Evolution-Based Methods for Selecting Point Data for Object Localization: Applications to Computer-Assisted Surgery

Shumeet Baluja[1] and David A. Simon[2]

November 1, 1996
CMU-CS-96-183

[1]Department of Computer Science [2]The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

Object localization has applications in many areas of engineering and science. The goal is to spatially locate an arbitrarily-shaped object. In many applications, it is desirable to minimize the number of measurements collected for this purpose, while ensuring sufficient localization accuracy. In surgery, for example, collecting a large number of localization measurements may either extend the time required to perform a surgical procedure, or increase the radiation dosage to which a patient is exposed.

Localization accuracy is a function of the spatial distribution of discrete measurements over an object when measurement noise is present. In [Simon et al., 1995a], metrics were presented to evaluate the information available from a set of discrete object measurements. In this study, new approaches to the discrete point data selection problem are described. These include hillclimbing, genetic algorithms (GAs), and Population-Based Incremental Learning (PBIL). Extensions of the standard GA and PBIL methods, which employ multiple parallel populations, are explored. The results of extensive empirical testing are provided. The results suggest that a combination of PBIL and hillclimbing result in the best overall performance. A computer-assisted surgical system which incorporates some of the methods presented in this paper is currently being evaluated in cadaver trials.

Shumeet Baluja was supported by a National Science Foundation Graduate Student Fellowship and a Graduate Student Fellowship from the National Aeronautics and Space Administration, administered by the Lyndon B. Johnson Space Center, Houston, TX. David Simon was partially supported by a National Science Foundation National Challenge grant (award IRI-9422734).

Keywords: object localization, surface-based registration, evolutionary optimization, genetic algorithms, population-based incremental learning.

1 Introduction

Object localization is the problem of determining the spatial pose (position and orientation) of an arbitrarily-shaped object. Localization requires that a geometric model of the object be matched to measurements of the object acquired while the object is in a fixed configuration. In the current work, the model is a 3-D surface representation and the measurements are discrete 3-D surface points. The matching process (often referred to as *registration*) is to find a spatial transformation which aligns the measurements with the model, as measured by a suitable cost metric. For example, Figure 1 shows data from a physical model (or “phantom”) of a human pelvis before and after registration. The shaded triangle mesh surface model was constructed from computed tomographic (CT) images of the pelvis phantom, and the spheres represent discrete 3-D surface measurements which were collected in selected regions of the same phantom using a digitizing probe. Once the registration process is performed, it provides a mapping between any point in the CT coordinate system and the corresponding point in the digitizing probe coordinate system, and vice versa. In surgical applications, this would allow a pre-operative plan constructed from CT data to be executed in a coordinate system related to the surgical tools or patient (i.e., the digitizing probe’s coordinate system). Solution of similar registration problems is required in a wide variety computer assisted surgical techniques [Taylor et al., 1995].

Before:

After:

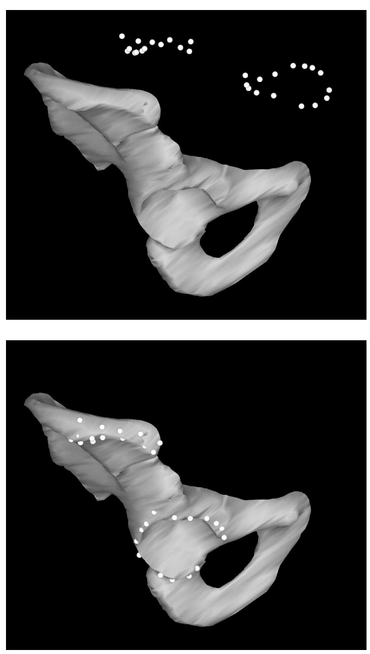

Figure 1: Registration of discrete measurements to a 3-D surface model of a human pelvis

In surgical applications, each data measurement can increase the time and risk associated with the surgical procedure. In order to minimize the number of measurements required, the collection of measurement data should be carefully planned to ensure that the resulting object localization is sufficiently accurate. The problem of selecting sets of discrete point measurements which have maximum utility for localization is difficult for at least three reasons. First, the space is highly discontinuous and contains many locally optimal point sets. Second, it is difficult to define meaningful derivatives in the space of candidate measurement points. Third, the space can be large with many dimensions; exhaustively exploring point configurations is far beyond computational resources. Initial attempts to find good point sets were based on steepest ascent hillclimbing methods [Simon et al., 1995a]. Recently, methods such as genetic algorithms have been proposed for combinatorial optimization problems [Homaifer et al., 1993], [Levine, 1993], [Fang et al., 1993]. In this paper, alternate hillclimbing methods, genetic algorithms (GAs), and a GA hybrid, termed Population Based Incremental Learning (PBIL), are applied to this problem.

2 Discrete Point Data Selection

The goal of the registration problem in Figure 1 can be stated mathematically as follows:

$$\min_{\boldsymbol{R},\boldsymbol{T}} \sum_i \|M_i - (\boldsymbol{R}D_i + \boldsymbol{T})\|^2 \quad (1)$$

where M_i represents points in the triangle mesh surface model, D_i represents the discrete point measurements, and $\boldsymbol{R}$ and $\boldsymbol{T}$ are a rotation and translation respectively, which minimize the least-squares distance between the points. In this paper, it is assumed that the scale of the model and data measurements are the same, although in general it is possible to estimate an unknown scale parameter in the registration process. A solution method for this class of problems was proposed in a paper describing the Iterative Closest Point (ICP) algorithm [Besl and McKay, 1992]. In the first step of this algorithm, the closest points between each D_i and the surface model are computed and labeled M_i. In the second step, equation (1) is solved using a least-squares solution method such as the one described in [Horn, 1987]. The resulting transformation is then applied to each of the D_i. This process of computing closest points and solving the resulting least-squares problem is repeated iteratively until convergence. While convergence to the *global* minimum is not guaranteed, in practice the algorithm does converge globally when the initial pose estimate is good.

In certain situations, there is a fundamental ambiguity when attempting to solve the registration problem [Simon et al., 1995a]. For example, consider the problem of localizing the slotted cylinder shown in Figure 2 using only discrete point measurements in locations indicated by the marks. It is impossible to localize the cylinder translationally along the central axis or rotationally about it. For this particular measurement set, there is a singularity in equation (1) with respect to $\boldsymbol{R}$ and $\boldsymbol{T}$. An infinite number of pose solutions exist. However, by repositioning several of the discrete measurement points to the end regions and within the slot, it becomes possible to determine the location of the cylinder unambiguously.

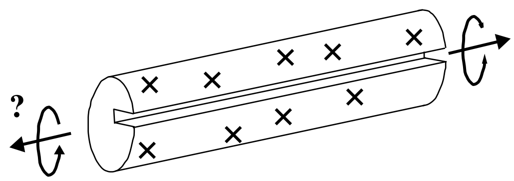

Figure 2: Slotted cylinder singularity

A similar example is shown in Figure 3 in which the goal is to localize each of the cubes using the discrete point measurements indicated by the dots. Assuming that the data in each case is corrupted by unbiased noise, it can be shown that the resulting registration accuracy is a function of the number and positions of the points which are used. This is demonstrated in Figure 4, in which the normalized pose error resulting from 1000 registration experiments is plotted for each of the measurement configurations of Figure 3. The configuration with the largest number of points (128) results in the best overall accuracy (i.e., the smallest error).

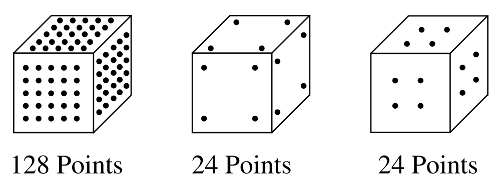

Figure 3: Cube data configurations

In Figure 4, the pose errors resulting from the two 24 point measurement configurations are not the same. Given the choice between these two configurations, the one with the points nearer the cube corners is clearly superior. Note that if the registration experiments had been performed without measurement noise, all three configurations would result in the same error. In the absence of noise, far fewer points would be necessary; the theoretical minimum number of points required for registering discrete points to an arbitrary surface is six. Additional insight into the results of Figure 4 is provided in the discussion below.

As suggested above, for certain applications the costs associated with collecting object localization measurements can be high. Therefore, it is desirable to minimize the amount of data acquired, while ensuring that accuracy requirements are satisfied. Furthermore, it is necessary to ensure that singularities such as the one illustrated in Figure 2 do not arise, even for

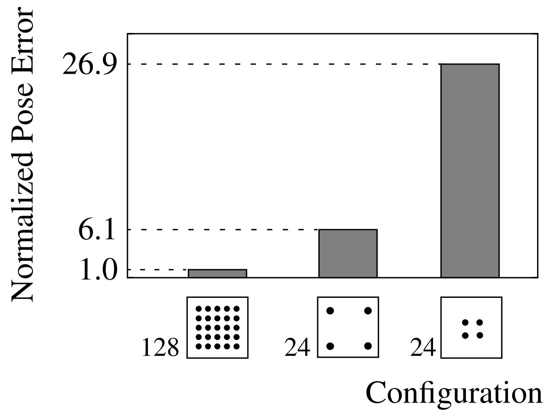

Figure 4: Configuration dependent cube pose error

arbitrarily-shaped objects such as the pelvis in Figure 1. The goal of discrete point data selection is to select a fixed number of discrete point measurements which result in the highest possible localization accuracy.

In [Simon et al., 1995a] a framework is proposed for analyzing the "geometric constraint" between a surface model and a set of discrete points. A complete formulation of the analysis is beyond the scope of this paper; however, a brief summary is presented here.

Consider the following equation which is presented without derivation (for additional details see [Simon et al., 1995a]).

$$E = \sum_{i=1}^{6} \lambda_i (\boldsymbol{dt}^{\mathrm{T}} \boldsymbol{q}_i)^2 \tag{2}$$

In the equation, E is a first order approximation to the least-square error term in equation (1), the λ_i are a set of scalar eigenvalues, $\boldsymbol{q}_i$ are the corresponding 6-dimensional eigenvectors, and $\boldsymbol{dt}$ is a 6-dimensional vector which represents an arbitrary, small 3-D transformation. Given an object surface model and a configuration of proposed discrete point surface measurements, the λ_i and $\boldsymbol{q}_i$ of equation (2) can be derived. The $\boldsymbol{q}_i$ define a set of basis vectors which span the space of all possible transformations. The λ_i can be interpreted as sensitivity measures (or gains). They express the rate of change in E which results when a set of discrete point measurements is transformed about the global minima of equation (1) in the direction $\boldsymbol{q}_i$. For example, if one or more of the λ_i are equal to 0, a singularity such as the one demonstrated in Figure 2 exists. The corresponding $\boldsymbol{q}_i$ represents the direction in which the singularity occurs. In general, it is desirable for all of the λ_i to be as large as possible so that E is maximally sensitive to all transformations. The following scalar sensitivity measure is used in the remainder of this work:

$$\frac{\lambda_6^2}{\lambda_1} \tag{3}$$

where λ_6 corresponds to the smallest eigenvalue and λ_1 corresponds to the largest eigenvalue. The intuition behind this measure is that maximizing it makes the smallest eigenvalue large, while ensuring that the variability in the eigenvalues is small. A discussion of this measure and several alternatives can be found in [Navhi, 1994] and [Kim and Khosla, 1991].

Recall that equation (2) is a first order approximation of the error term in equation (1). Therefore, we would expect a relation between the sensitivity of the former equation and the accuracy of registration performed using the latter equation. This relation is illustrated in Figure 5. The normalized pose errors of Figure 4 are plotted versus the sensitivity measure of equation (3). Note the strong relation between pose error and the sensitivity measure. Larger values of the sensitivity measure result in smaller pose errors. This relationship allows equation (3) to be used as a criterion function for selecting discrete point measurement configurations which tend to result in high registration accuracy.

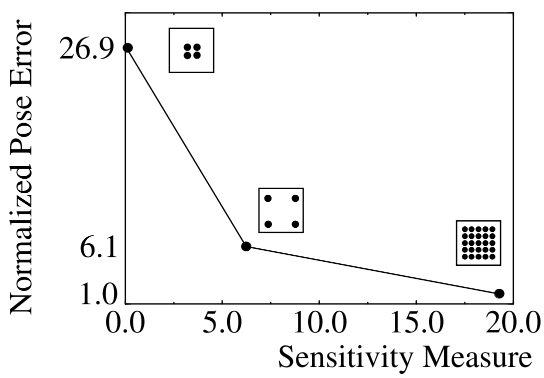

Figure 5: Pose error vs. Sensitivity

Each pose error along the Y-axis of the plot in Figure 5 required 1000 registration trials to be performed from random initial starting locations. Execution time to estimate this pose error was roughly 60 minutes per datum on a Sun Sparc-10 workstation. In contrast, each sensitivity measure along the X-axis was computed in less than 1 milli-second per configuration. Due to this disparity in computation time, it is much more efficient to use the sensitivity measure as a criterion function for selecting discrete point measurement configurations via optimization methods.

The geometric constraint analysis summarized above provides a global metric of registration measurement sensitivity. Local measures of sensitivity based on features such as surface curvature are also feasible; however, such measures are not guaranteed to completely constrain the pose. In particular, there are situations in which collecting measurement data only within high curvature regions will result in singularities such as the one evident in the slotted cylinder of Figure 2. Identifying such singularities requires a global measure of sensitivity such as the one proposed above.

3 Search Algorithms

Selecting measurement points which maximize the sensitivity measure in equation (3) is a combinatorial search problem. The goal is to select **N** points from a set, **V**, such that equation (3) is maximized. In practice, the points in set **V** are the vertices of the triangle mesh surface model. For this task, many search algorithms are considered: two hillclimbing methods, many variants of genetic algorithms, and Population-Based Incremental Learning. These methods are described in this section.

3.1 Steepest Ascent Hillclimbing

The first type of hillclimbing is Steepest Ascent Hillclimbing. Initially, **N** vertices are randomly chosen from the set of possible vertices, **V**. Call this selected set **S**. Let **Evaluate(S)** represent the performance of **S** (from equation (3)) which is to be maximized. Select a vertex **v** from **S**. In turn, replace **v** with every vertex in **V**. Repeat this procedure for each **v** in **S**. Replace the vertex in **S** with the single substitution which leads to the largest increase in performance. Repeat the process. Continue until no better moves are found. This is the method which was initially used for the point selection experiments in [Simon et al., 1995a].

3.2 Next-Ascent Hillclimbing

The second form of hillclimbing is Next-Ascent Hillclimbing. This method is similar to steepest ascent, except that the set **S** is immediately changed when a vertex substitution is found which increases **Evaluate(S)**. This is in contrast to the previous method, in which the substitution chosen maximizes **Evaluate(S)** over all possible substitutions. Next-Ascent Hillclimbing usually takes more steps than the previous method, but there are fewer evaluations per step. In addition, vertex substitutions for Next-Ascent Hillclimbing are generated randomly.

3.3 Genetic Algorithms

Genetic algorithms (GAs) are biologically motivated adaptive systems which are based upon the principles of natural selection and genetic recombination. A GA combines the principles of survival of the fittest with a randomized information exchange. It has the ability to recognize trends toward optimal solutions, and to exploit such information by guiding the search toward them. Recently, genetic algorithms have been proposed as general purpose function optimization tools for combinatorial problems such as the Traveling Salesman Problem [Homaifer et al., 1993], Jobshop Scheduling [Fang et al., 1993], Set Partitioning [Levine, 1993] and actuator placement [Furuya and Haftka, 1993], to name a few.

In the standard GA, candidate solutions are usually encoded as fixed length binary vectors. The initial group of potential solutions is chosen randomly. These candidate solutions, called "chromosomes," are allowed to evolve over a number of generations. At each generation, the fitness of each chromosome is calculated; this is a measure of how well the chromosome optimizes the objective function. The subsequent generation is created through a process of selection, recombination, and mutation. The chromosomes are probabilistically selected for recombination based upon their fitness. General recombination (crossover) operators merge the information contained within pairs of selected "parents" by placing random subsets of the information from both parents into their respective positions in a member of the subsequent generation. Although the chromosomes with high fitness values have a higher probability of selection for recombination than those with low fitness values, they are not guaranteed to appear in the next generation. Due to the random factors involved in producing "children" chromosomes, the children may, or may not, have higher fitness values than their parents. Nevertheless, because of the selective pressure applied through a number of generations, the overall trend is towards higher fitness chromosomes. Mutations are used to help preserve diversity in the population. Mutations introduce random changes into the chromosomes. Good overviews of GAs can be found in [Goldberg, 1989] [DeJong, 1975].

One of the tunable parameters in the GA is the type of crossover/recombination used. The crossover operator, which is unique to genetic algorithms, randomly mixes the contents of two "parent" solution strings into two "children" solution strings. There are three common crossover operators; *one point crossover*, *two point crossover*, and *uniform crossover*. Since it is not *a priori* known which crossover will perform better in this search problem, all three were tried. In one point crossover, a random point in the solution strings is chosen, and the material beyond that point is swapped in the children solutions. In two point crossover, two points are chosen, and the material between the two points is swapped. In uniform crossover, each position in the children has a 50% chance of coming from parent A, and a 50% change of coming from parent B. The crossover operators are shown in Figure 6.

The simple single population GAs described above have been extended by using parallel evolution of multiple smaller subpopulations, termed "islands". This idea is motivated by the study of punctuated equilibria, allopatric speciation, and ho-

```
Parent A: XXxxXxX|XxXX      Child A: XXxxXxXYyyy
Parent B: YYyyyYY|Yyyy ---> Child B: YYyyyYYXxXX

              One Point Crossover

Parent A: XXxx|XxX|XxXX      Child A: XXxxyYYXxXX
Parent B: YYyy|yYY|Yyyy ---> Child B: YYyyXxXYyyy

              Two Point Crossover

Parent A: XXxxXxXXxXX        Child A: XYxyyxXXyyX
                       --->
Parent B: YYyyyYYYyyy        Child B: YXyxXYYYxXy

               Uniform Crossover
```

Figure 6: Common crossover operators for genetic algorithms.

meostasis [Cohoon et al., 1988]. These ideas are implemented by evolving multiple smaller subpopulations independently. At infrequent intervals (in this study once every 100 generations), the best solutions between subpopulations are copied, and the evolution is continued. This allows diversity to be re-introduced into each subpopulation by the introduction of new solution strings. Methods based upon this idea have empirically shown to perform substantially better than single-population methods [Gordon and Whitley, 1993] [Davidor, 1991] [Whitley and Starkweather, 1990]. The subpopulations are arranged in a linear order. The swapping scheme is shown in Figure 7; this scheme is taken from [Whitley and Starkweather, 1990].

3.4 Population-Based Incremental Learning

Recently, a simplified statistical model of the GA has been introduced in [Baluja and Caruana, 1995], termed Population-Based Incremental Learning (PBIL). This method has been compared to standard GAs on a variety of benchmarks with promising results [Baluja, 1995]. Like the standard GA, the version of PBIL presented here operates on solutions encoded as binary vectors. A brief introduction to the PBIL method is given below, and the algorithm is shown in Figure 5 (description and figure from [Baluja, 1995]):

> PBIL is a combination of evolutionary optimization and hillclimbing. The object of the algorithm is to create a real valued probability vector which, when sampled, reveals high quality solution vectors with high probability. For example, if a good solution to a problem can be encoded as a string of alternating 0's and 1's, a suitable final probability vector would be 0.01, 0.99, 0.01, 0.99, etc.
>
> Initially, the values of the probability vector are set to 0.5. Sampling from this vector yields random solution vectors because the probability of generating a 1 or 0 is equal. As search progresses, the values in the probability vector

gradually shift to represent high evaluation solution vectors. This is accomplished as follows: A number of solution vectors are generated based upon the probabilities specified in the probability vector. The probability vector is pushed towards the generated solution vector(s) with the highest evaluation. The distance the probability vector is pushed depends upon the learning rate parameter. After the probability vector is updated, a new set of solution vectors is produced by sampling from the updated probability vector, and the cycle is continued. As the search progresses, entries in the probability vector move away from their initial settings of 0.5 towards either 0.0 or 1.0. The probability vector can be viewed as a prototype vector for generating solution vectors which have high evaluations with respect to the available knowledge of the search space.

The manner in which the updates to the probability vector occur is similar to the weight update rule in supervised competitive learning networks, or the update rules used in Learning Vector Quantization (LVQ) [Hertz et al., 1991]. Many of the heuristics used to make learning more effective in supervised competitive learning networks (or LVQ), or to increase the speed of learning, can be used with the PBIL algorithm. This relationship is discussed in greater detail in [Baluja and Caruana, 1995].

To the best of our knowledge, PBIL has only been explored using a single probability vector, as shown in Figure 8; however, multiple probability vectors are also possible. The resulting model is similar to the parallel GA model described in the previous section. The implementation is as follows: several instantiations of the PBIL algorithm, shown in Figure 8, are run independently. The PBIL algorithm is modified to include interactions analogous to crossover in genetic algorithms. Every 100 iterations, a probability vector from another population is copied to the current population. Samples are generated based upon the original and new probability vectors by simulating "one-point" crossover. A random point is cho-

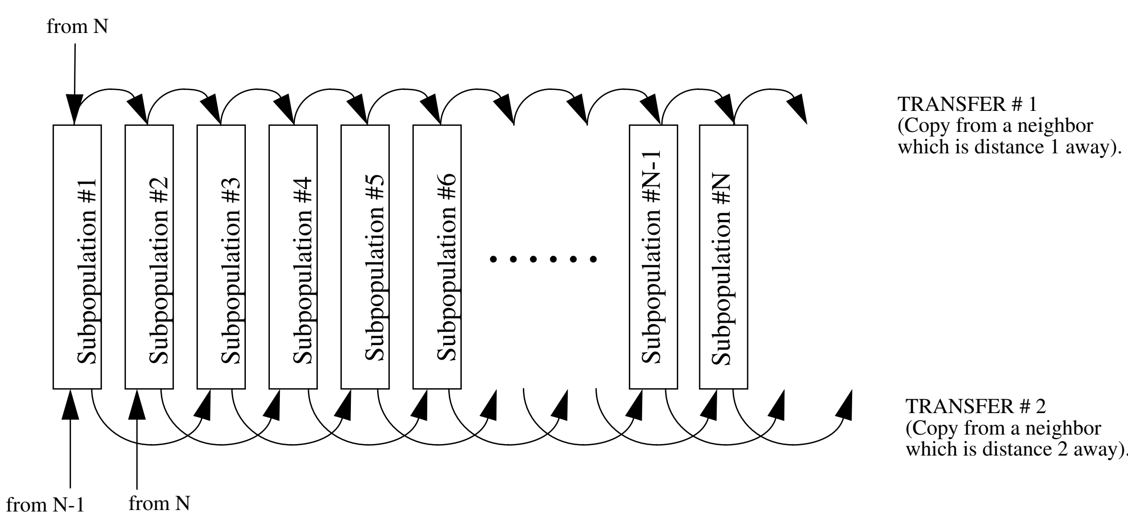

Figure 7: The swapping scheme for parallel GAs. The arrows represent the copying of the best chromosome from the donating population into the receiving population. The receiving population replaces its worst chromosome with the new chromosome. This transfer takes place every 100 generations. The distance of the donating population is the transfer occurrence modulo the number of subpopulations. In these experiments, a total of 10 subpopulations are used.

```
****** Initialize Probability Vector *****
for i :=1 to LENGTH do P[i] = 0.5;

while (NOT termination condition)
    ***** Generate Samples *****
    for i :=1 to SAMPLES do
        sample_vectors[i]:=generate_sample_vector_with_probabilities (P);
        evaluations[i] :=Evaluate_Solution (sample[i]);
        best_vector := best_evaluation (sample_vectors, evaluations);

    ***** Update Probability Vector Towards Best Solution *****
    for i :=1 to LENGTH do
        P[i] := P[i] * (1.0 - LR) + best_vector[i] * (LR);

    ***** Mutate Probability Vector *****
    for i :=1 to LENGTH do
        if (random (0,1) < MUT_PROBABILITY) then
            if (random (0,1) > 0.5) then mutate_direction := 1
            else mutate_direction := 0;
            P[i]:=P[i]*(1.0-MUT_SHIFT)+ mutate_direction*(MUT_SHIFT);
```

USER DEFINED CONSTANTS (Values Used in this Study):
SAMPLES: vectors generated before update of the probability vector (100).
LR: the learning rate, how fast to exploit the search performed (0.1).
MUT_PROBABILITY: probability for a mutation in each position (0.02).
MUT_SHIFT: amount a mutation alters the value in the bit position (0.05).
LENGTH: length of encoded solution (problem dependent).

Figure 8: Single vector PBIL algorithm for binary encoded solution strings (adapted from [Baluja, 1995]).

sen, all bits before the chosen point are determined based upon the original probability vector, and all bits after the chosen point are based upon the new probability vector. The best vectors generated in the population are used to update both of the probability vectors. In this manner, both of the probability vectors converge to the same vector over time. This simulates, at a coarse level, the focusing of search seen in the parallel-GA subpopulations. The generation of solutions based on two probability vectors is shown in Figure 9; this procedure is used in each subpopulation that is evolved. Each subpopulation generates samples from its own two probability vectors. The order in which the probability vectors are swapped is the same as the one used in parallel GA, shown in Figure 7. Other methods of multiple population combination were tried and are currently under study; however, these issue are beyond the scope of this paper.

```
***** Generate Samples With Two Probability Vectors*****
for i :=1 to SAMPLES do
   vector1 := generate_sample_vector_with_probabilities (P1);
   vector2 := generate_sample_vector_with_probabilities (P2);
   crossover_point := random (0, LENGTH);
   for j := 1 to crossover_point do
        sample_vector[i][j] := vector1[j]
   for j := crossover_point to LENGTH do
        sample_vector[i][j] := vector2[j]
   evaluations[i] :=Evaluate_Solution (sample[i]);
   best_vector := best_evaluation (sample_vectors, evaluations);

***** Update Both Probability Vectors Towards Best Solution *****
for i :=1 to LENGTH do
   P1[i] := P1[i] * (1.0 - LR) + best_vector[i] * (LR);
   P2[i] := P2[i] * (1.0 - LR) + best_vector[i] * (LR);
```

USER DEFINED CONSTANTS (Values Used in this Study):
SAMPLES: vectors generated before update of the probability vector (10-50 per population, depending on run).
LR: the learning rate, how fast to exploit the search performed (0.1).
LENGTH: length of encoded solution (problem dependent).

Figure 9: Generation of samples based on two probability vectors. Shown with 1 point crossover. Mutation are used as shown in Figure 8. There are a total of 10 subpopulations.

4 Problem Encoding and Search Space Representation

This section examines solution encoding and search space representation. The encodings represent the model vertices directly in the solution string and therefore only search along the surface of the model. Alternative encodings, which represents points in 3-D space which are mapped onto the appropriate vertex on the surface of the model, are described later.

4.1 Default Encoding

In a problem of selecting **N** vertices from a set of vertices, **V**, assign a unique value to each vertex [1..|**V**|]. The solution representation is then just **N** integers, each in the range [1..|**V**|]. This encoding works well for the steepest and next ascent hillclimbing method. In steepest ascent hillclimbing, because all possible single replacement steps are examined before a move is selected, the encoding has the least impact on the effectiveness of search. The encoding also does not have a large impact with next-ascent hillclimbing, since it randomly tries swapping vertices into the current set. In this study, as well as many others which use either GAs or PBIL, solutions are commonly represented as binary strings. Therefore, each of the **N** integers is represented with $\log_2[|\mathbf{V}|]$ bits. The decoding of the $\log_2[|\mathbf{V}|]$ bits into integers can either be done by interpreting the bits as a number encoded in standard binary notation, or as a number encoded in gray code. However, with this encoding, the interpretation will make little difference. Because there is no constraint to map nearby integer values to spatially close vertices, selecting nearby points may require making large modifications of the solution string. This creates a difficult space to search, since the integer encoding of vertices does not yield any information of where to explore next. In order for the GA or PBIL to effectively search the space, the encoding must reveal information which can be used to select new sample points. This problem has been identified in a more general form in [Jones and Forrest, 1995]. In the next encoding, this problem is addressed.

4.2 Neighborhood Encoding

To create neighborhoods in which the GA can easily search, two criteria were followed. First, the representation used for similar sets of points should be similar. Second, points which are spatially close to the current point should be accessible via a small number of moves. One method to create such an encoding is described below.

Each point is assigned a priority, initialized to 0. A single point is randomly chosen, it is assigned to be point 1. Its **C** closest neighbors are chosen and their priority is incremented. The element with the highest priority which has not yet been selected is chosen as the next point, and its priority is set to 0. Then, the priorities of its **C** closest unselected neighbors are incremented. This process continues until all elements are chosen. Ties are broken in favor of the elements closest to the one last selected. This selection procedure has the effect of growing a region in which higher priority values represent unselected points close to the set of currently selected points. The size of **C** was chosen empirically based upon the observation that if **C** is too large or too small, the effectiveness of the resulting encoding decreases. With **C** too small, long “chains” are formed such that neighboring points may have very dissimilar values. With **C** too large, the distance between successive points can increase. In these experiments, **C** was chosen to be 10. Figure 10 shows how the regions grow on the “Venus” model.

The general problem of mapping 2-D or 3-D spaces into a 1-D space has many solutions; the appropriateness of each depends upon the requirements of the final mapping. The mapping needed in this study should have the property that spatially close points should have small differences in the vertex numbers assigned to them. This problem is itself a difficult combinatorial search problem; only heuristic methods were explored here. Alternative mapping schemes are described later. For an overview of these issues and an introduction to some of these techniques, see [Goodhill et al., 1995].

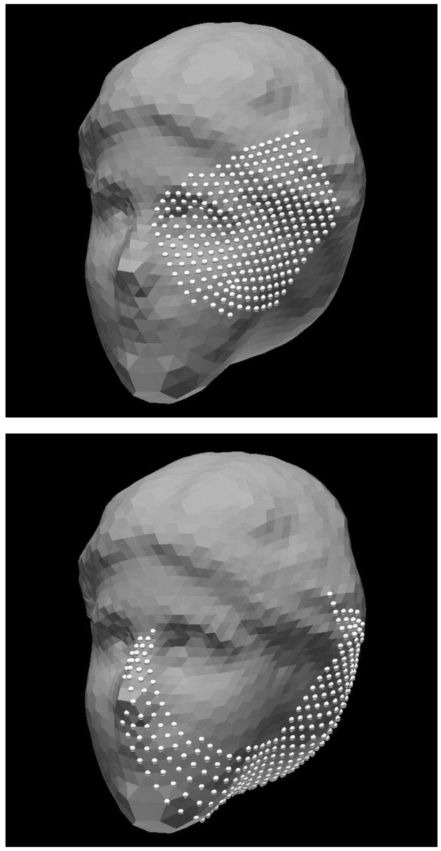

Figure 10: Neighborhood Encoding. Top: points labeled 1-300. Bottom: points labeled 300-600.

5 Point Set Selection: Results

This section compares the search algorithms described in Section 3. The results are presented in order of increasing performance. All results are computed from the average of at least 5 runs, started with random initial conditions. Each method was allowed approximately 5×10^5 evaluations per run. The models tested are shown in Figure 11. For each of these models, three point set sizes were constructed, containing 30, 60 and 90 points.

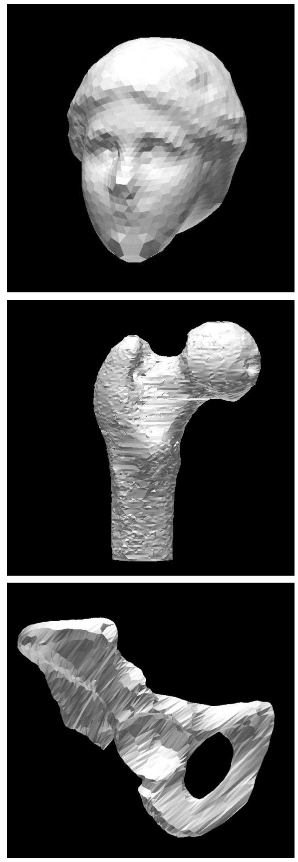

Figure 11: Venus, femur and pelvis models

5.1 The Venus Model

The most extensive exploration of different search methods is provided for the Venus model, which contains 2432 vertices. All of the experiments in this section are done with the neighborhood encoding, defined in Section 4.2. The results are shown in Table 1. The first method explored is steepest ascent hillclimbing (SAH); this was described in 3.1. The second search technique used is parallel genetic algorithms, with 10 subpopulations, as described in section 3.3. One of the parameters that has a large impact on the efficacy of search and on the computational time required is the population size. Three different population sizes were attempted, 100, 200 and 500 with the number of members equally divided between subpopulations. Another modifiable operator in GAs is the crossover operator. One point crossover, two point crossover and uniform crossover (Figure 6) were all attempted. The mutation rate was hand-tuned to work well. All nine combinations of crossover and population size were attempted. They uniformly improved the performance over SAH. Other GAs were also tried, using different scaling of values and/or steady-state selection [Whitley and Starkweather, 1990], and/or single large populations. None of these methods provided uniformly better performance over the GA results presented here.

The third method explored is next ascent hillclimbing (NAH) described in section 3.2. This was able to uniformly do better than any of the genetic algorithms attempted. Recently, [Juels and Wattenberg, 1994] have compared stochastic hillclimbing methods with GAs and have found similar results. For the runs reported in this paper, NAH was restarted multiple times until the total number of evaluations equalled those used by the GA and PBIL methods. The best evaluation found, over the multiple restarts, was considered the final evaluation in the run.

The final method examined, parallel Population Based Incremental Learning, described in section 3.4, also has a tunable population size parameter. The population sizes used for PBIL correspond to the three used in the GA. The performance improves for each respective population size when compared to the GA. However, only the performance of PBIL with 50 members per population surpasses the performance of NAH.

Finally, if the hillclimbing methods are used with the best solutions found by PBIL as the starting point, performance improves over either method alone. PBIL quickly finds regions of high performance, while hillclimbing is an effective way of finding local optima. Although only simple methods of integrating these two methods were explored here, further exploration into the appropriate division of computational time between PBIL and NAH is warranted.

Point selection results are shown in Figure 12. Each of the spheres in the figure corresponds to one or more selected measurement points. The radius of a sphere corresponds to the number of times that a single point is selected. There are several reasons why it makes sense to select a single point multiple times. From a mathematical viewpoint, multiple point selection can increase the sensitivity measure of equation (3). Points which are locally unique in terms of underlying surface orientation can improve localization sensitivity in directions which other nearby points cannot. Thus, it may be desirable to acquire multiple measurements in a small region about a single point.

Note that the measurement configuration shown in Figure 12 is not symmetric. For example, measurements are acquired from only one side of the nose. Once again, this can be justified with reference to the underlying mathematics. From the perspective of equations (1) and (2), there is no implicit reason that symmetric points should be preferred. Points on the right side of the nose serve exactly the same purpose as do points on the left. Thus, while an asymmetric measurement configuration might not satisfy our aesthetic sensibilities, functionally it is justified.

Due to the formulation of the geometric constraint analysis in [Simon et al., 1995a], there is a dependence of the sensitivity measure in equation (3) on the scale of the underlying object. While scale normalization is performed, in general it is im-

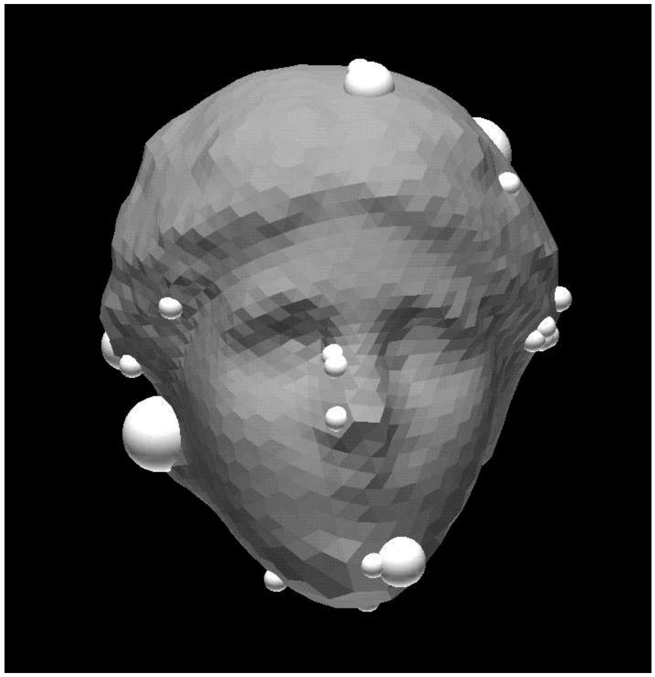

Figure 12: Venus point selection results - 30 points

Table 1: Venus Results (Sensitivity Measure)

Method	Point Set Size		
	30	60	90
Steepest Ascent Hillclimbing (SAH)	0.97	2.10	2.93
GA - One Point Crossover - Pop. 10 (10 populations)	1.29	2.57	3.73
GA - Two Point Crossover - Pop. 10 (10 populations)	1.19	2.67	3.90
GA - Uniform Crossover - Pop. 10 (10 populations)	1.19	2.62	3.70
GA - One Point Crossover - Pop. 20 (10 populations)	1.35	2.65	3.88
GA - Two Point Crossover - Pop. 20 (10 populations)	1.43	2.63	3.84
GA - Uniform Crossover - Pop. 20 (10 populations)	1.26	2.63	3.73
GA - One Point Crossover - Pop. 50 (10 populations)	1.52	2.85	3.93
GA - Two Point Crossover - Pop. 50 (10 populations)	1.50	2.94	3.86
GA - Uniform Crossover - Pop. 50 (10 populations)	1.46	2.66	3.47
Next Ascent Hillclimbing (NAH)	1.71	3.35	5.05
PBIL - Pop. 10 (10 populations)	1.63	3.28	4.44
PBIL - Pop. 20 (10 populations)	1.68	3.48	5.28
PBIL - Pop. 50 (10 populations)	1.80	3.70	5.40
PBIL + SAH	1.84	3.84	5.65
PBIL + NAH	1.86	3.91	5.65

possible to fully eliminate the effects of scale on the sensitivity analysis. This phenomenon explains the difference between the magnitude of the results reported in Table 1 and the results which follow.

5.2 EASIER MODELS: FEMUR & PELVIS

In this section, two other surface models, the femur and the pelvis, are explored. These contain 2530 and 2375 vertices respectively; the neighborhood encoding is used. Results for these models are shown in Table 2 and Table 3, respectively. Because genetic algorithms are the most computationally expensive algorithms used in this study, and they did not reveal as good performance as either NAH or PBIL even when allowed more evaluations, they are not explored further. Point selection results are shown for the femur in Figure 13.

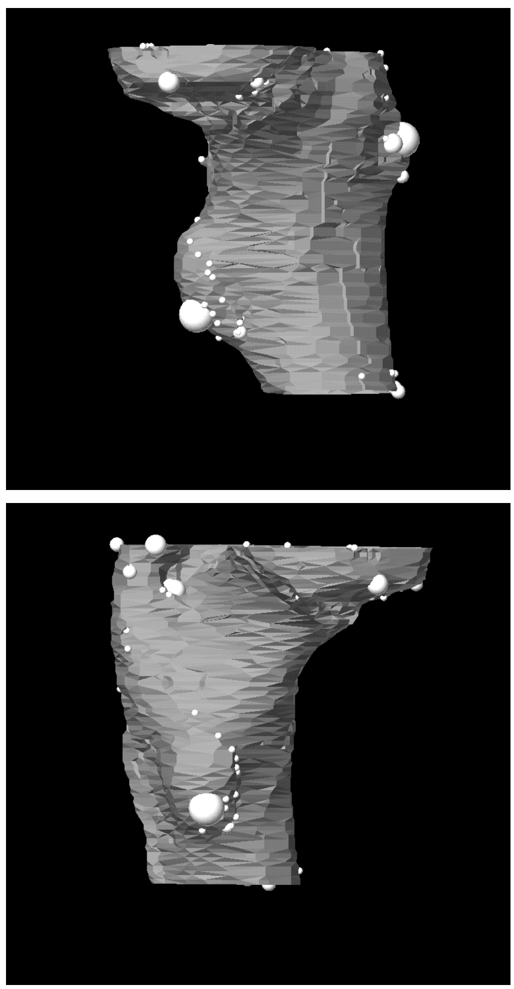

Figure 13: Femur point selection results - 90 points

Table 2: Femur Results

Method	Point Set Size		
	30	60	90
Steepest Ascent Hillclimbing (SAH)	7.85	17.17	27.40
Next Ascent Hillclimbing (NAH)	8.68	18.85	28.68
PBIL - Pop. 50 (10 populations)	8.62	18.35	28.14
PBIL + SAH	8.99	19.20	29.30
PBIL + NAH	8.89	19.28	29.13

Table 3: Pelvis Results

Method	Point Set Size		
	30	60	90
Steepest Ascent Hillclimbing (SAH)	6.87	15.72	25.12
Next Ascent Hillclimbing (NAH)	8.09	17.71	26.63
PBIL - Pop. 50 (10 populations)	8.27	17.97	27.31
PBIL + SAH	8.65	18.64	28.87
PBIL + NAH	8.68	18.81	28.57

5.3 A LARGER EXAMPLE

The results on a larger example of the femur, which contains 4841 vertices, are shown in Table 4. The neighborhood encoding is used. The requirements for this model were set sizes of 10, 20, 30, 40, and 50 points. Here, the NAH procedure has a clear advantage on the smallest point sets (10 points). This advantage decreases as the points sets grow larger. As the points set sizes increase, PBIL, PBIL + SAH, and PBIL + NAH, are favored.

Table 4: A Larger Model

Method	Point Set Size				
	10	20	30	40	50
SAH	1.15	2.65	4.96	7.23	8.37
NAH	1.86	3.76	5.74	7.67	9.37
PBIL - Pop. 50 (10 populations)	1.26	3.72	5.93	8.08	10.25
PBIL + SAH	1.44	3.90	6.10	8.73	10.88
PBIL + NAH	1.51	3.87	6.38	8.83	11.14

5.4 THE EFFECTS OF REPRESENTATION

In the discussion of representations, it was mentioned that if the vertices are randomly assigned a number, methods such as PBIL or GAs will not perform well. A comparison of the PBIL method using both neighborhood and random encoding is given on the Venus model in Table 5. The PBIL algorithm used a population size of 50 for each experiment. All of the parameters were held constant across both runs. Assigning values to the vertices randomly does not work as well as assigning numbers based on neighborhoods.

Table 5: Effects of Representation (PBIL population size is 50 per population)

Representation	Point Set Size		
	30	60	90
Neighborhood Assignment	1.8	3.7	5.4
Random Assignment	1.3	2.6	3.5

6 CONCLUSIONS & FUTURE WORK

This paper has presented novel techniques for selecting point data for object localization. In this paper, the object localization problem was explored in the context of computer assisted surgery; however, object localization is a problem which occurs in many areas of science and engineering. The point selection problem described has application to any localization problem in which it is desirable to minimize the amount of measurement data. The techniques developed in this paper are currently employed for selecting discrete point measurement configurations for use in surgical registration. Initial results from a cadaver trial suggest that these methods will lead to more accurate and reliable registration compared to existing methods. Additional details regarding shape-based registration and intelligent selection of registration data with application to computer-assisted surgery can be found in [Simon, 1996].

Discrete point data selection for maximizing sensitivity is a high dimensional combinatorial optimization problem. This paper explored several techniques for exploring this space: two variants of hillclimbing, genetic algorithms, and Population Based Incremental Learning. Empirically, it was found that a combination of PBIL and hillclimbing provided the best results.

There are several immediate future directions of this research. The most important direction is to determine how well the selected point sets work in practice. In order to perform this evaluation, it is necessary to have highly accurate estimates of true object pose. Accurately estimating the true object pose for use as ground-truth is a difficult engineering problem. Initial progress on this problem is reported in [Simon et al., 1995b].

The near-optimal measurement point configurations which are generated in this work represent blueprints or plans for the acquisition of measurement data. The actual measurements remain to be acquired by either a human (e.g., a surgeon), or an automated data collection device (e.g., a robot). In the process of acquiring the measurements, the precise locations specified in the acquisition blueprints may not be faithfully maintained. Due to this "acquisition uncertainty", registration accuracy may not be as good as predicted by the sensitivity analysis in [Simon et al., 1995a]. Experiments are currently being performed to evaluate the effects of this acquisition uncertainty. The sensitivity analysis may be modified to account for this uncertainty.

Discrete point measurements have been used in this paper for object localization. In general, other types of measurement data are possible. For example, the work described in [Lavallee and Szeliski, 1995] uses contour data extracted from X-Ray images to perform object localization. Since X-Ray radiation is harmful to humans, it is desirable to minimize the number of X-Ray images required to perform localization while ensuring sufficient localization accuracy. A variant of the discrete point selection method can be applied to selecting near-optimal contour measurements.

In this paper, only one search space encoding was used; however, there are many ways to encode the solution strings. An alternative to representing the points explicitly in the solution string (as was done in this study) is to allow the 3-D coordinates for **N** points to be selected. The selected point in 3-D space could then be mapped onto the closest point on the surface of the model. One of the advantages of this method is that it is easy to find neighboring points, as it is possible to directly move the search point in any direction. The largest disadvantage, and the reason that this method was not chosen, is the computation cost. Finding the closest point is an expensive operation.[1] Another expense is that the number of parameters which must be searched is tripled. Since many evaluations (10^5-10^6) are performed per run, these expenses should be carefully considered. Another encoding currently under investigation is using Kohonen Feature Maps to assign the points which are close to each other with "similar" values. Similarities can be defined by differences in integer values, or their Hamming distance from each other when the integers are represented in binary. Both methods are under investigation.

Finally, in terms of the search mechanisms, one of the promising directions found in this study was a combination of PBIL and hillclimbing. PBIL has the ability to find regions of high performance quickly. However, like genetic algorithms, it performs local optimization slowly. Hillclimbing has complementary properties; it is very quick for local optimization. The combination of these two techniques provided the best performance. In the future, studies should be done to determine the percentage of search which should be done by each of these methods, and the best way to integrate these methods.

1. However, methods such as k-d trees may be able to reduce this cost.

7 REFERENCES

Baluja, S. (1995). An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In Prieditis, A. and Russell, S., editors, *Proceedings of the International Conference on Machine Learning (ML-95)*, pages 38–46, San Mateo, CA. Morgan Kaufmann Publishers.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

Cohoon, J., Hedge, S., Martin, W., and Richards, D. (1988). Distributed genetic algorithms for the floor plan design problem. Technical Report TR-88-12, School of Engineering and Applied Science, University of Virginia, Charlottesville, VA.

Davidor, Y. (1991). A naturally occurring niche and species phenomenon. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 257–263, San Mateo, CA. Morgan Kaufmann Publishers.

DeJong, K. (1975). *An Analysis of the Behavior of A Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan.

Fang, H., Ross, P., and Corne, D. (1993). A promising approach to job-shop scheduling, re-scheduling and open-shop scheduling problems. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 375–382, San Mateo, CA. Morgan Kaufmann Publishers.

Furuya, H. and Haftka, R. (1993). Genetic algorithms for placing actuators on space structures. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 536–543, San Mateo, CA. Morgan Kaufmann Publishers.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Goodhill, G., Finch, S., and Sejnowski, T. (1995). Quantifying neighborhood preservation in topographic mappings. Technical Report INC-9505, Institute for Neural Computation.

Gordon, V. and Whitley, D. (1993). Serial and parallel genetic algorithms as function optimizers. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 177–184, San Mateo, CA. Morgan Kaufmann Publishers.

Hertz, J., Krogh, A., and Palmer, R. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, MA.

Homaifer, A., Guan, S., and Liepins, G. (1993). A new approach on the tsp by genetic algorithms. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 460–466, San Mateo, CA. Morgan Kaufmann Publishers.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642.

Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Forrest, S., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, San Mateo, CA. Morgan Kaufmann Publishers.

Juels, A. and Wattenberg, M. (1994). Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. Technical Report CSD-94-834, University of California - Berkeley.

Kim, J. and Khosla, P. K. (1991). Dexterity measures for design and control of manipulators. In *Proceedings of the International Workshop on Intelligent Robots and Systems*, pages 758–763, Osaka, Japan. IEEE/RSJ.

Lavallee, S. and Szeliski, R. (1995). Recovering the position and orientation of free-form objects from image contours using 3d distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):378–390.

Levine, D. (1993). A genetic algorithm for the set partitioning problem. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 481–487, San Mateo, CA. Morgan Kaufmann Publishers.

Navhi, A. (1994). *Identification and Control of the Redundant Linear Drives of Two Anthropomorphic Robots*. PhD thesis, McGill University, Montreal.

Simon, D. A. (1996). *Fast and Accurate Shape-based Registration*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Simon, D. A., Hebert, M., and Kanade, T. (1995a). Techniques for fast and accurate intra-surgical registration. *Journal of Image Guided Surgery*, 1(1):17–29.

Simon, D. A., O'Toole, R. V., Blackwell, M. K., Morgan, F., DiGioia, A. M., and Kanade, T. (1995b). Accuracy validation in image-guided orthopaedic surgery. In *Proceedings of the Second International Symposium on Medical Robotics and Computer Assisted Surgery*, Baltimore.

Taylor, R. H., Lavallee, S., Burdea, G. G., and Mosges, R., editors (1995). *Computer-Integrated Surgery.* The MIT Press, Cambridge, Massachusetts.

Whitley, D. and Starkweather, T. (1990). Genitor II: A distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:189–214.