. We have found, however, the true nearest neighbor is often found, especially when the bins are large. Further, when the true nearest neighbor is not found, the approximation is often very close to the true value, due to the proximity of points within a region.

We implemented the method and tested it on two sets $\mathbf{P}$ and $\mathbf{Q}$ of 10000 3-D points distibuted uniformly and randomly over the unit cube. $\mathbf{P}$ was used to construct the Ak-d tree with a given bin size, and $\hat{\vec{p}}_{ci}$ was determined for each point $\vec{q}_i$ in $\mathbf{Q}$. Following this, a full (i.e., backtracking) k-d tree search was executed on the same tree structure, and the true $\vec{p}_{ci}$ for each $\vec{q}_i$ was determined. Figure 1 shows a histogram that tabulates the frequency of distances between the respective $\hat{\vec{p}}_{ci}$ and $\vec{p}_{ci}$ for bin size $B = 50$. The large spike at value 0 indicates that a large percentage (79 $\%$) of the resulting approximate values were in fact true nearest neighbors, i.e., $\hat{\vec{p}}_{ci} = \vec{p}_{ci}$. The remaining approximations formed a bell-shaped distribution centered around 0.05, which was roughly twice the interpoint separation of $\mathbf{P}$.

The above results are plotted in Figure 2, with the true separations $\|\vec{p}_{ci}-\vec{q}_i\|$ on the x-axis and the corresponding differences between the true and approximate results $\|\vec{p}_{ci}-\hat{\vec{p}}_{ci}\|$ on the y-axis. The solid bar of values at $y = 0$ represents the cases where $\hat{\vec{p}}_{ci} = \vec{p}_{ci}$. The remaining relations indicate that there is a strong correlation between the true separation and the closeness of the approximation. We can therefore generally expect $\hat{\vec{p}}_{ci}$ to be a tighter approximation when $\vec{p}_{ci}$ is nearer to $\vec{q}_i$.

Another experiment characterized the effect of bin size on the tightness of the approximation. Trees were constructed for a variety of bin sizes from $B = 5$ to 295, in increments of 5, and for each tree the true and approximate searches were executed. The percentage of trials where the resulting true and approximate results were the same was plotted as a function of B in Figure 3. As expected, the approximate search had a greater likelihood of finding the true nearest neighbors for larger bins. In the extreme case when $B = N_{\mathbf{P}}$, all $N_{\mathbf{P}}$ points in $\mathbf{P}$ would reside in the

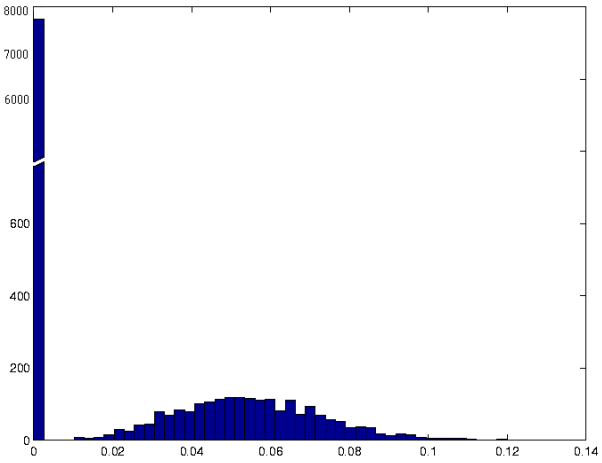

Figure 1: Histogram of $\|\vec{p}_c - \hat{\vec{p}}_c\|$

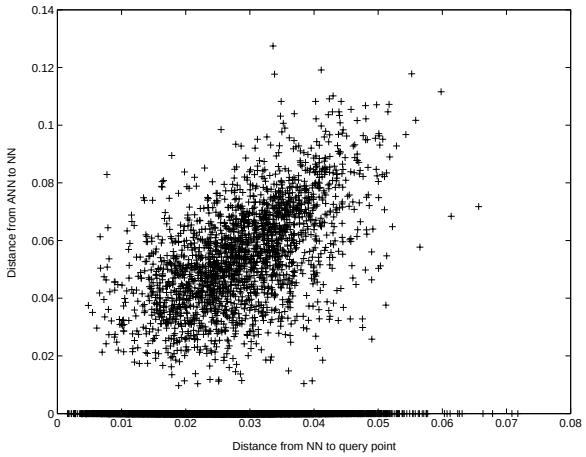

Figure 2: $\|\vec{q} - \hat{\vec{p}}_c\|$ vs. $\|\vec{p}_c - \hat{\vec{p}}_c\|$

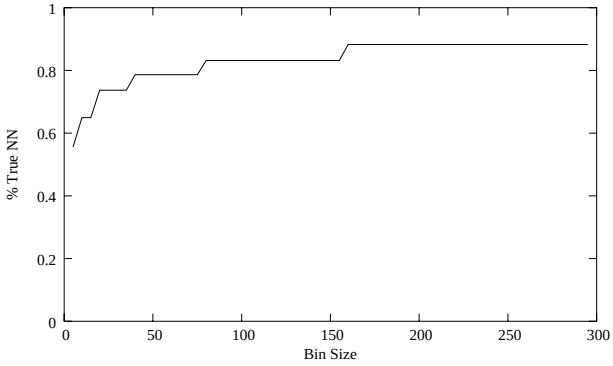

Figure 3: Binsize (B) vs. % True Nearest Neighbors Found

a) sensor view

b) rotated view

Figure 4: Test Image (56260 points)

bin of the single tree node, and the two searches would be equivalent.

4 Application to ICP

At each ICP iteration, correspondences are determined based upon the current relative position of the two surfaces **P** and **Q** under registration. The unique transformation that minimizes the mean square error (i.e., the *mse*) of the correspondences is then calculated and applied to **Q** prior to the next iteration. The algorithm is said to *fully converge* when the *mse* remains constant for two consecutive iterations. This occurs when there is no change in the correpsondences between two iterations, and there would therefore be no change in the correspondences or the *mse* in any subsequent iterations.

If the initial misregistration between the two surfaces is small, then the ICP will converge to a *global minimum* resulting in a correct registration. Alternately, if the initial misregistration is large then ICP will converge to a *local minimum* resulting in an incorrect registration. The *minimum potential well space* is the set of initial relative transformations between two surfaces that would result in correct convergence to a global minimum.

If an ANN were used in place of an NN to calculate correspondences at each iteration, then convergence could be affected. As seen in Figure 3, for reasonably large bin sizes there will still be some points $\vec{q}_i$ whose approximate nearest neighbors $\hat{\vec{p}}_{c_i}$ will not equal their true nearest neighbors $\vec{p}_{c_i}$. In these cases, even though $\hat{\vec{p}}_{c_i}$ may be close to $\vec{p}_{c_i}$ (especially when $\|\vec{p}_{c_i} - \vec{q}_i\|$ is small as evident in Figure 2) the convergence properties will be different. The minimum

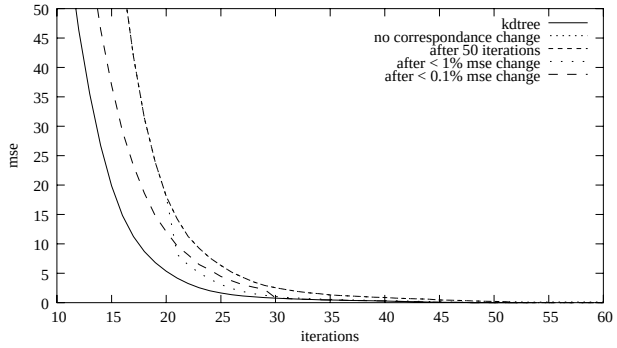

Figure 5: Iterations vs. *mse*

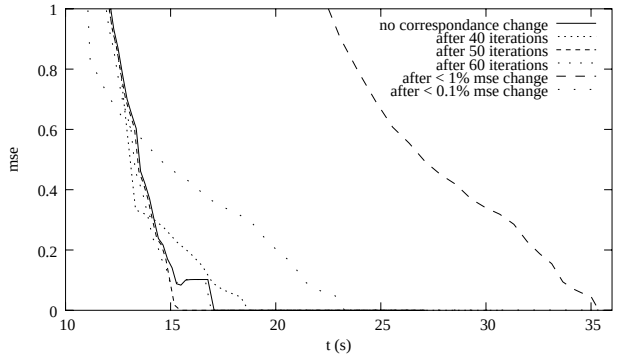

Figure 6: Time (s) vs. *mse*

potential well space will change as a result, and the global minimum may never be realized.

Rather than using a pure ANN method, it is therefore attractive to base an ICP on a hybrid of ANN and NN. The simplest possibility is to start the iterations using the Ak-d tree search and at some iteration transition to the use of the k-d tree. With this scheme ICP would benefit both from the improved time efficiency of the Ak-d tree in early iterations, and the full accuracy of the k-d tree in later iterations. The transition could occur after a certain fixed number of iterations, after the *mse* becomes sufficiently small, or after the change in the *mse* across iterations is sufficiently small.

These three variations of the hybrid method were implemented and tested on the data set illustrated in Figure 4. This range image contained 56260 points and was acquired with a Biris range sensor mounted on a linear motion stage. The floating point set $\mathbf{Q}$ was an exact copy of $\mathbf{P}$ that was perturbed to a slightly different initial position. The rotational offset was 5 degrees about all 3 axes, and the translational offset in all 3 directions was 50 mm, which was $\sim 25\%$, 14%, and 47% of the extent of the image along the x-, y-, and z-axes respectively. The main benefits of an experimental setup where $\mathbf{P} = \mathbf{Q}$ is that an exact convergence event can be easily identified when all of the indeces between corresponding points are respectively equal. When this occurs, the *mse* will vanish in the subsequent iteration.

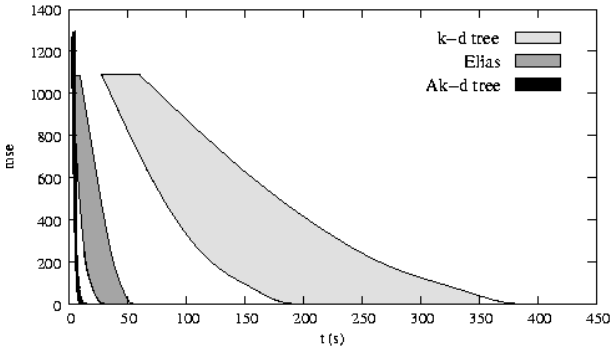

Figure 7: Time (s) vs. *mse*

A set of k-d trees were constructed using bin sizes of $B = 20$ and 100. Trials of the ICP were executed for each tree starting from the same initial misregistration and using both the full k-d tree search, as well as the hybrid method that transitioned from the Ak-d tree to the k-d tree search after a fixed number of iterations.

Figure 5 plots the evolution of the *mse* for these trials as a function of number of iterations. The *mse* is seen to reduce earlier for the k-d tree trial. For each of the hybrid trials, following the transition the *mse* values for the hybrid trials still lag slightly behind that of the k-d tree trial. Eventually, all of the trials converge to the global minimum, with the k-d tree reaching this point with fewer iterations.

In the hybrid trials, prior to the transition some of the correspondences computed by the Ak-d tree search are approximate. The calculated *mse* is therefore an overestimate of the true *mse* that would have been calculated were an NN method used to determine all correspondences. After the transition, all correspondeces are calculated using the full backtracking k-d tree search, and so the calculated *mse* will be correct.

A cursory examination of the plot in Figure 5 might lead one to conclude that k-d tree outperforms the Ak-d tree hybrids. An examination of the relative time performance of the two methods, however, supports the opposite conclusion. Figure 6 shows a plot of the evolution of *mse* vs. elapsed runtime for the same set of trials. The k-d tree trial is not shown as its time exceeded the scale of this plot. Irrespective of the transition criterion, the hybrid method converges completely before the k-d tree has completed its second iteration. Although Figure 5 shows that k-d tree minimizes *mse* in fewer iterations, it is clear that the different methods iterate with different time efficiencies. The k-d tree iterations are more time expensive than those of

the Ak-d tree, especially in early iterations when the $\vec{q}_i$ are farthest from their correspondents, and the expense of computing the true NN is greatest.

Figure 7 plots the time results from an additional set of trials that included tests of an implementation of the Elias method. The Elias method is a true NN method which subdivides space regularly and groups $\mathbf{P}$ into voxels. The k-d tree bin size was varied over a range of $B = [10, 200]$ and the voxel bin parameter for the Elias method was varied over a range of $V = [20, 85]$, where the total number of voxels is equal to V^3. These ranges included the optimal values for both methods. We varied the Ak-d tree bin size over the range of $B = [1, 200]$.

The Ak-d tree method performed very favorably, converging with a best time of 16 seconds as compared to 41 seconds for Elias and 210 seconds for k-d tree. These times included the costs of constructing the initial data structures, which was 2.2 seconds for k-d tree and Ak-d tree (which used the same tree structure) and 0.1 seconds for Elias. The best time performance for k-d tree occured for $B = 100$, while the Ak-d tree was optimal for a smaller bin size of $B = 20$. In addition to being more time efficient, the Ak-d tree hybrid method has the added advantage of being less sensitive to the selection of parameters values. This makes it more likely to get a good result with an arbitrary choice of bin size, as often occurs in practise.

When implementing the above methods, a moderate and equivalent amount of attention was afforded to the optimization of each. For example, square root calculations were avoided wherever possible, but no assembly level coding or other machine specific techniques were used. The implementations were developed with an eye to making the timing comparisons as fair as possible. It is believed that these implementations (and thus the measured time values) are representative of what a practitioner in the field would typically reproduce. The comparisons between the k-d tree and Ak-d tree methods are particularly even-handed, as the exact same code was used with only minor modifications between the two searches.

5 Experiments

A set of experiments was executed to determine the effectiveness of the method on a more general data set, as may be encountered in a model-building application. Range images of four toy objects (a Brachiosaurus, Mr. Clown, a Truck, and TRex) were collected using a Biris range scanner mounted on the end-effector of a CRS A465 6 dof articulated manipulator, shown in Fig. 8. Biris is a profile sensor, and the most distal joint of the robot was used as the scanning mechanism. The encoders readings of the remaining 5 joints allowed each scanned image to be localized within the robot coordinate frame.

From this data set, the 6 image pairs illustrated in Fig. 9

Figure 8: Robot Mounted Scanner

were selected to perform the tests. The version of ICP that was used did not have any outlier filter mechanism, so pairs were selected that had a significant amount of overlap, although some outliers did exist. The initial misalignment within the image pairs was due to the absolute robot positioning error. The degree of misalignment depends upon a number of factors including the length of the path that the robot travelled between scan locations. It is believed that these initial misalignments are similar to those that typically result from manually assisted pre-alignment, which is a common preprocessing stage prior to ICP.

For each image pair ICP iterated to convergence using both the k-d tree and the Ak-d tree hybrids. The Ak-d tree hybrid method transitioned to the use of k-d tree at the iteration when the *mse* drew less than a threshold percentage of the intial *mse* as measured in the first iteration. The two transition thresholds tested were 1% and 0.1%. The results were manually inspected and in each case all registrations were successful. The resulting alignments of the two TRex image pairs using Ak-d tree are illustrated in Fig. 10

The results for these trials are tabulated in Table 1. In each test, the time to converge (t_f) was faster for both versions of the Ak-d tree than for the k-d tree. For half of the trials, the final *mse* (mse_f) was identical for the at least one version of the Ak-d tree and the true methods. For two of the six tests (Clown, Truck), the Ak-d tree trials resulted in a larger final *mse*, although by a very small margin.

The most remarkable result was from the TRex 2 test where the Ak-d tree trials were not only faster than either the k-d tree or the Elias trials, but they also *converged to a smaller final* mse. This indicates that ANN can converge to

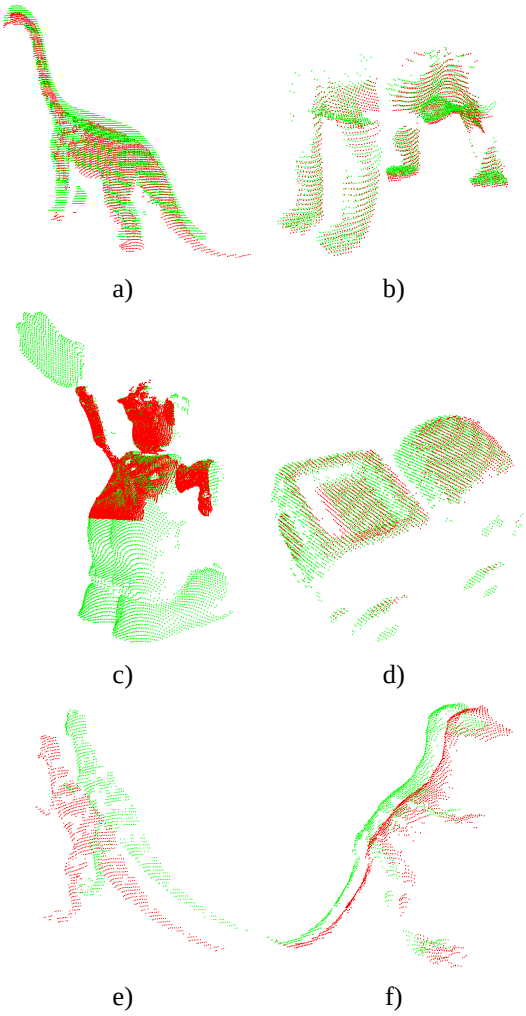

Figure 9: Initial Misregistered Image Pairs

a smaller minimum *mse* when outliers are present, which is almost always the case in real data sets.

A comparison of the Ak-d tree against Elias showed roughly equal performance. The Ak-d tree had a better time performance in two of the six tests. In both of these tests the *mse* was the same or better than that of the Elias trial.

6 Discussion and Summary

We have demonstrated the use of an ANN to enhance the time performance of ICP, and our experimentation has revealed a number of interesting properties. The time efficiency of the hybrid method is better than the standard k-d tree. In initial tests with a moderately sized image of 56260 points, performance was improved significantly. On tests with smaller image pairs there was also a consistent time performance improvement when compared with optimal choice of binsize for k-d tree. The images used in these tests were much smaller than those often encountered in practise, which may easily be 2 to 3 orders of magnitude larger. Any speedup of Ak-d tree over k-d tree or Elias will scale with image size, as is evident from a comparison with the results achieved in Section 4. We therefore expect the time savings to be proportionally larger for larger images.

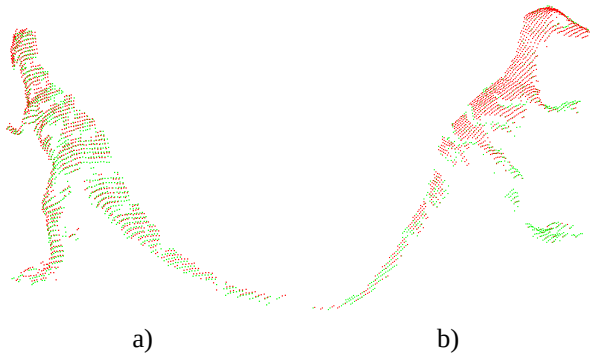

Figure 10: Registration Results

The shape of the potential well is affected by the ANN, although out experimentation indicated that the change is not drastic. Half of the six tests on real image pairs resulted in the exact same final *mse* using either k-d tree or Ak-d tree. In two of the trials the Ak-d tree final *mse* was larger, although only by a slight margin. In the remaining trial, the ANN final *mse* was significantly *smaller* than that of the k-d tree and Elias trials. This result indicates that the use of a true NN to calculate ICP correspondences does not guarantee converge to the global minimum when outliers are present. The reason is that outliers by definition do not have true correspondences, and there is no particular reason to believe that the true nearest neighbors of outliers represent an optimal choice of correspondent. An explanation for the observed behaviour is that the correspondence returned from the Ak-d tree is less likely to be a true NN when the separation is greater, which is generally the case for outliers. As it is not straightforward to detect and remove outliers, a conclusion that we draw from this result is that the use of an ANN may be an effective method to improve ICP convergence in the presence of outliers.

For moderately large point sets, the time performance of the hybrid method was better than both k-d tree and Elias, and it is expected that this result will also hold for large point sets. For smaller sets, the time performance was comparable to Elias, which is known to be very time efficient, with $O(\log N)$ expected time performance. Despite its time efficiency, Elias is often considered inpractical due to its large memory requirement which is proportional to the volume of the image i.e., $O(N^3)$. This is of particular

Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM 2003)
0-7695-1991-1/03 $17.00 © 2003 IEEE

		$N_{\mathbf{P}}$	$N_{\mathbf{Q}}$	B	i_x	i_f	$\boldsymbol{mse}_f$	t_x	t_f
Brach 1	k-d tree	5008	3284	20	-	69	**5.188546**	-	**2.21**
	Elias	”	”	30	-	”	”	-	**1.39**
	Ak-d tree(1%)	”	”	20	15	63	”	0.22	**1.55**
	Ak-d tree(0.1%)	”	”	20	15	63	”	0.21	**1.52**
Brach 2	k-d tree	2125	2905	20	-	46	**3.161124**	-	**1.01**
	Elias	”	”	20	-	”	”	-	**0.67**
	Ak-d tree(1%)	”	”	20	8	41	”	0.09	**0.79**
	Ak-d tree(0.1%)	”	”	10	12	28	**3.161231**	0.11	**0.47**
Clown	k-d tree	6224	16604	50	-	44	**21.823087**	-	**9.85**
	Elias	”	”	35	-	”	”	-	**5.80**
	Ak-d tree(1%)	”	”	20	7	36	**21.823814**	0.52	**6.77**
	Ak-d tree(0.1%)	”	”	75	10	56	**21.823087**	1.02	**8.81**
Truck	k-d tree	3962	2438	20	-	41	**2.165853**	-	**0.95**
	Elias	”	”	20	-	”	”	-	**0.50**
	Ak-d tree(1%)	”	”	20	4	32	**2.165873**	0.09	**0.69**
	Ak-d tree(0.1%)	”	”	10	5	36	”	0.10	**0.70**
TRex 1	k-d tree	1457	1573	20	-	25	**0.753817**	-	**0.28**
	Elias	”	”	25	-	”	”	-	**0.19**
	Ak-d tree(1%)	”	”	20	11	25	”	0.06	**0.18**
	Ak-d tree(0.1%)	”	”	10	16	21	**0.753835**	0.07	**0.12**
TRex 2	k-d tree	1068	1075	20	-	37	**2.007675**	-	**0.26**
	Elias	”	”	25	-	”	”	-	**0.19**
	Ak-d tree(1%)	”	”	10	8	33	**1.956773**	0.03	**0.14**
	Ak-d tree(0.1%)	”	”	10	20	34	”	0.05	**0.11**

Table 1: Experimental Results

concern for large images. In contrast, tree-based methods have a memory requirement of only $O(N)$.

An attractive practical aspect of the Ak-d tree search is that it is trivial to implement given an existing k-d tree method, which most ICP implementations are based upon. Its performance also appears to be relatively insensitive to the selection of bin size.

In future work, we plan to further explore the effects of the use of the Ak-d tree search on ICP convergence. We will investigate the effects of ANN on point thinning techniques, which can also improve the time performance of ICP, and we will investigate the use of other ANN methods for enhanced ICP.

Acknowledgements

The authors would like to thank the reviewers for their helpful comments, and Guy Godin, Chang Shu, and Alan Brunton for their contributions.

References

[1] Paul J. Besl and Neil D. McKay. A method for registration of 3d shapes. 14(2):239–256, February 1992.

[2] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3DIM01: 3D Imaging and Modelling*, pages 145–152, 2001.

[3] Y. Chen and G.G. Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[4] Guy Godin, Denis Laurendeau, and Robert Bergevin. A method for the registration of attributed range images. In *3DIM01: 3D Imaging and Modelling*, pages 179–186, 2001.

[5] Gérard Blais and Martin D. Levine. Registering multiview range data to create 3D computer objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824, August 1995.

[6] David A. Simon, Martial Hebert, and Takeo Kanade. Real-time 3-D pose estimation using a high-speed range sensor. In *IEEE Intl. Conf. Robotics and Automation*, pages 2235–2241, San Diego, California, May 8-13 1994.

[7] Michael Greenspan and Guy Godin. A nearest neighbor method for efficient icp. In *3DIM01: 3D Imaging and Modelling*, pages 161–168, 2001.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.