Proceedings of 2004 JUSFA
2004 Japan – USA Symposium on Flexible Automation
Denver, Colorado, July 19-21, 2004

Paper Number UL 077

PARAMETER CALIBRATION FOR RECONFIGURATION OF A HAND GESTURE TELE-ROBOTIC CONTROL SYSTEM

Helman Stern
Department of Industrial Engineering and Management
Ben Gurion University of the Negev
Beersheva 84105 Israel
Email: helman@bgumail,bgu,ac,il

Juan Wachs
Department of Industrial Engineering and Management
Ben Gurion University of the Negev
Beersheva 84105 Israel
Email: wachs@bgumail.bgu,ac,il

Yael Edan
Department of Industrial Engineering and Management
Ben Gurion University of the Negev
Beersheva 84105 Israel
Email: yael@bgumail.bgu,ac,il

ABSTRACT

In this work we design two versions of a local neighborhood search algorithm. These versions are customized for a system operational parameter calibration task. The first and second methods perform complete and incomplete probabilistic neighborhood searches, respectively. The domain of application of the method is that of hand gesture control of telerobotic systems. The primary need for recalibrations of such systems is frequent relocation to other environments such as; laboratories, and remote control stations. A secondary need for recalibration, is the custom redesign of the gesture control language. This occurs for new users, new control tasks and new vocabularies. Allowing for fast recalibration of system parameters provides the system flexibility to respond to such new system setups. The two proposed methods were compared using a test case of 13 gesture commands. Both provided hand gesture recognition accuracies of 99.78. However, the probabilistic version had 48.5 percent less solution evaluations. Needless to say, the methodology reported here has wider application.

Index Terms— fuzzy c-means, gesture recognition, hand gestures, neighborhood search, telerobotics, supervised clustering, automated setup. parameter calibration, reconfiguration.

1. INTRODUCTION

Hand gestures are a common method for telerobotic control [1]. This type of communication provides an expressive, natural and intuitive way for humans to control robotic systems. One benefit [2] of such a system is that it is a natural way to send geometrical information to the robot such as: left, right, etc. Gestures may represent a single command, a sequence of commands, a single word, a phrase, and may be static or dynamic. Correct classification in reasonable time must be obtained for practical use [3].

Human-robot interaction using hand gestures provides a formidable challenge. This is because the environment contains a complex background, dynamic lighting conditions, a deformable hand shape, and a real-time execution requirement. There has recently been a growing interest in gesture recognition systems with a number of researchers providing some novel approaches, many of which are quite elaborate and require intensive computer resources. For example, Quek [4] develops a flow field computational algorithm. Koons *et al.,* [5] describes an approach in which hand data is classified into features of posture, orientation and motion. In [6] color, motion, and tracking is used in a hand posture system for robot control. Classification is based on elastic graph matching. In [7] an edge based technique is used to extract image parameters from simple silhouettes. An excellent review of gesture modeling approaches is that of Huang et al*.,* [8]. In this paper we define 13 static gesture postures (Fig. 1**)** for telerobotic control using a Supervised Fuzzy C-Means (FCM) recognition system. The robustness of the feature extraction and classification methods can affect the success or failure of any human-robot system using hand gesture control

In this paper we deal with another aspect affecting the success or failure of human – robot interaction. That is the often-ignored issue of system reconfigurability. The primary need for recalibrations of such systems is frequent relocation

to other environments such as laboratories and remote control stations. A secondary need for recalibration occurs due to demands for custom redesign of the gesture control language. This occurs for new users, new control tasks and new vocabularies. Allowing for a fast recalibration of system parameters provides the system flexibility to respond to such new system setups.

To test the methodology we address the difficult problem of simultaneous calibration of the parameters of an Image Processing - Fuzzy C Means (FCM) hand gesture recognition system. The approach taken to automate the calibration of the parameters of such a system is that of local neighborhood search. Thus, the design of a hand gesture recognition system is transferred into an optimization problem. We design two versions of the local neighborhood search algorithm involving a complete and probabilistic neighborhood search. The two proposed methods were compared using a test case of 13 gesture commands shown in Figure 1.

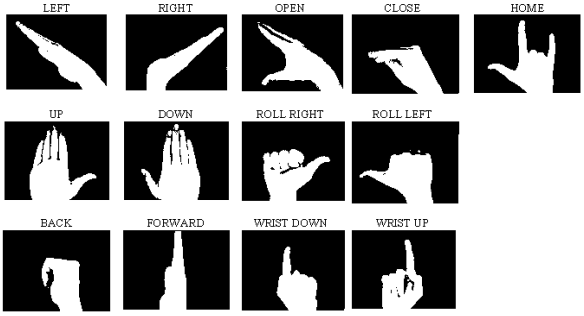

Fig. 1. Hand gesture vocabulary

In the following section the hand gesture recognition system is described. This is followed in section 3 by a discussion of parameter calibration. In section 4 local neighborhood search algorithms are described. In sections 5 and 6 our proposed complete and probabilistic neighborhood search algorithms are presented. In section 7 a computational comparison between the two algorithms is made on an experimental test set of gestures. Section 8 introduces Tele-Gest, a real time implementation of the system and usability test. The final section provides conclusions and future work.

2. THE HAND GESTURE RECOGNITION SYSTEM

The system is comprised of two main modules. These are; gesture image feature extraction, and fuzzy c-means (FCM) recognition.

Preprocessing and feature extraction

Preprocessing of the image starts with the segmentation of the hand from the background using a black/white threshold. A bounding box is constructed around the hand and divided by a block partition. A feature vector of the image is comprised of the aspect ratio of the bounding box, and the average intensity of each sub-block (fraction of white pixels) in the partition. Let R_b and C_b represent the number of rows and columns, respectively, of the block partition. This results in a feature vector of length $v = 1 + R_b \times C_b$, denoted as $f = (f_1, \ldots, f_i, \ldots, f_v)$. The first feature represents the aspect ratio of the bounding box, the remaining represent block averages indexed row wise from left to right. An illustration of the feature extraction operation is shown in Figure 2.

(a) **(b)**

Fig. 2. Illustration of a feature extraction. (a) hand gesture within bounding box. (b) 3x4 block partition

The resultant feature vector of the example is: f = (102 176 52 2 2 68 249 171 16 3 13 253 188). All feature values are scaled to lie in the range [0,255] (0-white, 255-black). One can see that the aspect ratio is 102 and blocks 3 and 4 are close to zero (white) while blocks 6 and 11 are close to 255 (black). The particular image processing threshold and the size of the image partition are design variables that must be determined.

Let $w = (w_1, \ldots, w_i, \ldots, w_v)$ represent the weight vector where, w_i is the weight attributed to feature i. The weights are normalized to sum to one.

$$\sum_{i=1}^{v} w_i = 1 \ , \ 0 \le w_i \le 1 \qquad (1)$$

Let $x = (w_1 f_1, \ldots, w_i f_i, \ldots, w_v f_v)$ be a weighted feature vector (also referred to as a data pattern) [9].

Feature Weighted Fuzzy C-Means Clustering and Recognition

The Fuzzy C-Means Clustering algorithm (FCM) is an unsupervised clustering algorithm [10]. We use a weighted feature FCM algorithm in which a weighted feature vector represents each gesture. The set of feature vectors are clustered for subsequent use in a recognition system.

Given a set of n data patterns $X=\{ x_1, \ldots, x_k, \ldots, x_n \}$,the algorithm minimizes a membership weighted within-group sum of squared error objective function, $J(U,V)$.

$$J(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} \mu_{ik}^{m} d^2(x_k, v_i) \qquad (2)$$

A matrix U with common element μ_{ik} represents a fuzzy c-partition of X. Let V be a c-tuple with common element v_i . Also, define the following: x_k is the weighted feature vector of the k^{th} exemplar gesture in the training set ($x_k \in R^v$), where; $X \subseteq R^v$, R^v represents the set of v tuples of the reals. v_i is the prototype feature vector of cluster i ($v_i \in R^v$), μ_{ik} is the degree of membership of x_k in the i-th cluster, m is a weighting exponent of the fuzzy membership values, $d(x_k,v_i)$ is a distance measure between data pattern x_k and cluster center v_i, q is the number of examples in the training set, and c is the number of clusters.

The objective function $J(U,V)$ is minimized via an iterative process, using (3), in which the degrees of membership u_{ik} (where j is a cluster index), and the cluster centers v_i are

updated:

$$v_i = \frac{\sum_{k=1}^{q} \mu_{ik}^m x_k}{\sum_{k=1}^{q} \mu_{ik}^m} \quad , \quad \mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{ij}} \right)^{\frac{2}{m-1}}} \qquad (3)$$

Where, the μ_{ik} satisfies:

$$0 < \sum_{k=1}^{q} \mu_{ik} < n \ \forall i \ , \mu_{ik} \in [0,1] \ , \sum_{i=1}^{c} \mu_{ik} = 1 \ \forall k \qquad (4)$$

After convergence of the FCM each weighted feature vector x_k is assigned to a cluster by finding: $\mu_{i'k}$ = Max $\{\mu_{ik}$, i=1,…,c$\}$. Once the clusters have been created, they must be labeled using a gesture name. This in effect allows the FCM clustering algorithm to be "supervised". Normally, the labeling of clusters presents no problem using the most popular class appearing in each cluster as its label - the Max Rule. This works well when the within cluster objects are close to homogeneous. However, since this is not always the case, a cluster labeling method described in our earlier work [11] is used. A new gesture is recognized by finding the cluster that maximizes its membership value using (3). The crucial question of the best combination of number of clusters and feature weights is addressed in section 3.

3. CALIBRATING OPERATIONAL PARAMETERS

A critical step in the design of a gesture recognition system is to determine the values of the operational parameters. Denote the vector $p \in R^n$ as the set of input parameters. Two types of input parameters are used: image processing features (block partition size, b/w threshold, feature weights of the aspect ratio and grayscale block features), and FCM parameters (number of clusters, and weighting exponent). The lower bound, LBj, upper bound, UBj, and increment values, Δpj, determine the discrete values of each parameter. Values for our example problem are recorded in Table 1.

Table 1. Parameter Definitions

Parameter, j	Definition	LB_j	UB_j	Δp_j
p_1	Number of Clusters, c	2	8	1
p_2	Weighting Exponent, m	1	4	0.25
p_3	b/w threshold, τ	0	255	1
p_4	Number of rows for image partition, R_b	2	8	1
p_5	Number of columns for image partition, C_b	2	8	1
p_6	Weight of the aspect ratio, w_1	0	1	0.1
p_7,..,p_i ...,p_n	Weights of the image block features, w_i	0	1	0.1

To determine the accuracy, A, associated with, p, define a mapping $\mathcal{A} : p \rightarrow A$. Determination of the functional value $\mathcal{A}$, for a given solution p, requires extraction of a new set of image features, executing the FCM algorithm, cluster label assignments, gesture classification, and analysis of the confusion matrix (Fig 3).

The best set of operational parameters within the parameter limits shown in the table is determined by local neighborhood search. Classical gradient search algorithms cannot be used, because the objective function is not well behaved with an unknown analytical form (and therefore obviously non differentiable) We will test two proposed versions of the local neighborhood search method using an application from the domain of gesture controlled robotic teleoperation. Gestures performed by a user are recognized using the highest membership value. System performance is evaluated using a confusion matrix that contains information about actual and classified gestures. The process of searching for optimal parameters for the combined image processing/supervised FCM stages is shown in the flow chart of Fig. 3.

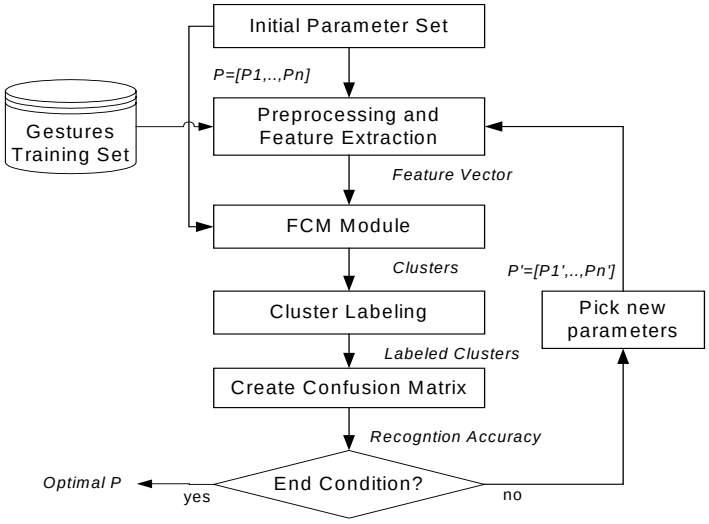

Fig. 3. Supervised FCM algorithm with parameter search

The output of the process is a set of parameters achieved by maximizing the recognition accuracy. The proposed neighborhood search algorithms are described in the following sections.

4. LOCAL NEIGHBORHOOD SEARCH ALGORITHMS

Local neighborhood search (NS) algorithms start from an initial solution p and continuously replace any solution with a better solution $\hat{p}$, in its neighborhood *N(p)* until no better solution can be found in *N(p)*. The neighborhood *N(p)* is the set of solutions obtainable from p by slight variations. A survey of neighborhood search techniques can be found in [12]. For an input parameter vector, *p*, a solution *p* is called locally optimal (with respect to the neighborhood), if no better solution exists in *N(p)*. We call the replacement of a current solution *p* by an equal or better solution a move or iteration. There are two commonly used move strategies; (a) the first admissible move strategy and (b) the best admissible move strategy. In the first admissible move strategy, the solutions in the neighborhood *N(p)* are examined and evaluated according to a prespecified or random order. The move is made to the first improved solution. In the best admissible move strategy; solutions in the entire neighborhood are evaluated before a move is made to the best solution found in *N(p)*.

To instantiate a particular case of the NS algorithm it is necessary to specify how an initial solution is generated, define a neighborhood *N*, give a specific criteria for selecting neighbors and define a termination condition. Different specifications will yield a variety of procedures. The most common NS algorithm is that which used the best admissible move strategy classically known as "steepest "decent" (or ascent in the case of a maximization problem), in which all neighbors are evaluated, and the one that most improves the current $\mathcal{A}(p)$ value is selected. The algorithm terminates when no improvement can be found. Resulting in a local optimal solution.

To escape from local optima, the improvement procedures can be embedded in a "metaheuristic" such as; simulated annealing, tabu search or a genetic algorithm. Genetic algorithms belong to the classical local search framework where improvement is sought within each move in the neighborhood of the current solution. In contrast, tabu search and simulated annealing allow the selection of an inferior solution once a local optimum has been reached.

In this work we propose and compare two local NS algorithms. The first is based on steepest "ascent" where the entire neighborhood is examined, while the second involves incomplete probabilistic sampling of the neighborhood. We refer to the first as the complete neighborhood search (CNS) algorithm, and the second as the probabilistic neighborhood search (PNS) algorithm This algorithm is related to the so called evolutionary strategy [13] where the value of each element of a string of parameters is mutated by a given probability

Initial solutions

A common problem with gradient search methods is that the global or local extrema reached is very sensitive to the starting solution used. To help alleviate this difficulty we generate nine starting solutions using a heuristic rule for each parameter. Three stratified values of the block partition variables R_b and C_b were selected to represent square partitions of size 2, 5, and 8. The weights of aspect ratio and image block features, w_j, were determined by creating a global feature dataset for each feature *j*, and calculating σ_j the standard deviation of the *j*th feature. The *j*th weight is calculated using (5).

$$w_j = 1 \Big/ \left(\sigma_j \sum_{i=1}^{v} \frac{1}{\sigma_i} \right) \qquad (5)$$

Allowing for the wrap around operation in the search algorithm, three initial cluster values equally spread over the range of possible values were selected as c = 13,16,20. A value of 2 was used as FCM weighting exponent, m for all initial solutions. Otsu's method [14] was applied to the average image of a random subset of gesture images to obtain the initial B/W threshold value τ. Note, that the initial starting points are spread out over the solution space in a stratified manner. Once the search starts from an initial point the trajectory of the successive points are free to roam over the entire solution space in the search for the global (or local) maxima. Using the initial solutions, nine runs of the NS algorithm will be run; and the best of them selected. Two proposed local search algorithms will be compared, each starting with the identical set of starting solutions. It is noted, that a finer distribution of starting solutions may increase the chance of reaching a better overall solution, but at the cost of additional computational time.

5. THE COMPLETE NEIGHBORHOOD SEARCH ALGORITHM (CNS)

In the CNS algorithm the search is conducted by examining all solutions in an n dimensional unit square neighborhood about any current solution point. The maximal discrete gradient within this neighborhood determines the selected coordinate axis in which a unit step is made (although improvements are made by increasing the step size when plateaus are discovered).

For any feasible solution $p=(p_{1,\ldots,}p_n)$ for the recognition system, define a set $N(p)$ of neighboring solutions of the vector p. The number of neighbors of p is $2n$ as each parameter is incremented up and down by an amount Δp_j (wrap around is used when the boundary values *LB, UB* are exceeded). The set of feature weight parameters are updated in a special way because of their inter-dependence derived from the sum to unity constraint. Moreover, the number of feature weights depends on the block partition parameter values. Given a block partition of R_b rows and C_b columns, the number of feature weight parameters is the same as the number of features, $v = 1 + R_b \times C_b$. The dimensionality, n, of our parameter vector is variable, and depends on the minimum and maximum block partition values. In general, for a square block partition of size b the length of the parameter vector is $n = b^2 + 6$. In our sample problem the minimum and maximum square block partition values range from 2 to 8. Here, the number of feature weight parameters can vary from 5 to 65 resulting in parameter vectors of dimension 10 to 70. To handle such variable length parameter vectors we consider p_4 and p_5 as control parameters, which turn 'off' and 'on' the appropriate weight parameters according to the following rule: whenever p_4 or p_5 change, the length of p is set to $n = 5 + v$. Let $\{p_i : i = 1,\ldots,n\}$ the current set of parameter values. To find the permissible neighbor values of a parameter p_i, increment p_i up and down by the discrete gradient Δj. The CNS algorithm starts with an initial solution p^0. A pseudo code for the CNS algorithm is shown below:

Algorithm CNS

1. **Begin**
2. Create an initial feasible solution $p^0=(p^o_{1,\ldots,} p^o_n)$
3. ***local_maxima=false***
4. **Repeat**
5. **Begin**
6. Find $\mathcal{A}(p')$ for all $p' \in N(p)$
7. Let $p'' = Arg\max\{\mathcal{A}(p') \mid \forall p' \in N(p)\}$,
8. **If** there are ties:
9. Pick the last p''
10. **If** p'' not in *Stack*, **Push** p'' into *Stack*

11. **Else** ***local_maxima=true***
12. **End if**
13. **If** $p'' = p$ then ***local_maxima=true***
14. Replace p by p''
15. **End**
16. **Until** $\mathcal{A}(p'') = 100\%$ **or** ***local_maxima=true*** **do**
17. Output p'', the local or global max solution
18. **End**

The main idea behind the CNS algorithm is to continuously find a better solution by advancing in the parameter space in one coordinate direction each time. Define an iteration as one cycle starting from the current solution p until the best neighbor solution p'' is selected. Note, that each iteration consists of an evaluation of all $2n$ neighborhood solutions in $N(p)$. If the accuracy in the iteration did not increase i.e. p'' is equal p, then a local maximal was found and the algorithm stops. However, if there are ties, a plateau has been reached. In case of plateau, the algorithm will try to find a better solution by advancing in the parameter space along the direction of a tied solution. However, if the best neighbor p'' has been visited before (kept in a stack) and no improvement is made in the entire plateau, an expansion of the neighborhood is made by doubling the step size for the next five iterations, in the hope of escaping from the local maxima.

The recognition accuracy function $\mathcal{A}$ is a non-decreasing function of k, the number of iterations, i.e., $\mathcal{A}(p^{k''}) \geq \mathcal{A}(p^{k-1''})$ where p^k is the parameter vector at iteration k. The algorithm stops when two successive iterations give the same accuracy value after exploring all neighbor solutions, if a plateau is reached. A plateau is the case where at least one neighborhood solution has the same value as p''. Since $\mathcal{A}$ is bounded above by 100 percent, termination in a finite number of steps is guaranteed. Detailed proofs may be found in [11].

6. PROBABILISTIC NEIGHBORHOOD SEARCH ALGORITHM (PNS)

Unlike the CNS algorithm, where the entire neighborhood is examined before a move is made, in the PNS algorithm, solutions in the neighborhood $N(p)$ are randomly sampled and evaluated. A move is made to the first improved solution found. If no improvement is made after K evaluations, the neighborhood is expanded, and a probability distribution is sampled to generate a new solution. The expansion increases the chance of escaping from local maxima. For any parameter p_j the following is defined:

Δp_j = the smallest step size increment taken in any coordinate direction $j = 1,\ldots,n$ in the solution space.

s = the number of steps made in either the positive or negative coordinate direction j

Δj = the discrete gradient in the j^{th} coordinate direction, where

$$\Delta j \in \{s\Delta p_j : s = 0,\pm1,\pm2,\pm3,..\pm S\}$$

S= maximum neighborhood size

$\nabla p = \{\Delta j\}$ = the gradient direction of a move from p (an n dimensional vector) with common element Δj.

$\hat{p} = \Psi(p, \nabla p)$, = the updated solution, where Ψ is a special operator mapping a vector p of size n_p to a vector $\hat{p}$ of size $n_{\hat{p}}$ ($n_p = or \neq n_{\hat{p}}$).

Identical neighborhood sampling probability distributions are defined for each coordinate parameter. Discretized Gaussians or equivalent binomial approximations (using probability of success = 0.5) have the property that an increased standard deviation not only spreads out the distribution but also reduces the peak value. As this behavior is undesirable because we want to control the proportion of parameter changes, we designed a special mixture type point distribution model. The characteristics of this distribution are that the tails can be spread out while the peak probability remains constant. The two parameters, h and S, characterize the probability distribution $P_S(x|h)$.

Let, S = maximum number of step increments.

h = probability of no change

x_j = a random variable representing the signed (positive or negative coordinate direction) number of step size changes for parameter p_j.

$\Omega = \{0,\pm1,\pm2,...,\pm S\}$ = the universe of the random variable x, of size 2S +1.

$P_S(x|h)$ the probability of step size x, given h and S.

$$\text{Ps}(x \mid h) = \begin{cases} h, & x = 0 \\ h((1-h)^{|x|}) / 2, & x = \pm1,\pm2,,,,\pm(S-1) \\ ((1-h)^{|x|}) / 2, & x = \pm S \end{cases} \quad (6)$$

In (6) |x| represents the absolute value. For example, if h = .9 and S = 3, the probability x = -2 is .0045.The probability mass function is symmetric, with a peak at x = 0 representing the probability that the parameter remains unchanged. The range of the distribution is linear in S. For a fixed h, the probability of x = 0 remains the same. The range of the distribution increases linearly in S, which expands the neighborhood allowing larger steps while the probability of not moving remains constant. This will have an average effect of allowing a same number of parameters to be changed; but by larger possible step sizes, increasing the chance of escaping from a local extrema. If, for example, h = .9 in the long run 10 percent of the parameters in the parameter string will change and 90 percent will remain the same.

Algorithm PNS

Given a solution p its updated value is determined by:

$$\hat{p} = \Psi(p, \nabla p) \quad (7)$$

where, $\Delta j = s\Delta p_j$

s = x with probability $P_S(x|h)$

$\hat{p}_j = p_j + \Delta_j, \forall p_j$

and p and p_j are same size vectors.

If one or more weight parameter changes occur then a repair operation is applied, in order to insure that the weights are normalized (sum to one). The new neighbor feature weights are updated using formula (8) below.

$$w_i' = \left\{ \left(w_i + \Delta j \right) \Big/ \left(1 - \sum_{k=1}^{v} \Delta_k \right), i = 1,..,v \right\} \qquad (8)$$

A run starts with an initial solution. Once an improved solution p is found, a new iteration commences with the improved solution. At the start of an iteration the neighborhood size is set to $S = 1$. For a parameter vector of size n, an evaluation consists of n samples taken from the neighborhood sampling distribution. If an improvement is found before K evaluations are made, the current solution is updated according to (7). If no improvement is found, the neighborhood is further expanded to $S = 2$ and then to $S = 3$. If, after three-neighborhood expansions no further improvement is found the algorithm terminates. As each evaluation of p, to determine the accuracy value of the functional $\mathcal{A}$, is time consuming, (recall that determination of the functional value $\mathcal{A}$, for a given solution p, requires extraction of a new set of image features, executing the FCM algorithm, cluster label assignments, gesture classification, and analysis of the confusion matrix) in order to reduce computation time a list of prior solution vectors p is maintained. After generating a new parameter vector by sampling from the neighborhood distribution, all previous solutions on the list are checked. If the new solution appears in the list it is dismissed, otherwise an evaluation is made. Although, searching the list of previously generated solutions involves additional computational effort, it is orders of magnitude less than the time spent during accuracy evaluation. If a sufficiently high accuracy is not achieved the maximum value of S can be increased and the problem rerun. This requires additional computational time with the usual returns to scale applying; that is, additional increases of max S will result in smaller incremental increases in accuracy.

7. COMPARISON OF CNS AND PNS AGORITHMS

An example test is conducted to illustrate the performance of the CNS and PSN algorithms using 35 samples per gesture for 13 gestures to obtain a training set of 455 samples. Using the starting solution heuristic the nine starting solutions shown in Table 2 were generated.

Table 2. Initial Solutions Used for CNS and PNS Runs

Run	*c*	*m*	*τ*	R_b	C_b	w_i	*A(%)*
1	13	2	146	2	2	0.73 0.075 0.074 0.058 0.063	84.84
2	16	2	146	2	2		96.04
3	20	2	146	2	2		95.60
4	13	2	146	5	5	0.343 0.03 0.024 0.023 0.025 0.04 0.025 0.022 0.027 ...	77.80
5	16	2	146	5	5		88.35
6	20	2	146	5	5		77.80
7	13	2	146	8	8	0.195 0.016 0.015 0.012 0.012 0.013 0.013 0.018 0.02 ...	83.30
8	16	2	146	8	8		85.93
9	20	2	146	8	8		90.33

Both CNS and PNS algorithms were tested with the same starting solutions. The results appear in Table 3 where, the accuracies obtained are shown in columns 6 and 7. Both algorithms obtained identical accuracy values of 99.78, shown in bold. Columns 4 and 5 contain the total number of iterations and the number of accuracy evaluations up to the start of the last iteration, which then runs for 3K more evaluations. This sum was added to column 4. For all PSN runs, the values of h = 0. 9, and K = 30 were used. It was found that using three neighborhood expansions (S=1,2,3) was sufficient to reach an acceptable accuracy (over 99%). The number of step increments for S can be further expanded, if desired, based on an incremental improvement threshold value. Figure 4 shows the convergence for run 5. The algorithm is designed to terminate after three neighborhood expansions without an increase in accuracy, or if A reaches its upper bound of 100 %.

Table 3. Comparison of CNS and PNS Algorithms on the Basis of Computational Steps and Accuracy

Initial Solution	Number of Iterations		Evaluations to Best Sol [1]		Accuracy, A(%)	
Run	PNS	CNS	PNS[2]	CNS[3]	PNS	CNS
1	17	3	359	60	97.08	97.8
2	5	1	124	20	99.12	97.8
3	3	2	20	40	98.02	98.02
4	2	7	69	434	97.14	99.12
5	16	12	376	744	**99.78**	**99.78**
6	1	8	1	496	96.04	99.34
7	8	4	115	560	95.6	99.12
8	2	3	16	420	96.92	98.46
9	1	7	45	980	99.34	99.56
Total			1,935	3,754		

[1] Evaluations to last parameter change for the run

[2] 3K=90 evaluations added after the last parameter change for each run (1,125 + 810 = 1,935)

[3] (Number of iterations)*No of evaluations /iteration)= total evaluations, where no of evaluations /iteration is fixed at 20,62 and 140 for sol 1-3, 4-6 and 7-9,respectively.

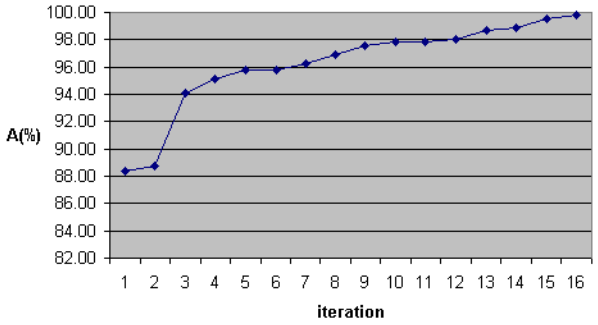

Fig. 4. Convergence curve for PNS (run 5)

Table 4 provides an example of the number of parameter changes at each iteration for run 5 of the PNS algorithm. This shows the trajectory of p through the solution space for the best run obtained.

Analysis of results

For the test run the PSN algorithm finds the same best solution as the CNS in 48.5 percent shorter computational time (1,935 vs. 3,754 evaluations). The reduction, however, is a conservative estimate as it is based on equal time evaluations

for the PNS algorithm runs. This is because the times to evaluate a solution are not equal , but are directly proportional to the size of the solution vector p, which is dynamic. Thus, a weighted evaluation time function should be used whereby, given t(n) as the time to evaluate a solution vector p of length n, an adjusted evaluation time can be determined as t(n)*r(n) where r(n) is the number of times a solution vector of size n is evaluated. This adjustment is left for future work.

Table 4. Number of Parameter Changes at Each Iteration

Iteration	Sol. No	c	m	τ	R_b	C_b	w	Total
1	83	1	0	0	0	1	0	2
2	109	0	1	1	0	0	0	2
3	137	0	1	0	0	0	0	1
4	174	0	0	0	1	0	0	1
5	178	1	0	0	0	0	0	1
6	183	0	0	1	0	0	0	1
7	204	0	0	1	0	0	0	1
8	249	0	0	0	0	0	1	1
9	257	1	0	0	1	0	0	2
10	260	1	0	0	0	0	0	1
11	269	0	0	1	0	0	0	1
12	295	0	0	0	0	1	0	1
13	366	1	0	0	0	1	0	2
14	372	0	0	0	1	0	0	1
15	375	0	0	0	0	0	1	1

8. REAL-TIME IMPLEMENTATION

Tele-Gest is a real time implementation of hand gesture teleoperated control of a remote fixed robot manipulator [15]. To control a robot movement, the user evokes a gesture from a gesture vocabulary. The user lays his/her hand under a video imager, and a raw image is acquired. The gesture is classified using a recognition module based on the weighted feature FCM algorithm and is sent to the robot for execution. The components of the system consist of a five-degree of freedom articulated robot, a Pentium 4, 1400 MHz PC with a Matrox Meteor frame grabber, two 3Com PC digital USB cameras, and a Panasonic video imager.

A set of recognized gestures is sent through the TCP/IP communication protocol to a distant robot PC server. The server is connected to the robot controller and two USB cameras, which continually capture the robot scene. Both side and front views of the scene are sent to the client, using FTP, and presented in a user interface. To enable the user to control the robot over the Internet, TCP/IP and FTP client-server applications were developed and installed on each computer. The hand gesture recognition system flow diagram is shown in Fig. 5.

Upon presentation of the robotic scene in the user's interface, a gesture γ (selected from the gesture vocabulary $\{G_1, G_2, \ldots, G_{13}\}$ is evoked. A vision system converts the captured image of the gesture into a feature string, which is subsequently recognized and sent to the robot PC server. After the robot executes the command, camera views of the robot environment are transmitted back to the interface. Fig. 6 is a screen dump of the hand gesture robotic control interface.

The interface contains; feedback on the current gestures recognition status, a gesture reminder display, and two actual robot views. Because of real robot sluggish movement, for large-scale usability testing, a second interface was designed, using a virtual robotic model. (See Fig. 7).

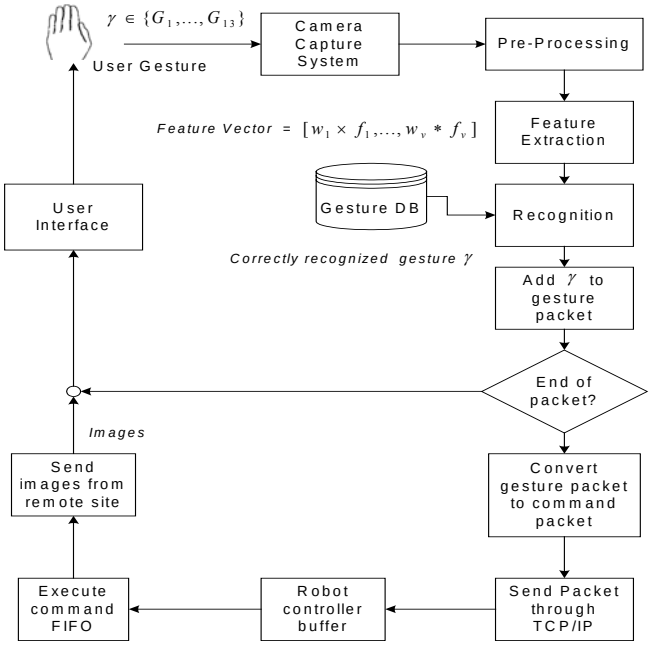

Fig. 5. System flow diagram of Tele-Gest

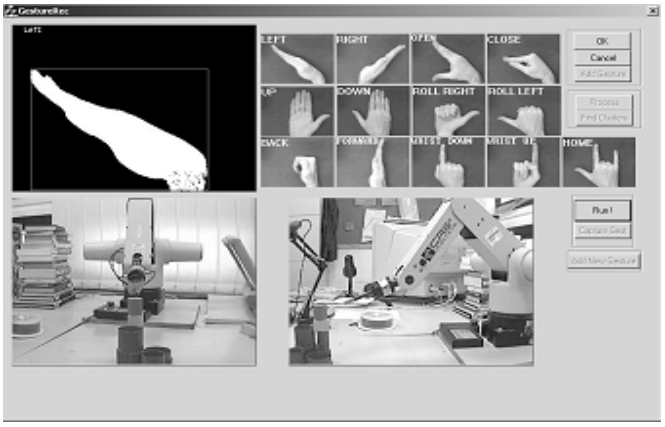

Fig. 6. Tele-Gest: User interface with real robot

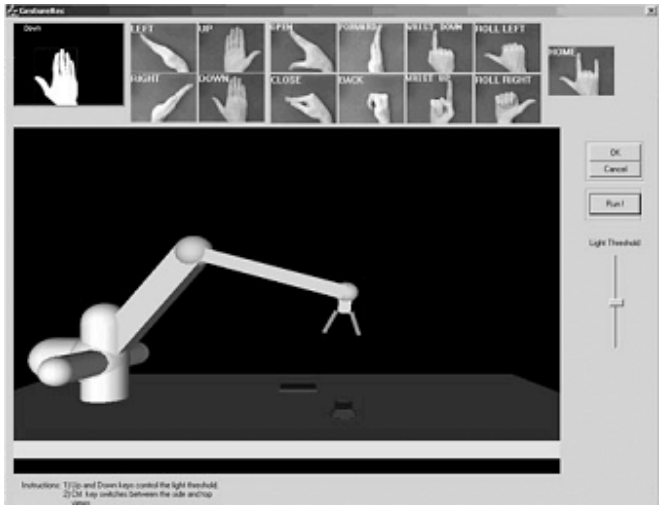

Fig. 7. Tele-Gest: User interface with virtual robot

Usability tests

An experiment was conducted to measure the usability of the system. Users guide a tele-robot to transfer a marker from one box to another using only hand gesture commands. Four new users performed five trials of the task using user trained dependent (D) and independent (I) systems. For each system, the users tried the gestures in the vocabulary for 2 to 10 minutes until they became familiar with the gestures. The time to complete each trial was recorded. The learning curves found for the novice users, indicated that the user will learn the task faster using the I system (learning rate of 0.81) than that of the D system (0.85). An example of the learning curve for the independent system is shown in Fig.8.

Fig. 8. Learning Curve for Independent System

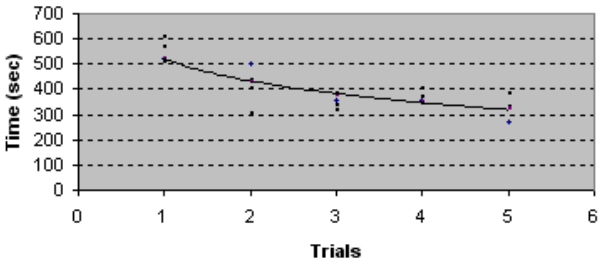

Fig. 8. Learning Curve for Independent System

9. CONCLUSIONS AND FUTURE WORK

In this work we design two versions of a local neighborhood search algorithm. These versions are customized for a system operational parameter calibration task, where the number of parameters in our solution vector is dynamically changing. The first and second methods perform complete and incomplete probabilistic neighborhood searches, respectively. The domain of application of the method is that of hand gesture control of telerobotic systems. The primary need for recalibrations of such systems is frequent relocation to other environments such as laboratories and remote control stations. A secondary need for recalibration occurs due to demands for custom redesign of the gesture control language. This occurs for new users, new control tasks and new vocabularies. Allowing for a fast recalibration of system parameters provides the system flexibility to respond to such new system setups. The two proposed methods were compared using a test case of 13 gesture commands. We found both provided hand gesture recognition accuracies of 99.78. However, the probabilistic version performed 48.5 percent less solution evaluations. Caution is advised however, in coming to premature conclusions that the probabilistic procedure is superior. The preliminary results reported here, however, warrant additional planned testing

Although extensive testing for the CNS algorithm has been performed as reported [11], in order to perform more comprehensive comparative testing more experience with the PNS algorithm is planned. Further work on the neighborhood search algorithm is also in order such as: investigating the effect of various dynamic strategies for expanding and contracting the neighborhood size. Since the length n of the solution vector is variable, it is planned to explore the effect of a dynamic maximum sample size *K*, where for example *K* can vary linearly in n. It is also intended to explore the effect of using repeated random trials for each of the heuristic starting solutions. Provide more accurate estimates of evaluation time to take into account the dynamic nature of the size of the solution vector. Finally, comparison with other algorithms such as; the genetic algorithm, tabu search, or simulated annealing will be undertaken.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Defense MAFAT Grant No. 1102, and partially supported by the Paul Ivanier Center for Robotics Research and Production Management, Ben-Gurion University of the Negev.

REFERENCES

[1] Katkere, A., Hunter, E., Kuramura, D., Schlenzig, J., Moezzi, S., and Jain, R., 1994, "ROBOGEST: Telepresence Using Hand Gestures," Technical Report VCL-94-104, University of California, San Diego.

[2] Kortenkam, D., Huber, E., and Bonasso, R. P., 1996, "Recognizing and Interpreting Gestures on a Mobile Robot," AAAI96.

[3] Pavlovic, V., Sharma, R., and, Huang, T., 1997, "Visual Interpretation of Hand Gestures for Human Computer Interaction: A Review," IEEE PAMI, Vol. 19, pp. 677-695.

[4] Quek, F. K. H., 1996, "Unencumbered Gestural Interaction," IEEE Multimedia, pp. 36-47.

[5] Koons, D. B., "*Integrating Simultaneous Input from Speech, Gaze, and Hand Gestures*," 1993, Intelligent Multimedia Interfaces, M. T. Maybury (Ed), MIT Press, pp. 257-276.

[6] Triesch, J., and Malsburg C. V. D., 1998, "A Gesture Interface for Human-Robot Interaction," Proceedings. of 3rd IEEE Intl. Conf. on Automatic Face and Gesture Recognition, pp. 546-551.

[7] Segen, J., 1993, "Controlling Computers with Gloveless Gestures," Proc. of Virtual Reality Systems.

[8] Huang, T. S., and Pavlovic, V. I., 1995, "Hand Gesture Modeling, Analysis, and Synthesis," Proceedings of Intl. Conf. on Automatic Face and Gesture Recognition.

[9] Modha, D. S., and Spangler, W.S., 2003, "Feature Weighting in k-Means Clustering," Machine Learning, 52**, 3**, pp. 217-237.

[10] Bezdek, J. C., 1973, "Cluster Validity with Fuzzy Sets," Cybernetics. **3, 3**, pp. 58-73.

[11] Wachs, J., Stern, H., and Edan, Y., 2003, "Parameter Search for an Image Processing – Fuzzy C- Means Hand Gesture Recognition System," Proceedings. IEEE Int. Conf. Image Processing, Barcelona, Spain, Vol. 3, pp. 341-344.

[12] Ahuja, R. K., Ergun, O., Orlin, J. B., and Punnen, A. P., 2002, "A Survey of Very Large Scale Neighborhood Search Techniques," Discrete Applied Math., **123, 1-3**, pp. 75-102.

[13] Dumitrescu, D., Lazzerini, B., and Jain, L., 2000, Evolutionary Computation, CRC Press, Boca Raton, New York.

[14] Otsu, N., 1979, "A Threshold Selection Method from Gray-Level Histograms," IEEE Trans. Systems, Man and Cybernetics, **9, 1**, pp. 62-66.

[15] Wachs, J., Stern, H., Edan, Y., and Kartoun, U., 2002, "Real-Time Hand Gesture Using the Fuzzy-C Means Algorithm," Proceedings. of WAC 2002, Florida.