# Parallel Evolutionary Registration of Range Data

Craig Robertson[1] and Robert B. Fisher

*Vision Group, Institute for Action, Perception and Behaviour,*
*Division of Informatics, University of Edinburgh,*
*Edinburgh, UK EH1 2QL*

E-mail: cr@igs-pla.net

Most range data registration techniques are variants on the *iterative closest point* (ICP) algorithm, proposed by Chen and Medioni [2] and Besl and McKay [3]. That algorithm, though, is only one approach to optimizing a least-squares point correspondence sum proposed by Arun et al [1]. In its basic form ICP has many problems for example, its reliance on pre-registration by hand close to the global minimum and its tendency to converge to sub-optimal or incorrect solutions.

This paper reports on an evolutionary registration algorithm which does not require initial pre-alignment and has a very broad basin of convergence. It searches many areas of a registration parameter space in parallel and has available to it a selection of evolutionary techniques to avoid local minima which plague both ICP and its variants.

*Key Words:* range data registration, evolutionary algorithms, parallel processing

## 1. INTRODUCTION AND PREVIOUS WORK

The majority of range data sampling hardware presents the user with a so-called $2\frac{1}{2}$ dimensional image with a regular sampling grid of depth measurements, either orthogonally or spherically scanned. The function of registration is the fusion of these images to form a 3D data cloud of the scanned object.

This data fusion is often posed as the optimization of a least squares sum, such as that proposed by Arun et al [1](shown in equation 1) and there are several ways to solve this kind of sum, either non-iteratively [5] or iteratively [4, 1, 2, 3]. The best known and most used method is the *iterative closest point* algorithm [2, 3], which was proposed as long ago as 1991 and is still in constant use and development.

Given two partially overlapping sets of range data (technically they should be subsets of the same dataset) and an initial (possibly hand-aligned) estimate of their relative positions, ICP will iteratively improve the transformation estimation. This

[1]Supported by EPSRC grant GR/M97138/01

is done by determining point-to-point correspondences between the data sets and then minimizing their least squares sum. Iterations of the method continue until this sum falls to some threshold value or simply stops changing. It is generally accepted that ICP is a 'hands-on' process, needing both initial data pre-alignment and careful scrutiny as it progresses. A good review of work in this field and the many ICP variations is given by Goshtasby [6].

ICP is a multi-part algorithm with major parts as follows :

1. *Pre-alignment.* Without pre-alignment, variants of ICP will find local minima.

2. *Selection.* This is invariably a pre-process to determine the points selected from each dataset.

3. *Point matching.* An attempt is made to match the selected points in one dataset to the other.

4. *Weighting.* A weight is then assigned to the point pair.

5. *Rejecting.* Points which are deemed unsuitable are rejected. Often this phase is used to reject outliers and to make the algorithm robust to some extent. A common rejection metric is to reject two points if the distance is greater than some tolerance.

6. *Error metric calculation.* ICP in its basic state uses an unweighted sum of the minimum distance from all points in one dataset to all points in the other set. There are, however, many variations on this scheme.

7. *Optimization.* This sum is then minimized in some way. In ICP, a set of conjugate pairs is obtained from the closest point evaluation and the absolute orientation problem is solved using them. The point sets are not true conjugate pairs, so an approximation to the true transformation is obtained. The transformation is then applied and this scheme is then iterated through steps 2 to 7.

Efficient variants of ICP, addressing the speeding-up of these parts, have been summarised by Rusinkiewicz and Levoy [7]. They detail many of the ICP variants currently available and compare several different strategies for the above processes. We cannot detail all of the different variations here, so we would direct the reader to that paper. At the heart of ICP there is a computationally expensive $O(n^2)$ nearest-neighbour calculation which is often a candidate for speed-ups.

A generic speedup method for any optimization is generally parallelization and recently this has been elegantly done using pICP as proposed by Langis, Greenspan and Godin [8]. In their method, the final ICP calculation is performed in parallel and has the advantage that there is approximately n-times speedup for n processors (for up to 18 nodes was reported). Essentially though, it has the original ICP methodology at its core. In this paper, we address one of ICPs major problems, the fact that the error space has many local minima and if the algorithm is not started in the vicinity of the global minimum, it will either not converge or will converge to the nearest local minimum. A good discussion of this behaviour is found in [10] where they use a simulated annealing method to try and address this problem. There have been efforts to automate pre-alignment [20] and widen the basin of convergence [21] but no general purpose approach to hands-free registration exists.

We propose an alternative registration algorithm based on a pose-space search (as opposed to ICP which is a correspondence based search). The algorithm treats registration as a multi-point search optimization and offers an evolutionary alternative to ICP which has the following features:

- *Broad basin of convergence.* Given a wide variety of intial conditions, the algorithm will converge to the global minimum, as detailed in section 3.6.
- *Robust.* It is at least as robust as simulated annealing to both Gaussian and salt-and-pepper noise.
- *Many search points,* as many as there are chromosomes in the population.
- *Parallel in execution.*
- *Modular.* The least-squares evaluation function is able to use any of the efficient variants of ICP-style metrics, especially in the nearest-neighbours calculation.
- *Hands-free.* Since it is built on an evolutionary method, it can use any of a multitude of evolutionary algorithm (EA) operators to accelerate convergence. Also, there are methods to reduce speciation, or convergence to local minima, available.

In general, we have the advantages of the least-squares metric and evolutionary optimization, without the disadvantages of using a single search point in our solution space.

## 2. ALGORITHM

Evolutionary algorithms require efficient representations and fast evaluation functions. The representation we use for the registering transformation is a three parameter rotation vector and a three parameter translation vector. Our evaluation function incorporates an efficient version of a nearest-neighbours search.

### 2.1. Representation

The 'axis-angle' form is used for the representation of rotations. The data is rotated around the vector $\mathbf{n}$, specified by the pair $(\theta, \phi)$, by an amount $\rho$. The data translation is represented by the triplet $\vec{t} = (T_x, T_y, T_z)$ relative to the origin. The chromosomes used in the evolutionary search then consist of the six parameter vector $< \theta, \phi, \rho, \vec{t} >$. Singularities and discontinuities in the pose space are dealt with in the mutation operators, since this is the only place that new information is generated. For example, when a rotation gene reaches a boundary, it is 'barelled' rather than jumoing to the opposite side of it's domain. Translations are bounded by presetting their search domains.

### 2.2. Algorithm

Our registration algorithm is built around a basic evolutionary optimization core. Since it operates in parallel, there are two different parts which operate on the master process and slave process, shown in *algorithm 1 and 2*, below. Since the time for chromosome evaluation is long compared to the transmission latency of our hardware, we keep the entire population of chromosomes on one node and use the others as slave processors which only perform evaluations.

To begin with there is an initialisation of parameter chromosomes. In our scheme this is done randomly although this can be done in a directed way if necessary. For obvious reasons, we have a population which is a multiple of the number of slave processors. The chromosomes are then sent to the slave processes for evaluation and the fitness is returned. A new population is generated using an adapted crossover and mutation suitable for our particular chromosomes, as detailed in section 2.2.3. These three last steps are repeated until a given error level is achieved or the maximum number of population iterations is reached.

In the slave node, there is an initialization phase where both of the registration data sets are read and have their centroids translated to the origin. (This is done simply to reduce the domain for translations and not for some algorithmic purpose.) From this point on, the slaves simply receive chromosomes, unpack them into rotations and translations, perform those operations and then evaluate the fitness function detailed in the section 2.2.1.

Algorithm 1 (Parallel evolutionary registration - master process).

**begin**
  initialize chromosome set, $C$
  send chromosomes to processors (round robin)
  receive evaluations of chromosomes
  sort C into descending order
  stopped :=FALSE
  **while** !stopped
    generate new population
      compute reproduction probabilities
      perform selection of reproducing chromosomes
      perform reproduction and mutation
    send chromosomes to processors (round robin)
    receive evaluations of chromosomes
    sort C into descending order
    **if** end conditions met
      stopped:=TRUE
      send terminate signal to slaves
    **endif**
  **endwhile**
**end**

Algorithm 2 (Parallel evolutionary registration - slave process).

**begin**
  stopped :=FALSE
  read data files for $S_1$ and $S_2$
  translate data centroids to the origin
  **while** !stopped
    receive chromosome $C$
    interpret $C$ as transformation
    **if** $C$ is not valid, stopped:=TRUE
    **else**
      apply $C$ to set $S_2$ to get $S_2'$
      compute evaluation function, $E = f(S_1, S_2')$

send $E$ to master process
**endif**
**endwhile**
**end**

*2.2.1. Evaluation function*

We use the Arun [1] evaluation function. This has the advantage that many of the more robust variations of the optimization function are equally usable in the same context. In particular, we are aware of the variation of the scheme proposed by Zhang [11], which is often used. We use an adaptive spherical distance-window in our selection strategy, which has proved to be adequate.

We seek a translation vector $\mathbf{t}$ and a rotation vector $q_R$ to minimize :

$$E = \frac{1}{N}\sum_{i=0}^{i=N} ||\mathbf{x}_i - \mathbf{t} - \mathbf{R}\mathbf{p}_i||^2 \tag{1}$$

where $\mathbf{p_i} \in S_2$, $\mathbf{R}$ is the rotation matrix generated from our parameter vector $q_R$ and $\mathbf{x}_i$ is the closest point (i.e. nearest neighbour) to the transformed position of $\mathbf{p}_i$ in $S_1$. This evaluation function gives us a good indication of the convergence of the algorithm at runtime. Clearly, there is a nearest neighbour calculation for every point in $S_1$ at every iteration.

*2.2.2. Nearest neighbour calculation*

There are many efficient methods of nearest neighbour calculation, including recently one specifically for ICP [12] which is apparently up to 500% faster than the equivalent k-d tree. We use a reference-point method which is essentially the same as TINN, as described in [9]. We cut down the nearest-neighbour search by first finding the point distance to one of a reference set whose neighbours we already known. There are many ways of performing this search (as outlined in [19]), each with its own characteristics.

*2.2.3. Operators*

We use the two standard forms of evolutionary operators in our algorithm, mutation and crossover.

- Random in-domain mutation, where a gene is randomized inside a domain.
- Local search mutation, where a gene is changed based on a local search.
- Boundary mutation, where a gene is pushed to a limit of its domain.
- Creep mutation, where a gene is slightly incremented or decremented by a preset amount.
- Crossover by rotation, where a pair of chromosomes is crossed by their rotation parameters.
- Crossover by translation, where a pair of chromosomes is crossed by their translation parameters.

Each of the operators has a probability of selection associated with it, stored in a normalised unit vector. The probability of selection is increased each time the operator yields a better offspring than either its parent (in the case of mutation), or its parents (in the case of crossover). Crossover chromosomes are selected by tournament selection.

*2.2.4. Termination conditions*

The termination conditions for the EA are based on the following criteria (in order of importance):

• *Maximum number of iterations.* If this is reached then the algorithm is terminated.

• *First derivative of the error curve.* When this gets close to zero the algorithm is terminated.

• *Absolute error assessment of best chromosome.* The limit for this can be set intially and then termination is based on that preset. The theoretical minimum is relatively easy to approximate if the degree of overlap and interpoint distance is known before hand.

## 3. EXPERIMENTS AND RESULTS

### 3.1. Hardware and libraries

We should point out that this approach benefits significantly from parallel hardware. Evolutionary algorithms are particularly well suited to *Beowulf*-style hardware [15], where although inter-processor communication is fairly low-bandwidth, CPU power is high. We use an array of 11 × 1GHz Athlon computers connected by a 100Mbps ethernet hub. Each machine has 512Mb of RAM and runs the GNU operating system, built on the Linux 2.4.2 kernel. In order to run the machine as an evolutionary workbench we have developed an evolutionary algorithm using chromosome distribution software. This was built as additional libraries to MPI (*EasyMPI* [14]).

### 3.2. Experimental setup

Since we have 11 nodes in our parallel machine, shown in figure 2, one of which acts as the master process, it makes sense to use chromosome populations that are multiples of 10. It should be remembered that these chromosomes can also be thought of as search points in the solution space, which gives us the ability to search much more of the space in the same time.

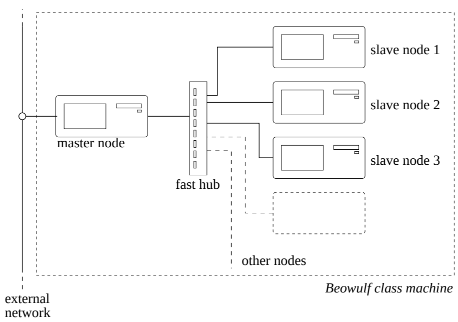



**FIG. 1.** Experimental setup

### 3.3. Artificial Data

Many sets of artificial data were examined in both 2D and 3D. We present here the two most interesting ones. Both sets are **pathological cases** for most variations of ICP when presented without very good initial pre-alignment since the registration possibilities are enormous, meaning that local minima abound in the solution space. The first set is two sawtooth patterns with ten peaks in each. This allows many possible local minima but only one global minimum. The second set is two $sin() \times sin()$ functions presenting many peaks and many possible local minima for registration.

Results are shown in table 1.

**TABLE 1**

**Artificially generated data registration results (10 runs)**

| Data set | No. points | Added noise SD | Populations | Mean point distance after registration (SD) |
|---|---|---|---|---|
| sawtooth | 9000 | 0.1 units | 50 | 0.29 (0.14) units |
| sin(x)*sin(y) | 20200 | 0.1 units | 61 | 0.30 (0.13) units |

### 3.4. Real Data

For real data we have used data sets from four different objects:

1. A block of drilled steel with spheres placed in some of the drilled holes. This object is useful because there is a large plane at the top of the block which can be used to visually assess the quality of the registration.
2. A small plastic toy cow, from which three scans have been taken with approximately 75% overlap between each scan. Results are shown in table 2.
3. An asymmetric corner of a prism with 75% overlap
4. A rescaled version of the Stanford bunny with data taken at $0^o$ and $45^o$

**TABLE 2**

**Scanned data registration results (10 runs)**

| Data set | No. points | Populations | Mean inter-point distance after registration (SD) |
|---|---|---|---|
| Block with holes and spheres | 5655 | 175 | 0.44 (0.02) mm |
| Toy Cow no.1 and 2 | 13971 | 175 | 0.60 (0.08) mm |
| Toy Cow no.1 and 3 | 13561 | 175 | 0.91 (0.14) mm |
| Asymmetric corner | 14028 | 200 | 0.00 (0.00) mm |
| Stanford bunny | 80353 | 200 | 0.086 (0.009) mm |

### 3.5. Timing Issues

Timings are heavily dependent on number of points in each data set. For comparison, we show in table 3 the evaluation function time given the number of points in the datasets. Note that each point in the second dataset is compared with each point in the first. Given our equipment setup, there are some start-up latencies (the order of micro seconds) associated with the process initialization but these are generally small when compared to the evaluation function time. If low numbers of points are used, however, they can play some part in overall processing times.

Approximate population iteration times are also given for a fairly normal population of 154 chromosomes. The number of population iterations is naturally dependent on initial alignment and fortuitous mutations and initial random starting sets but would rarely have to exceed 100 populations.

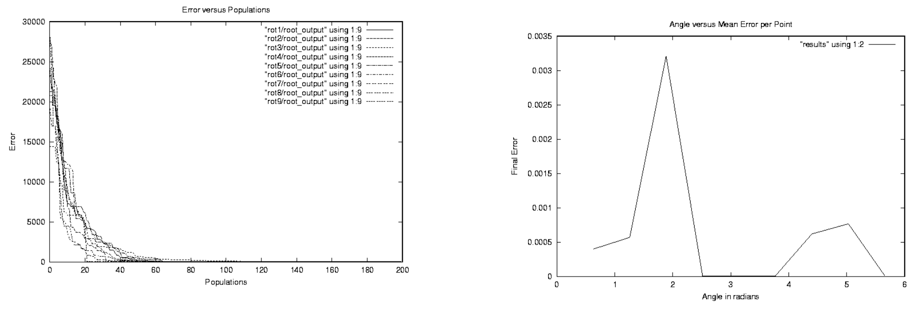
**TABLE 3**

**Evaluation times for different point set sizes**

| Points in each data set | Time for evaluation (seconds) | Typical population iteration time (seconds) |
|---|---|---|
| 10100 | less than 0.01 | 0.2 |
| 40100 | 5.0 | 110 |
| 82500 | 11.0 | 242 |

### 3.6. Algorithm Convergence

In order to test the basin of convergence for our algorithm, we took two pairs of data sets and rotated the intial conditions in steps through $360^o$ around the Z axis to examine the differences in convergence times and the differences in final mean inter-point errors. The convergence graphs for each of the data sets is shown in figures 2a and 3a. A graph of final error against rotation angle is shown in figures 2b and 3b.

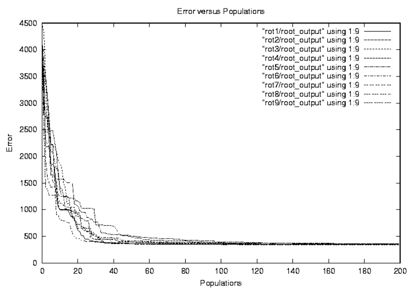



**(a)** Error Convergence for Asymmetric Corner Data with Rotated Start Positions

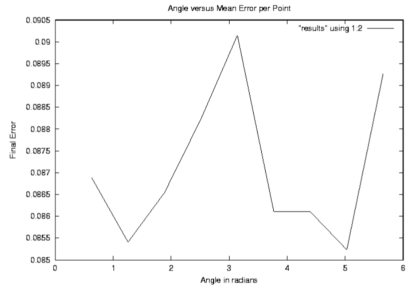
**(b)** Mean Inter-Point Error after Convergence

**FIG. 2.** Results for rotated starting positions: Asymmetric corner dataset

As can be seen from the graphs, the rotations in the registration data set make little difference to either the rate of convergence or the final mean inter-point distances for either pair of data sets.




**(a)** Error Convergence for Stanford Bunny Data with Rotated Start Positions

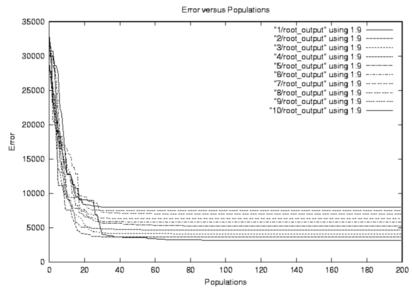



**(b)** Mean Inter-Point Error after Convergence

**FIG. 3.** Results for rotated starting positions: Stanford bunny dataset

### 3.7. The Effect of Salt-and-Pepper Noise

Evenly distributed Gaussian noise on all three measurement components has little effect on the convergence in ICP. Salt-and-Pepper (S+P) noise over the measurement domain, however, represents a significant difficulty since all of the noise points are outliers and can contribute heavily to any error metric. This can have the effect of skewing the convergence position of one data set with respect to the other. To examine the effect of S+P noise we took one pait of data sets and added noisy points between 1% and 10% of the data points.




Final registration error versus populations for S+P data sets

The graph shows that convergence to the best available minimum is achieved in every case. The final error is naturally higher in case as the noise is deliberately generated inside the spherical window that we use as our cutoff. This clearly shows that the algorithm is very robust to S+P noise. We used randomly generated noise whose domain wqs within the bounding box of the test data.

## 4. COMPARISON WITH ICP

All of the experiments in the previous sections were also performed with a 'vanilla' version of ICP with the same random initialisation rotations and translations. None of the results shows a convergence to the global minimum although some do successfully find bad local minima.

It is interesting to note, however, that once a good convergence for our algorithm in established, running ICP afterwards will decrease the mean inter-point error, although often at the cost of other points which are treated as outliers and discarded from the final registered data set. Results are shown in table 4.

**TABLE 4**
**Ordinary ICP results on the same data. NC=no convergence, LM= local minimum found.**

| Data set | Converged mean inter-point distance after registration | Quality |
|---|---|---|
| sawtooth | - | NC |
| sin(x)*sin(y) | 0.429 units | LM |
| Block with holes and spheres | 0.2mm | LM, 90% deleted |
| Toy Cow no.1 and 2 | 25 mm | LM |
| Toy Cow no.1 and 3 | 33 mm | LM |

## 5. CONCLUSIONS

We have demonstrated an alternative pose-space searching method for registration that requires no data pre-alignment and has the ability to find a good approximation to the global minimum in the least-squares closest point error space. The algorithm uses the best features of both ICP and evolutionary methods, giving it both a robust evaluation function and a robust search optimization capability.

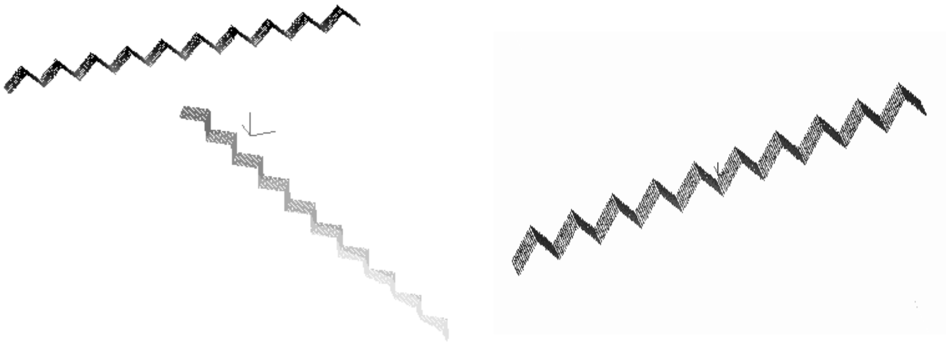
In this paper, we have used the bare minimum of algorithmic complexity to produce a functional method. In future work, we will employ a more robust evaluation function and employ more evolution operators both for mutation and crossover, in order to speed up the convergence. There is also room for speeding up the evaluation function, using the enhancements mentioned in the text in the relevant sections. If this were done, more chromosomes could be used in the populations, having the knock-on effect of improving convergence speed.

We also intend to expand this method to register multiple datasets simultaneously by reducing a globally assessed evaluation function.

## APPENDIX: EXAMPLE RANGE IMAGES AND REGISTERED DATA SETS

### Artificial Data

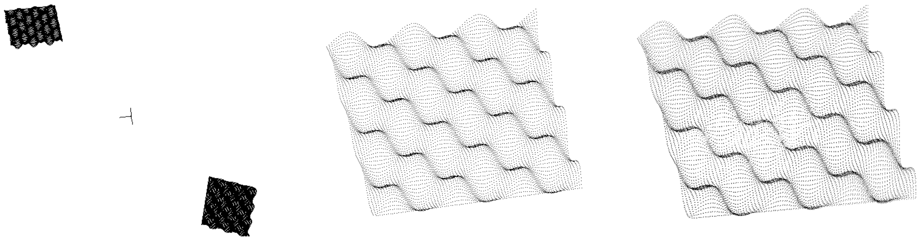
The unregistered artificial data referred to in section 3.3 are shown in figure A1(a) and figure A2(a). The registered versions are shown in figures A1(b) and figures A2(c). Figure A2(b) shows a close-up view of the $sin(x) \times sin(y)$ function, which illustrates the potential for local minima in the error function. There are, however, only two equal global minima.



(b)

(a)

**FIG. A1.** Sawtooth data **(a)** initial random orientation and **(b)** after registration



(a) (b) (c)

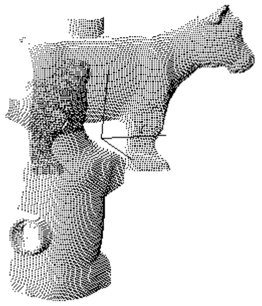
**FIG. A2.** Product of sin() functions **(a)** initial random orientation and **(b)** close-up of single data set and **(c)** after registration

### Scanned Data

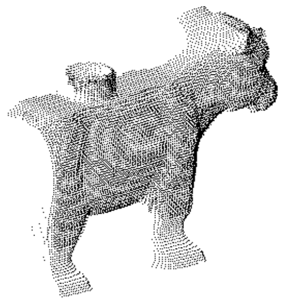
The unregistered scanned data referred to in section 3.4 is shown in figures 3(a) and 4(a). There are three scans of a toy cow from various angles and a block drilled with large holes with some spheres placed on it. Registered versions are shown in A3(b) and A4(b).

(a) (b)

**FIG. A3.** Scanned hole-block data **(a)** as scanned and **(b)** after registration



(a)



(b)

**FIG. A4.** Scans of a toy cow **(a)** 1 and 2, in random orientation **(b)** after registration

## REFERENCES


1. K. S. Arun, T. Huang and S. D. Blostein (1987), "Least-squares fitting of two 3-d point sets", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.9, no.5, pp 698-700.
2. Y. Chen and G. Medioni (1991), "Object modeling by registration of multiple range images", *Proceedings of the IEEE Conference on Robotics and Automation*, 1991.
3. P. J. Besl and N. D. McKay (1992), "A method for registration of 3D shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.14, no.2, pp 239-256.
4. T. Huang, S. D. Blostein and E. A. Margerum (1986), "Least-squares estimation of motion parameters from 3D point correspondences", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, pp 198 -200.
5. O. Faugeras and M. Hebert (1983), "A 3d recognition and positioning algorithm using geometrical matching between primitive surfaces", in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp 996-1002.
6. A. Goshtasby (1998), "Three-dimensional model construction from multiview range images; survey with new results", *Pattern Recognition*, Vol.31, No.11, pp.1705-1714.
7. S. Rusinkiewicz and M. Levoy (2001), "Efficient variants of the ICP Algorithm", *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, Quebec City, Canada, pp 145-152, IEEE Computer Society.
8. C. Langis, M. Greenspan and G. Godin (2001), "The parallel iterative closest point algorithm", *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, Quebec City, Canada, pp 195-202, IEEE Computer Society.
9. M. Greenspan, G. Godin and J. Talbot (2000) "Acceleration of Binning Nearest Neighbor Methods", National Research Council for Canada report 44167 and published in *Proceedings of Vision Interface 2000*, Montreal, Quebec, May 2000.
10. J. P. Luck, W. A. Hoff, R. G. Underwood and C. Q. Little (2000), "Registration of range data using a hybrid simulated annealing and iterative closest point algorithm", *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*. Available from **http://egweb.mines.edu/whoff/publications/2000/pami2000.pdf**
11. Z. Zhang (1992), "Iterative point matching for registration of free-form curves", INRIA research report No. 1658.
12. M. Greenspan and G. Godin (2001), "A nearest neighbour method for efficient ICP", *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, Quebec City, Canada, pp 161-168, IEEE Computer Society.
13. G. Godin, D. Laurendeau and R. Bergevin (2001), "A method of registration of attributed range images", *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, Quebec City, Canada, pp 179-186, IEEE Computer Society.
14. **http://www.dai.ed.ac.uk/homes/craigr/EasyMPI**
15. http://www.beowulf.org

16. D. B. Fogel (1994), "An introduction to simulated evolutionary optimization", in *IEEE Transactions on Neural Networks*, Vol.5, No.1, pp3-14.
17. **http://www.cs.rpi.edu/~hollingd/ga/students/taube/sharing.html**
18. P. J. Darwen and Xin Yao (1996), "Every Niching Method has its Niche: Fitness Sharing and Implicit Sharing Compared", *Proceedings of Parallel Problem Solving from Nature – PPSN IV.*
19. M. Smid (2001), "Closest-Point Problems in Computational Geometry", To appear in: *Handbook on Computational Geometry*, edited by J.-R. Sack, North Holand, Amsterdam.
20. S.M. Yamany, M. N. Ahmed, and A. A. Farag (1998), "Novel Surface Registration Using The Grid Closest Point (GCP) Transform," Proc IEEE, Int. Conf. on Img. Proc., ICIP, vol 3, pp 809-813, Chicago.
21. A. W. Fitzgibbon (2001), "Robust Registration of 2D and 3D Point Sets", Proc. British Machine Vision Conference BMVC01, Manchester, pp 411-420, September 2001.
22. **http://www.dai.ed.ac.uk/homes/craigr/Registration**