Cluster Labeling and Parameter Estimation for the Automated Setup of a Hand-Gesture Recognition System

Juan P. Wachs, *Student Member, IEEE*, Helman Stern, *Member, IEEE*, and Yael Edan, *Member, IEEE*

Abstract—In this work, we address the issue of reconfigurability of a hand-gesture recognition system. The calibration or setup of the operational parameters of such a system is a time-consuming effort, usually performed by trial and error, and often causing system performance to suffer because of designer impatience. In this work, we suggest a methodology using a neighborhood-search algorithm for tuning system parameters. Thus, the design of hand-gesture recognition systems is transformed into an optimization problem. To test the methodology, we address the difficult problem of simultaneous calibration of the parameters of the image processing/fuzzy C-means (FCM) components of a hand-gesture recognition system. In addition, we proffer a method for supervising the FCM algorithm using linear programming and heuristic labeling. Resulting solutions exhibited fast convergence (in the order of ten iterations) to reach recognition accuracies within several percent of the optimal. Comparative performance testing using three gesture databases (BGU, American Sign Language and Gripsee), and a real-time implementation (Tele-Gest) are reported on.

Index Terms—Automated setup, cluster labeling, fuzzy c-means, gesture recognition, hand gestures, neighborhood search, supervised clustering, telerobotics.

I. INTRODUCTION

THE use of gestures for telerobotic control [1] is a fairly new human-centered method for man–machine communication that provides an expressive, natural, and intuitive way for humans to control robotic systems. A special 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. In this work we use static hand-gesture representations of a vocabulary of commands that are vision-based. 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 by a number of researchers providing some novel approaches, many of which

are quite elaborate and require intensive computer resources. A representative survey of such work is given below.

A vision-based system able to recognize 14 gestures in real time to manipulate windows and objects within a graphical interface was developed in [3]. Gu and Tjahjadi [4] developed a family of feature detectors for detecting landmarks on gesture images. Abe *et al.* [5] describe a system that recognizes hand gestures through the detection of the bending of the hand's five fingers, based on image-property analysis. In [6], a three-dimensional (3-D) reconstruction to recover hand gestures from stereo is used. An excellent review of gesture-modeling approaches is that of Huang and Pavlovic [7]. Edge-based techniques to extract image parameters from simple silhouettes are used in [8].

In the work of Franklin *et al.* [9], a robot waiter is controlled by hand gestures using the Perseus architecture for gesture recognition. In Becker *et al.* [10], users can operate a semi-autonomous robot through hand gestures. The work presented by Kortenkamp *et al.* [2] shows a system able to recognize six distinct gestures using a coarse 3-D model of a human. In the work of Cipolla *et al.* [11], a gesture-based interface for robot guidance is based on uncalibrated stereovision and active contours. The research of Guo *et al.* [12] discusses vehicles controlled by hand gestures, based on color segmentation. Waldherr *et al.* [13] propose a vision-based interface that instructs a mobile robot using pose and motion gestures in an adaptive dual-color tracking algorithm. Yang *et al.* [14] employ two-dimensional (2-D) motion trajectories, and a time-delay neural network to recognize 40 dynamic American Sign Language (ASL) gestures. Yin and Xie [15] create a fast and robust system that colors segments and recognizes hand gestures for human–robot interaction using a neural network. A system intended to be particularly robust, in real world environments, is presented by Triesch and Malsburg [16]. The strength of the recognition is based on color features extracted from a cadre of training images, and an elastic-graph-match approach. Ghidary *et al.* [17] use three gestures combined with voice to command a robot. Sato and Sakane [18] use pointing gestures to command a robot, by projecting a light mark in the real workspace. The recent work of Heidemann and Ritter [19] employs gestures to guide an anthropomorphic robotic hand to grasp objects.

Most of the systems reviewed rely on the simple idea of detecting and segmenting the gesturing hand from the background using motion detection or skin color. Vision-based hand-gesture

The authors are with the Industrial Engineering and Management Department, Intelligent Systems Division, Ben-Gurion University of the Negev, Be'er-Sheeva 84105, Israel (e-mail: juan@bgumail.bgu.ac.il; helman@bgu.ac.il; yael@bgu.ac.il).

robot-control systems have dealt with real-world constraints with variable success. The robustness reached by proper selection of features or clues, and their combination with sophisticated recognition algorithms, can affect the success or failure of any existing and future work in the field of human–robot interaction using hand gestures. In this paper, we deal with another aspect affecting the success or failure of human–machine interaction. That is the often-ignored issue of system reconfigurability. Our approach provides a methodology to automate the calibration of such processes. To test the automated set-up procedure, we use a hand-gesture recognition system comprised of an image processing/fuzzy C-means (FCM) classifier. This system is part of a real-time telerobotic-control system [21] described later in the paper. As the emphasis will not be on the image-processing aspects for hand-gesture recognition, we use a simple environment comprised of uniformly illuminated 2-D hand postures placed on a table top, at a constant distance from a video-capture device.

Even though the recognition problem is not intractable, such a system involves a timely calibration process, consisting of tuning the parameters of both the image processing and recognition subsystems. Most often this effort is ad hoc, using trial and error, and performed suboptimally examining the two system components independently. Also, such methods often suffer due to system designer impatience. Still important, is the flexibility of the system to changes induced by the transfer of the system to another location, or when the gesture language is changed for research or operational considerations. Here again, system performance is often sacrificed in the interest of fast reset time. Thus, we respond to the need for an automated setup procedure, which treats the calibration of all tunable system parameters simultaneously. As such, the design of hand-gesture recognition systems is transferred into an optimization problem. Needless to say, the methodology presented here for automating such system setups has wider application.

In the following section, the image-processing feature-extraction operation is described, followed by a discussion of the unsupervised FCM clustering algorithm and the need for it to be supervised. In Section III, the heuristic cluster-labeling algorithm to supervise the FCM is described. In addition, a new formulation of the optimal labeling procedure as a linear-programming problem is proffered. In Section IV, our neighborhood parameter search algorithm is presented. Performance evaluations and a discussion of the results appear in Sections V and VI, respectively. The final section provides conclusions and future work.

II. HAND GESTURE RECOGNITION SYSTEM

The hand-gesture recognition system is comprised of an image-processing feature-extraction operation followed by an FCM gesture classifier. The FCM clustering algorithm [20] is a popular method for image-recognition tasks [21]–[23]. Although the speed of artificial neural-network classifiers allows real-time operation and comparable accuracy, an FCM is used because it requires smaller training sets and shorter training times. Another reason for selecting a fuzzy clustering framework is that it makes the clustering approach less sensitive
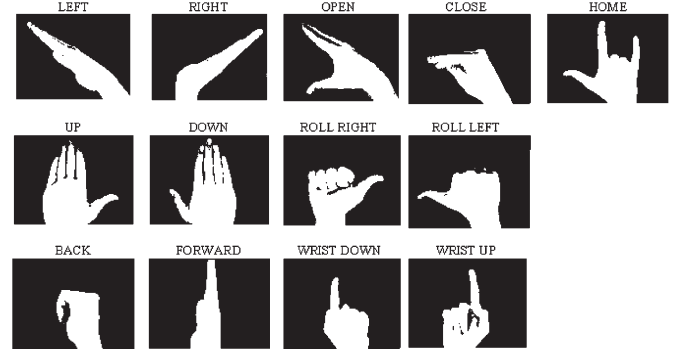

Fig. 1. Hand-gesture language.

Fig. 2. Illustration of a feature extraction: (a) hand gesture within bounding box and (b) 3×4 block partition.

to initialization compared to the crisp methods. The classical FCM algorithm is modified to handle feature-weighted clustering, and is supervised using a new cluster-labeling algorithm that is discussed in Section III. Also, discussed in this section is the problem of parameter estimation for the recognition system.

A. Feature Extraction

A database, denoted as BGU-R-DB and consisting of 13 static hand gestures, was constructed for training and testing purposes (Fig. 1). Preprocessing of the image starts with segmentation of the hand from the background using a threshold value τ to obtain a black and white image. The threshold value used is found through a parameter-search algorithm to be discussed in Section IV. Using a component-labeling algorithm, the largest component (assumed *a priori* to be the hand posture), is identified; and a bounding box is constructed around it to represent the segmented hand. The box is then partitioned into blocks. Although, the backgrounds of our gesture images are static with constant illumination, more complicated backgrounds can be handled by other methods such as that found in Stern and Efros [24].

The bounding box and a restriction on the height position of the posture makes the gesture position invariant in translation and size. A feature vector of the image is comprised of the aspect ratio of the bounding box, and the average intensity of each block (fraction of white pixels). 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, and the remaining represent block averages indexed row-wise from left to right.

The resultant feature vector of the example (Fig. 2) 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]. One can see that

the aspect ratio is 102, blocks 3 and 4 are close to zero (black), and blocks 6 and 11 are close to 255 (white).

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.

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

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).

B. Feature-Weighted FCM Gesture Classifier

In the weighted-feature FCM algorithm, a weighted-feature vector represents each gesture. The set of weighted-feature vectors are clustered for subsequent use in a recognition system. Note, that the particular clustering obtained depends on the number of clusters and the respective values of the feature weights. Let x_k be the weighted feature vector of the kth exemplar in a training set of gestures. Given q data patterns $X = \{x_1 \ \ldots \ x_k \ \ldots \ x_q\}$, and a fixed number of clusters c, the FCM algorithm finds v_i (the prototype weighted feature vector of cluster i), and μ_{ik} (the degree of membership of x_k in the ith cluster). This is done by minimizing a membership-weighted within-group sum-of-squared-errors objective function, where m is a weighting exponent on each fuzzy-membership value. In this application, the number of clusters should be set greater or equal to the number of gestures in the vocabulary.

After convergence of the FCM algorithm, each weighted feature vector x_k is assigned to a cluster by finding $\mu_{i'k} = \mathrm{Max}\{\mu_{ik}, i = 1, \ldots, c\}$. This simple method is selected to reduce computational complexity for real-time operation and to reduce the time taken for large-scale validation studies. Another fuzzy quantization approach is that of Karayiannis and Pai [25]. They offer a more complex method for transition from the fuzzy mode (where each training vector can be assigned to multiple clusters) to the crisp mode (where a training vector can be assigned to only one cluster). The method is based on an iterative shrinking of the set of fuzzy assignments. Once clusters are labeled by gesture class, a new gesture may be classified by selecting the cluster for which its membership value is maximal.

C. Parameter Estimation

Implementation of the feature-weighted FCM algorithm poses problems related to the selection of the number of clusters, selection of the "best" feature weighting, and the class labeling of clusters. Cluster labeling is addressed in the next section. The problem of optimal feature weighting, for the k-means clustering problem was approached by Modha and Spangler [26] using exhaustive search. We embed the search for more efficient weights within efficient parameter-estimation neighborhood-search (NS) routine discussed in Section IV. As a side note, feature weighting is actually a generalization of feature selection. Our set of free parameters is expanded beyond feature weights to include FCM variables such as, a fuzzy

weighting exponent, number of clusters, and image processing variables such as, black and white threshold and image partition resolution. Jointly solving for this extended parameter set by an NS algorithm is the subject of Section IV.

III. SUPERVISING THE FCM CLUSTERING ALGORITHM THROUGH HEURISTIC LABELING

The FCM clustering algorithm provides a fuzzy clustering of the training set of gestures, which are then converted into a crisp clustering. These clusters must then 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, i.e., "The Max Rule." This works well when the within-cluster objects are close to homogeneous. However, for samples that are badly clustered, this method fails (see Section III-D for an example). Badly clustered samples are the typical cases faced during the parameter-search initialization (described in Section IV). Thus, it is incumbent to devise a method for constructing a good cluster labeling. A valid cluster labeling must satisfy two conditions: 1) each cluster receives a unique label (a gesture name); and 2) each label must be assigned to at least one cluster. To evaluate a valid clustering, one needs a known ground-truth classification, i.e., labeled-gesture class data.

A. A Heuristic Labeling Algorithm (Alg-L)

A fast-labeling heuristic, denoted as Alg L, was developed to find near-optimal or optimal labeling. Given a set of crisp clusters and g class-labeled gestures, a matrix $\mathtt{N}$, called a gesture-cluster matrix of size $g \times c$ is constructed. Each row i corresponds to a gesture of known class i, and each column j corresponds to a cluster. Let n_{ij} be the ijth element of $\mathtt{N}$, which represents the number of gesture samples with known class i from the labeled training set that appears in cluster j. An example of a gesture-cluster matrix $\mathtt{N}$ is shown in Table I.

Heuristic Labeling Algorithm Alg-L

Step 1) For each column j of $\mathtt{N}$, find $\max\{n_{ij} | \forall i\} = n_{i'j}$. Mark the cells $(i'j)$. This represents an assignment of a label to each of the clusters.

Step 2) Let L_i be the number of marked cells in the ith row. For all $L_{i'} = 1$, remove row i' and column j associated with the marked cell. Let the set of remaining columns be $\mathtt{C}$. Assign label i' to cluster j.

Step 3) Let $\mathtt{R} =$ the set of rows for which $L_i = 0$. If $\mathtt{R} = \phi$, go to Step 5). Calculate the cost of moving the jth column mark to the submatrix $(\mathtt{R}, \mathtt{C})$. Set $n_{i'j} - n_{ij} = \bar{n}_{ij}$, for all $(i, j) \in (\mathtt{R}, \mathtt{C})$.

Step 4) Find $\min\{\bar{n}_{ij} | \forall (i, j) \in (\mathtt{R}, \mathtt{C})\} = \bar{n}_{i^*j^*}$. In case of ties, select the upper-left cell. Remove the mark from cell (i', j^*) and transfer it to cell (i^*, j^*). Set $L_{i*} = 1$. Return to Step 2).

Step 5) Stop. For all marked cells (i, j), assign label i to cluster j.

TABLE I
A REAL GESTURE-CLUSTER MATRIX, N

Gesture Class	Cluster														L_i
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
1	3	0	8	3	0	0	0	17	4	0	0	0	0	0	4
2	0	0	0	0	2	0	5	0	1	0	16	0	11	0	3
3	0	5	11	1	0	0	0	13	0	1	4	0	0	0	0
4	2	9	5	0	3	0	0	9	0	6	0	1	0	0	0
5	0	0	35	0	0	0	0	0	0	0	0	0	0	0	1
6	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	26	0	3	0	1	0	1	4	0	0	0	0	0
8	0	0	27	0	0	0	0	0	2	6	0	0	0	0	0
9	0	11	2	6	5	0	0	3	0	2	0	0	6	0	1
10	0	7	0	0	4	0	0	0	0	7	0	17	0	0	1
11	0	2	10	0	8	0	0	7	0	3	0	5	0	0	1
12	2	0	1	9	5	0	2	0	1	10	3	2	0	0	2
13	0	0	17	0	4	0	0	3	3	5	0	3	0	0	0

Alg-L terminates in a finite number of steps with a valid labeling. Since matrix (R, C) is finite, at Step 4), a cluster is labeled permanently, and a row and column are removed from (R, C). Only one mark remains in each column throughout the algorithm. At the end, no rows remain with $L_i = 0$. The computational complexity of Alg-L is $O(nm \log(nm))$, where $n =$ number of gesture classes and $m =$ number of clusters.

B. An Optimal Labeling Algorithm (LP-L)

To test the performance of the heuristic Alg-L, optimal labeling must be known. An optimal labeling is one in which the accuracy of the classifier for the training set is maximal over all feasible (valid) labelings. This problem is formulated as a linear program, and denoted as LP-L. Define an assignment variable x_{ij} equal to one, if gesture class i label is assigned to cluster j and zero, otherwise. Assume all clusters are nonempty, and for each gesture-type samples appear in at least one cluster. Denote C and G as the set of clusters and gesture class labels, respectively.

Optimal Labeling Algorithm LP-L

$$\max Z = \sum_{j \in C} \sum_{i \in G} n_{ij} x_{ij} \qquad (2)$$

s.t.:

$$\sum_{i \in G} x_{ij} = 1, \qquad j \in C \qquad (3)$$

$$\sum_{j \in C} x_{ij} \geq 1, \qquad i \in G \qquad (4)$$

$$x_{ij} = 0, 1, \qquad j \in C, \, i \in G. \qquad (5)$$

Constraint (3) ensures that each cluster j receives exactly one label. Constraint (4) ensures that each label is assigned at least to one cluster. In the optimal solution, the set of assignment variables with $x_{ij}^o = 1$ represents the number of correct assignments of gesture samples to clusters. These variables enter actively in the objective function so that the summation in (2) represents the total number of gestures in the training set

correctly classified. Note, that if one permutes the columns of N, according to the optimal solution $\{x_{ij}^o\}$, the familiar confusion matrix is obtained. After normalizing Z by the total number of samples in the training set, the recognition rate is obtained. Recognition accuracy A is defined as

$$A = \frac{\text{number of samples correctly classified}}{\text{number of gesture samples}} \times 100. \qquad (6)$$

Although LP-L is an integer linear program the integer constraints (5) can be relaxed, with integer solutions being assured, as the coefficient matrix is unimodular.

C. Performance of Alg-L

A set of test cases was generated to compare the results of Alg-L to the optimal solution obtained by LP-L. Each test case includes an equal number of samples per gesture, for a given vocabulary of gestures. For each test case, a matrix N is generated. The first case was created to give 100% recognition accuracy. The remaining cases were degenerated gradually, to produce matrices with reduced accuracies. Each test case was input to both algorithms and the resultant accuracies where recorded. For clarity, the following notation is defined: $g =$ number of gesture classes, $c =$ number of clusters, $s =$ number of samples of each gesture, $q =$ number of samples in a training or testing set $(s \times g)$, $T =$ number of test cases for Alg-L. To create the first case $(T = 1)$, N is filled as follows: 1) Set $n_{ij} = 0$ for all i, j; 2) Set $n_{ij} = s$ for all i, j where $i = j$; 3) Set $n_{ij} = s$ for all i, j where $i = g$ and $j > g$; 4) Calculate A using both LP-L and Alg-L.

The degeneration process applied to matrix N is an iterative procedure, where at each iteration, one sample is moved from one cluster to another at random. The probability that a sample is selected for transfer from some cluster to another is $P_r = 1/(g \times c)$.

Using $g = 13$, $c = 16$, $s = 30$, and $T = 200$, the average accuracy of Alg-L, and its average percent error from the optimal is shown in Table II. Optimal accuracy for each test case was compared to the accuracy of the heuristic Alg-L to determine the error of the heuristic Alg-L. The test cases were

TABLE II
AVERAGE PERCENT ERROR OF Alg-L

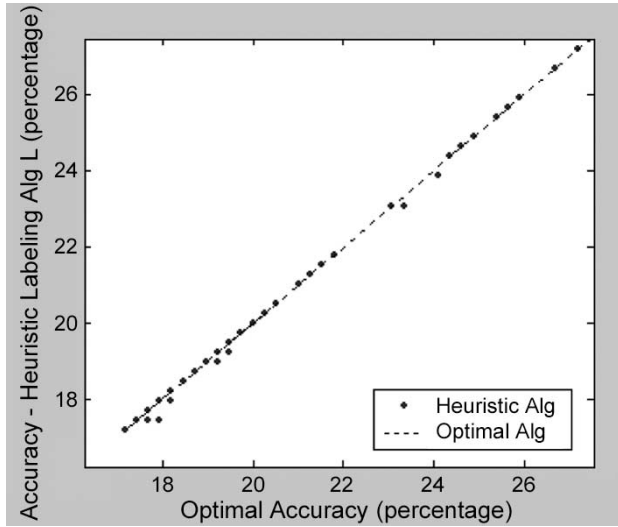| Accuracy Range	0–10%	11–20%	21–30%	31–40%	41–50%	51–60%	61–70%	71–80%	81–90%	91–100%
Test cases	0	62	32	23	18	17	11	7	8	6
Ave. A of Alg-L	0	18.65	25.21	35.95	44.54	55.22	65.97	75.16	85.38	95.21
Ave. A of LP-L	0	18.69	25.25	35.95	44.54	55.22	65.97	75.16	85.38	95.21
Ave. Err. Alg-L(%)	0	0.21	0.16	0	0	0	0	0	0	0

Fig. 3. Dispersion of Alg-L performance from the optimal accuracy.

then grouped according to optimal accuracy ranges of 10% (see row 1). The average heuristic errors for each group were then calculated and placed in row 5. Average accuracy for the test cases within each group for Alg-L and LP-L are recorded in rows 3 and 4, respectively.

As can be seen, Alg-L obtained the optimal solution for all cases with high accuracy solutions (30%–100%). However, Alg-L performs weakly compared to the optimal LP-L algorithm when the optimal accuracy is between 10% and 30%. The fact that the average percent error, for the 0%–10% accuracy category is zero requires a deeper insight. The worst case labeling accuracy occurs for a very high T, when the gesture-cluster matrix $\mathbb{N}$ is totally degenerate. In this case, the samples are uniformly distributed in $\mathbb{N}$ having $n_{ij} = s/c$ samples in each cell (i, j). Both labeling algorithms will assign an arbitrary label to each cluster; therefore, only s/c samples per cluster belong to the label assigned to that cluster. Hence, the correct number of samples assigned to clusters is s. Using (6), the recognition accuracy is $A = (s/q) \times 100 = (1/g) \times 100$.

This case provides a lower bound on the accuracy A. For the 13-gesture example, the lower bound is 7.69%. Consequently, it is impossible to find samples in the 0%–7.69% accuracy range. Cases slightly above the accuracy lower bound of 7.69% (those from 7.69% to 10%) are very unlikely to appear for the given T. This explains the reason no samples were generated in the first accuracy interval in Table II. This difference of performance in the labeling algorithms, for the accuracies between 11% and 30% can be visualized in a graph of optimal accuracy versus Alg-L (Fig. 3).

TABLE III
CLUSTER-LABEL ASSIGNMENTS

Cluster	1	2	3	4	5	6	7	8	9	10	11	12	13	14	A (percentage)
Label (max-rule)	1	9	5 or 6	12	11	-	2	1	1	12	2	10	2	-	Invalid
Label (Alg-L)	6	9	5	12	11	3	7	1	13	8	2	10	2	4	29.45
Label (LP-L)	4	9	6	12	11	5	7	1	13	8	2	10	2	3	29.89

The dashed line represents an optimal response to LP-L. The dots show the response of Alg-L.

D. Labeling of a Real Gesture-Cluster Matrix, $\mathbb{N}$

As an example, consider the real gesture-cluster matrix of Table I constructed for an independent-user system, with a total of 455 samples of 13 gesture classes and the following parameters: threshold = 220, fuzzy weight exponent = 4, clusters = 14, block partition rows and columns = 2, and the weights = (0.252 0.005 0.248 0.265 0.231). This case was selected to represent a particularly bad clustering, as each cluster contains many different gesture classes. Using the max rule, each cluster is labeled by the most popular gesture type (see bold, Table I) assigned to it. This results in an invalid clustering (see Table III) with multicluster labeling (clusters 1, 8, 9 labeled 1) and unused labels.

IV. PROBLEM FORMULATION—THE PARAMETER SEARCH ALGORITHM

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. Recognition accuracy, as defined by (6), is determined as a function of a set of input parameters of the system. The process of searching for optimal parameters for the combined image-processing/supervised-FCM system is shown in the flow chart of Fig. 4. The output of the process is a near-optimal set of parameters achieved by maximizing the recognition accuracy. The procedure used is an NS algorithm. A survey of NS techniques can be found in [27].

A. Input Parameter Vector p

Denote the vector $p \in R^n$ as the set of input parameters (Table IV). Two types of input parameters are used: image-processing features (block-partition size, b/w threshold, feature weights aspect ratio, and grayscale block levels) and FCM parameters (number of clusters and weighting exponent).
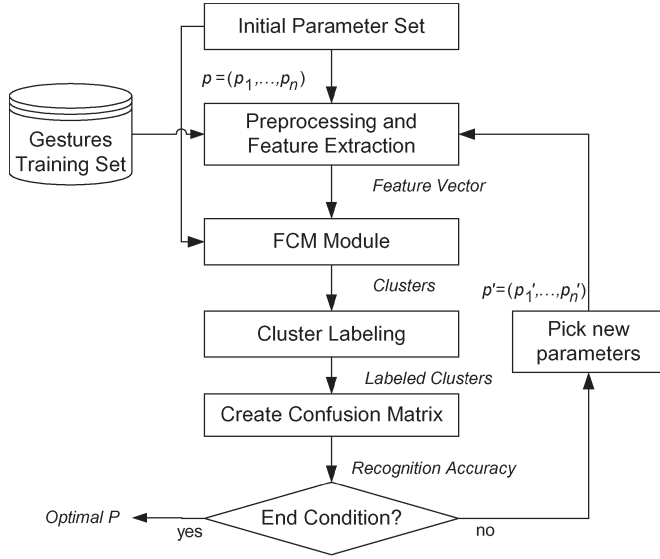
Fig. 4. Supervised FCM gesture-recognition algorithm with parameter search.

B. Neighborhood Solutions $N(p)$

For any feasible solution $p = (p_1, \ldots, p_n)$ for the recognition system, define a set $N(p)$ of neighboring solutions. The number of neighbors of p is $2n$ as each parameter is incremented up and down (wrap-around is used when boundary values are exceeded). The set of feature-weight parameters are updated in a special way because of their interdependence though (1). 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 pattern space is variable and depends on the minimum and maximum block-partition values. For block-partition values ranging from 2 to 8, the number of feature-weight parameters can vary from 5 to 65, resulting in pattern spaces of dimension 10 to 70. To handle such variable-length parameter vectors, we consider p_4 and p_5 as control parameters that 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 $\{w_i : i = 1, \ldots, v\}$ be the current set of weights. To find the neighbor values of a feature weight w_j, increment w_j up and down by the discrete gradient Δj. Since feature-weight normalization is necessary to ensure that (1) is satisfied, the new neighbor feature weights are:

$$w_i(\pm \Delta j) = \begin{cases} \frac{w_i}{1 \mp \Delta j}, & i \neq j \\ \frac{w_j \pm \Delta j}{1 \mp \Delta j}, & i = j. \end{cases} \quad (7)$$

C. Neighborhood-Search Algorithm

The NS algorithm starts with an initial solution p^0. To determine the accuracy A associated with p, define a mapping $A : p \rightarrow A$. Determination of the functional value of A, for a given solution p, requires extraction of a new set of image features, executing the FCM algorithm, cluster label assignments (Alg-L), gesture classification, and analysis of the confusion

matrix to determine the recognition accuracy (Fig. 4). The NS algorithm pseudocode is shown below.

> **Algorithm** neighborhood search;
> 1. **Begin**
> 2. Create an initial feasible solution $p^0 = (p_1^o, \ldots, p_n^o)$
> 3. *local_maxima = false*
> 4. **Repeat**
> 5.   **Begin**
> 6.   Find $A(p')$ for all $p' \in N(p)$
> 7.   Let $p'' = \arg\max\{A(p')|\forall p' \in N(p)\}$,
> 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** $A(p'') = 100\%$ **or** *local_maxima = true* **do**
> 17.   Output p'', the local or global max solution
> 18. **END**

The main idea behind the NS 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 this plateau case, 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 maximal.

The resulting recognition accuracy function A is a nondecreasing function of the number of iterations k, i.e., $A(p^k) \geq A(p^{k-1})$, 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 A is bounded above by 100%, termination in a finite number of steps is guaranteed. Detailed proofs may be found in [28].

Since most initial solutions generated by the starting-solution heuristic (see next section) for the NS algorithm are above 30% accuracy, Alg-L's accuracy is almost always optimal. Therefore, Alg-L is used exclusively in the cluster-labeling stage of the NS algorithm.

D. Determination of Starting Solutions for the NS

The NS algorithm is basically a gradient-search method, but because the objective function is not well behaved with an

TABLE IV
PARAMETER DEFINITION

Parameter	Meaning	Values
p_1	Number of Clusters, c	p_1 = g, g + 1, c_{max}
p_2	Weighting Exponent, m	p_2 = 1.5, 1.75, …,4
p_3	B/W threshold, τ	p_3 = 0, 1,…,255
p_4	Number of rows for image partition, R_b	p_4 = 2, 3,…,8
p_5	Number of columns for image partition, C_b	p_5 = 2, 3,…,8
p_6	Weight of the aspect ratio, w_j	p_6 = 0, 0.1,…,1
$p_7,…,p_i,…,p_n$	Weights of the image block features, w_i	p_i = 0, 0.1,…,1

unknown analytical form, classical gradient-search algorithms cannot be used. The search is conducted by examining 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).

A common problem with gradient-search methods is that the local extrema reached is very sensitive to the starting solution used. To help alleviate this difficulty, we introduce the element of exploration by using a set of starting solutions. Nine starting solutions were generated using the heuristic rules shown below. The selection of the number of clusters and block-partition resolutions was based on a stratified sample of the solution space. The numerical numbers used in the description are based on the 13 gesture example (Fig. 1).

Procedure: Initial Solution Generator

1) **Block partition variables**: R_b and C_b.

   To start from different regions of the solution space, generate three different partitions: course, medium, and fine, represented by square partitions of size 2, 5, and 8, respectively.

2) **Weights of aspect-ratio and image-block features**: w_j.

   The number of features for each block partition will be v. All feature values are scaled to lie in the interval [0, 255]. For each of the three block partitions, create a global-feature dataset for each feature j over all the data (number of samples/gesture/user) × (gestures) × (users). The jth weight is calculated using (8), where σ_j is the standard deviation of the jth feature.

$$w_j = \frac{1}{\sigma_j} \sum_{i=1}^{v} \frac{1}{\sigma_i} \qquad (8)$$

   Note, that the weights are inversely proportional to the standard deviation of each feature variable. This formula gives less weight to unstable features, and more to stable features. Its derivation preserves the unity constraint (1). It was found that using the standard deviation, in lieu of the variance, in (8) provided better performance and, therefore, was employed.

3) **Number of clusters**: c.

   Because of the wrap-around operation in the search algorithm, three cluster values equally spread over the

range of possible values (from $\underline{c}$ to $\bar{c}$) were determined using (9).

$$\left\{ \underline{c}, \quad \left(\frac{1}{3}\right)(2\underline{c} + \bar{c} + 1), \quad \left(\frac{1}{3}\right)(\underline{c} + 2\bar{c} + 2) \right\} \qquad (9)$$

where, $\underline{c}$ = the minimal number of clusters considered (normally set to g, number of gestures), and $\bar{c}$ = the maximal number of clusters considered. Using this rule, for the case where $g = 13$, $\bar{c} = 22$, the cluster sizes are 13, 16, and 20 (after rounding to the closest integer).

4) **FCM weighting exponent**: m.

   For all initial solutions, a value of 2 is used for the weighting exponent of the FCM algorithm. This value is a typical value used by most researchers to represent a good level of fuzziness in the FCM algorithm [29].

5) **B/W threshold**: τ.

   Using a sample of 50 images from the gesture image database, Otsu's method [30] was applied to the average image to find the best initial threshold τ. This method is applicable under the assumption of constant light intensity for all captured images used for the testing, training, and recognition (which is the case in hand). For situations of varying illumination, τ should not be considered as a parameter, but should be determined online for each image captured using appropriate adaptive-threshold methods.

   Using the above procedure the nine "heuristic" (non-random) solutions will start from different regions of the solution space. Among the solutions obtained by the NS, for each of the starting solutions, the best locally optimal solution will be selected.

E. Example Runs of NS Algorithm for the 13 Gesture DB (BGU-R-DB)

An example run is conducted to illustrate the performance of the NS algorithm using independent user system data in BGU-R DB. For this database, we extracted 35 samples for each of the 13 gestures, to obtain a training set of 455 samples. Using initial cluster-value limits of 13 and 22 in (9), and weights according to (8), the starting solution in Table V(a) was found. Using each starting solution, the corresponding final solution found by the NS algorithm is shown in Table V(b). Table VI presents the number of iterations to convergence, which range from 1 to 12 iterations. The best solutions appear in bold. The

TABLE V
PARAMETER-SEARCH RESULTS FOR EXAMPLE RUN (a) INITIAL SOLUTIONS (TRAINING) (b) FINAL SOLUTIONS (TRAINING)

(a)

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

(b)

Sol	c	m	τ	R_b	C_b	w_i	A (percentage)
1	22	2	146	2	2	0.775 0.063 0.062 0.048 0.053	97.80
2	16	1.75	146	2	2	0.73 0.075 0.074 0.058 0.063	97.80
3	20	1.75	146	2	2	0.7 0.083 0.082 0.065 0.07	98.02
4	22	1.75	146	5	4	0.604 0.022 0.017 0.016 0.018 0.029 0.018 0.016 0.019 ...	99.12
5	20	2	146	2	5	0.729 0.034 0 0.026 0.028 0.045 0.028 0.025 0.031 0.024 ...	99.78
6	20	1.75	143	3	4	0.728 0.026 0.021 0.02 0.022 0.035 0.022 0.019 0.023 ...	99.34
7	22	2	146	8	7	0.354 0.014 0.013 0.011 0.011 0.012 0.012 0.016 0.018 ...	99.12
8	16	2	146	8	2	0.653 0.026 0.025 0.02 0.02 0.021 0.021 0.03 0.033 ...	98.46
9	20	1.75	139	8	2	0.503 0 0.038 0.031 0.031 0.033 0.033 0.046 0.051 ...	99.56

sequence of solutions starting from initial solution 6 is shown in Table VII and Fig. 5. From this run, one sees the parameter changes throughout the convergence profile. In Table VII we note, the accuracy for iterations 6, 7, and 8 are identical as the search is checking a plateau. Actually, we are searching on a level ridge in the coordinate direction of the threshold parameter when we discover an upward move in the direction of one of the weights (iteration 9).

V. PERFORMANCE TESTING

Three gesture databases were used to test the NS algorithm. The first was a basic 13-gesture robotic-control vocabulary, BGU-R DB (Fig. 1). The poses were captured for different users and hand orientations to obtain samples that were representative of real-time use variation. The variability of the poses can be represented by the feature variances, which varied from 30 to 13 000. A second database, denoted as BGU-ASL DB, was constructed to test the effect gesture scale up from 13 to 24. Also, a comparative evaluation was made using the same database used to test the Gripsee system [10], which we denote as

Gripsee-ASL DB. In addition, we tested the performance of the method using Tele-Gest [31], a real-time implementation of a supervised FCM classifier for teleoperated control of a remote fixed-robot manipulator.

A. BGU-R DB

For the BGU-R DB, two different types of systems were used to train and test user-dependent (D) and independent (I) recognition systems. The D and I systems are defined as the systems that are trained by single and multiple users, respectively. A k-fold cross validation was used [32] to partition the original sample set into k subsets of equal size. Each subset represents a different session. Each subset in turn is used for testing and the remainder for training. The advantage is that all the data can be used for training and none has to be held back as a separate test set. Each partition of the data into a training set and a testing set will be defined as a "session." The partition used was $k = 4$. The misclassification average is defined as the k-fold crossvalidation estimate of the true error rate. For the hand gesture system, a corpus of $S = 520$ samples were used, 40 samples for each of the 13 gestures.

TABLE VI
PARAMETER SEARCH—INITIAL AND FINAL ACCURACIES

Solution no.	1	2	3	4	5	6	7	8	9
Initial accuracy (percentage)	84.84	96.04	95.6	77.8	**88.35**	**77.8**	83.3	85.93	90.33
Final accuracy (percentage)	97.8	97.8	98.02	99.12	**99.78**	**99.34**	99.12	98.46	99.56
Number of iterations	3	1	2	7	**12**	**9**	4	3	7

TABLE VII
PARAMETER-SEARCH RESULTS FOR SOLUTION 6

| Iter	c	m	τ	R_b	C_b	w_i	A (percentage)
1	20	2	146	5	5	0.343 0.03 0.024 0.023 0.025 0.04 0.025 0.022 0.027 0.021 0.026 0.028 0.024 0.06 0.025 0.025 0.021 0.022 0.035 0.026 0.023 0.021 0.02 0.021 0.022 0.022	77.80
2	20	1.75	146	5	5	0.343 0.03 0.024 0.023 0.025 0.04 0.025 0.022 0.027 0.021 0.026 0.028 0.024 0.06 0.025 0.025 0.021 0.022 0.035 0.026 0.023 0.021 0.02 0.021 0.022 0.022	96.04
3	20	1.75	146	5	4	0.383 0.034 0.027 0.026 0.028 0.045 0.028 0.025 0.03 0.023 0.029 0.031 0.027 0.067 0.028 0.028 0.023 0.025 0.039 0.029 0.026	98.46
4	20	1.75	146	5	4	0.445 0.03 0.024 0.023 0.025 0.04 0.025 0.022 0.027 0.021 0.026 0.028 0.024 0.06 0.025 0.025 0.021 0.022 0.035 0.026 0.023	98.46
5	20	1.75	146	3	4	0.584 0.04 0.032 0.03 0.033 0.053 0.033 0.029 0.036 0.028 0.034 0.037 0.032	98.68
6	20	1.75	146	3	4	0.633 0.035 0.028 0.027 0.029 0.047 0.029 0.026 0.031 0.024 0.03 0.033 0.028	98.90
7	20	1.75	145	3	4	0.633 0.035 0.028 0.027 0.029 0.047 0.029 0.026 0.031 0.024 0.03 0.033 0.028	98.90
8	20	1.75	143	3	4	0.633 0.035 0.028 0.027 0.029 0.047 0.029 0.026 0.031 0.024 0.03 0.033 0.028	98.90
9	20	1.75	143	3	4	0.728 0.026 0.021 0.02 0.022 0.035 0.022 0.019 0.023 0.018 0.022 0.024 0.021	99.34
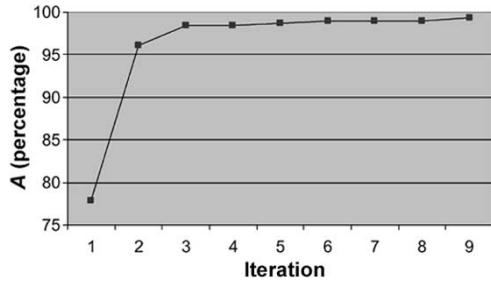
Fig. 5. Recognition accuracy versus iterations for solution 6.

TABLE VIII
NUMBER OF SYSTEM-USER TESTS

System	User Type			
	Owners	Experienced	Novice	No. Systems
D-Dependent	1	6	12	7
I-Independent	7	7	12	1

Fig. 6. BGU-ASL DB.

The samples were 320×240 pixel grayscale images. Three types of subjects were used in the experiments: owners (O), experienced users (E), and novice users (N). Owners trained all I and D systems and are also used to test these systems. Experienced users are users that test systems that were trained by others. These users were reused owners who play the role of experienced users at this stage. Novice users are new users who have never seen, trained, or tested a system. Seven owners and twelve novice users were elicited to test the D and I systems. The number of test runs for each system–user combination is shown Table VIII.

B. Scale-Up to 24 Gestures (BGU-ASL DB)

A test was also made to assess the effectiveness of the NS algorithm for larger databases. The 13 Gesture DB was scaled up to 24 static gestures (see Fig. 6), which includes all the letters of the ASL alphabet, except "j" and "z" (which are dynamic). Our own gesture-gallery database (denoted as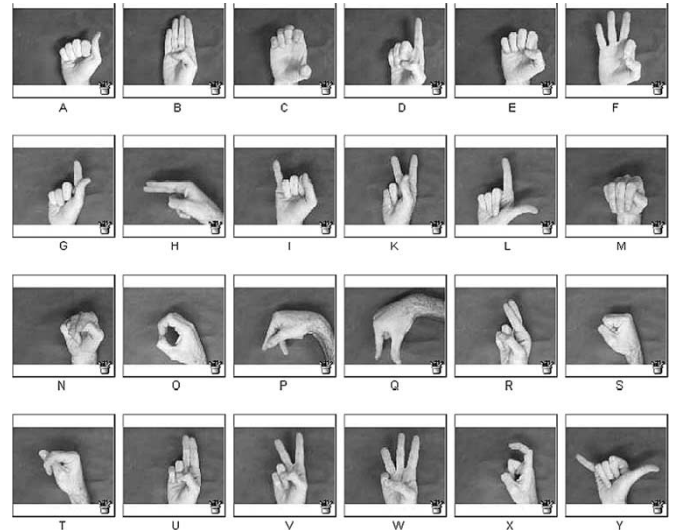 BGU-ASL DB) was created by capturing images of size 320×240 performed over a black uniform background. One owner performed the gestures, using as a guide the postures appearing in the ASL Browser developed by Michigan State University [33]. For each letter, a sample set of 30 and 10 grayscale images was used to train and test a dependent system, respectively.

C. Comparative Evaluation to Gripsee (Gripsee_ASL DB)

Although the recognition algorithm is not the main focus of this research, it was decided that a comparative test be made with a recognition algorithm found in the literature. The closest work found, comparable with our gesture assumptions using 2-D static-robotic gesture postures, was that of Gripsee [10]. We used their "simple background" data set, although their hand poses were more complex than ours, consisting of cuffs and sleeves at the base of the hand. Two hundred and forty grayscale images, each of size 128×128, with black backgrounds, created by 24 users, were downloaded from the
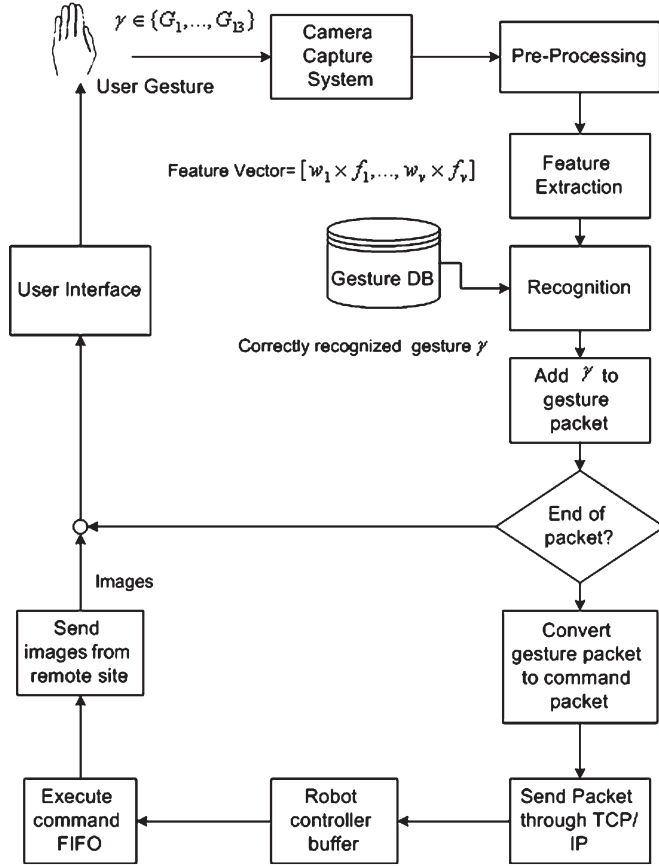
Fig. 7. System flow diagram of Tele-Gest.

Fig. 8. User interface with real robot.
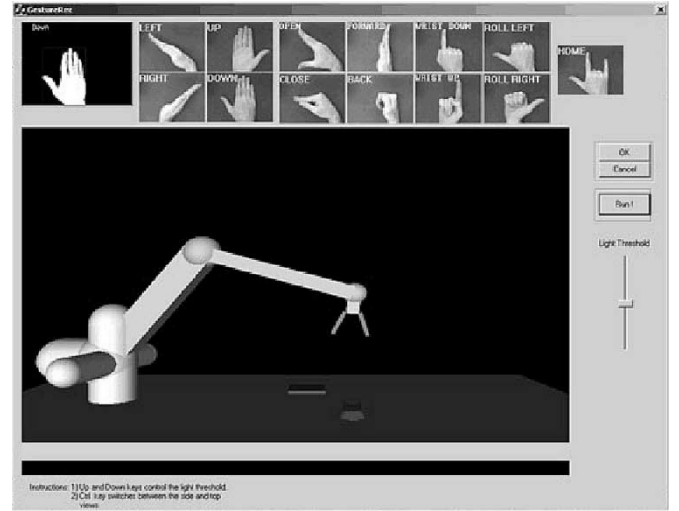
Fig. 9. User interface with virtual robot.

Sebastien Marcel Static Hand Posture Database [34]. The images were of uniform background hand postures of the ASL alphabetic letters (a, b, c, d, g, h, i, l, v, y).

D. Tele-Gest Real-Time Implementation

Tele-Gest is a real-time implementation of a supervised FCM classifier for teleoperated control of a remote fixed-robot manipulator [31]. To control a robot movement, the user evokes a gesture from a gesture vocabulary by placing his/her hand under a video imager. The gesture image is classified, using the recognition module described in Section II, and 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. Recognition speed t was estimated empirically using the following time parameters: t_c—camera-capture and frame-grabber time, t_p—preprocessing time, t_r—FCM recognition time. The mean time of 20 trails is: $t = t_c + t_p + t_r = 0.04 \text{ ms} + 7.58 \text{ ms} + 0.81 \text{ ms} = 8.43 \text{ ms}$. This time depends on the computer and frame grabber used, so comparison with other reported times is difficult. However, this time is sufficient to allow for real-time operation. Kjeldsen [35] reports 2 Hz for his pose real-time recognition algorithm using a 100 MHz PC.

A flow diagram of the Tele-Gest implementation is shown in Fig. 7. Upon presentation of the robotic scene in the user's interface, a gesture γ (selected from the gesture vocabulary:
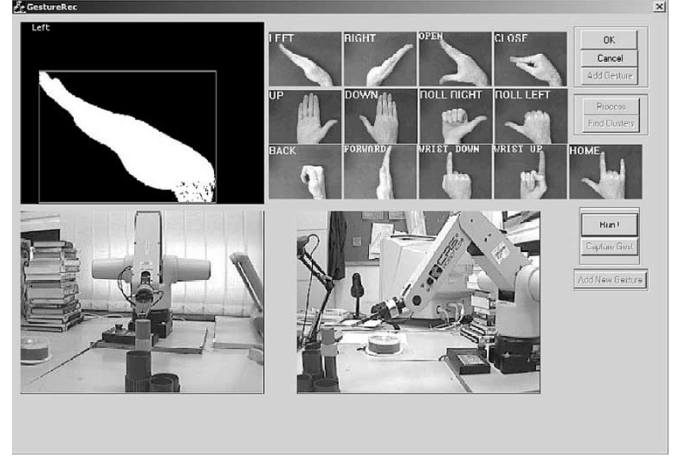
$\{G_1, G_2, \ldots, G_{13}\}$ in the case of BGU-R DB) is evoked. The feature string of the captured image of the gesture is recognized and sent to the robot PC server using a TCP/IP protocol. After the robot executes the command, two camera views of the robot environment are transmitted back to the user's interface. Fig. 8 is a screen dump of the hand-gesture robotic-control interface. The interface contains feedback to the user of the gesture's recognition status, a gesture reminder display, and the two views of the robot. Because of the sluggish movement time of the real robot, for large-scale usability testing a second interface was designed, consisting of a virtual-reality robotic model. (see Fig. 9).

VI. RESULTS AND DISCUSSION

A. Results and Hypotheses Tests (BGU-R DB)

For each of the user–system combinations in Table VIII, mean recognition accuracies were calculated from the results of the k-fold crossvalidation runs ($k = 4$). The mean-recognition accuracies between systems were compared using a two-tailed t-test. Table IX shows the hypothesis formulated, the

TABLE IX
PERFORMANCE COMPARISON BETWEEN SYSTEMS

No.	Hypothesis	n_1	n_2	x_1 (%)	x_2 (%)	S_1	S_2		Significance (percentage)
1	$A(D,E) > A(D,N)$	21 840	43 680	96.01	94.59	0.0383	0.0512	TRUE	0
2	$A(I,O) > A(I,N)$	3640	6240	98.21	95.77	0.0175	0.0405	TRUE	0
3	$A(D,O) > A(D,E)$	3640	21 840	98.90	96.01	0.0109	0.0383	TRUE	0
4	$A(D,O) > A(I,O)$	3640	3640	98.90	98.21	0.0109	0.0175	TRUE	0.69
5	$A(I,E) > A(D,E)$	3640	21 840	98.21	96.01	0.0175	0.0383	TRUE	0
6	$A(I,N) > A(D,N)$	6240	43 680	95.77	94.59	0.0405	0.0512	TRUE	0.0049

TABLE X
SYSTEM-RECOGNITION ACCURACY

Type of User	Type of System	
	Dependent (D)	Independent (I)
Owners (O)	98.9%	98.2%
Experienced (E)	96.0%	98.2%
Novice (N)	94.6%	95.7%

population used to compare each side of the hypothesis, recognition accuracy, variance, hypothesis result, and significance level. Recognition accuracy of system x, tested with user y, is represented by $A(x, y)$. A summary of the important results is shown in Table X. When the systems were tested using their own trainers, mean accuracy of D was better than the I system (98.9 over 98.2%). This is as expected since any learning system should have better performance when tested with its trainer. For experienced users, the opposite was true, testing recognition accuracies where better for I than D systems (98.2 over 96.0%). This also is expected as experienced users were testing systems trained by others. Here, the I system was trained with a wide variation of hand-gesture samples, and as a result, it had better generalization properties. These results were statistically significant. Similarly, novice user's testing accuracy was also better for I than D systems, resulting in 95.7 and 94.6%, respectively.

When compared to previous runs using five novice users, slightly better results were obtained as expected (96.1 and 95.1%). These values had a statistical significance at the 0.005 level. Again, these results are for novice users who have neither trained systems nor have had experience using them. Previous research [21] indicates that novice users can reach 98%–99% accuracy after several trials.

In addition to recognition accuracy, we examined the optimal number of clusters found by the NS algorithm. The final cluster values for the different D and I systems varied from 14 to 22.

B. Scale-Up Results (BGU-R DB)

Using the NS algorithm for the larger BGU ASL-DB, recognition accuracies of 96.25 and 93.33% were obtained for training and testing, respectively. This was lower than the accuracy obtained by our BGU-R DB consisting of only 13 gestures. Other researchers have reported this degradation in performance. For example, in [10], it is reported that performance becomes faster and more reliable when fewer gestures are used.
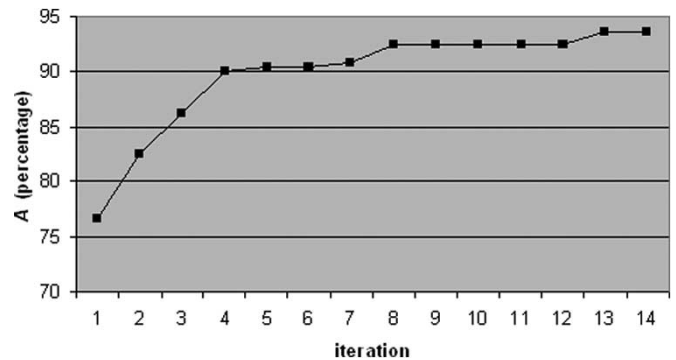

Fig. 10.   Recognition accuracy versus iterations using Gripsee-ASL DB.

C. Comparative Evaluation to Gripsee (Gripsee-ASL DB)

Applying our NS algorithm to the Gripsee-ASL DB, we obtained an accuracy of 93.75%, which is comparable to Gripsee's [10] published result of 93.8%. In this work, published in 1999, they used 418 samples with a black-and-white uniform background. In an earlier work in 1996 [36], they used 208 samples and obtained an accuracy of 93.3. The final parameters, (except for the weights), where: $R_b = 3$, $C_b = 4$, $c = 15$, $m = 2$, and $\tau = 98$. The convergence history to obtain this result is shown in Fig. 10.

D. Usability Test Results for Tele-Gest

To test the usability of the Tele-Gest implementation, we devised a task using hand gestures to control a robot to remove a marker inside a box and place it in a second box. Four novice users performed five trials of this task for both D and I systems. For each system, the users practiced the gestures in the BGU-R DB vocabulary for 2 to 10 min until they became familiar with the gesture vocabulary. The time to complete each trial was recorded to represent the performance of each user.

The model for the learning curve is $Y_n = Y_1 n^{-b}$, where Y_n is the estimated value of the completion time in seconds on the nth trial, n is the trial number, Y_1 is the time of the first trial, and b is $\log r / \log 2$ where r is the learning rate. The best-fit learning-curve equations for the D and I systems are $Y_n = 474.8 \, n^{-0.226}$ and $Y_n = 524.26 \, n^{-0.296}$, respectively. The learning curves show that the user would learn the "pick and place" task faster using the I system (learning rate of 0.81) than using the D system (0.85). Standard task times were reached relatively quickly, in the order of five or six trials.

VII. CONCLUSION

A methodology for tuning-system parameters is presented. To test the methodology, we address the difficult problem of simultaneous calibration of the parameters of the image processing–FCM components of a hand-gesture recognition system. In addition, we proffer a method for supervising the FCM algorithm using linear programming and heuristic labeling. The parameters of the image-processing and clustering algorithm were simultaneously calibrated using a neighborhood parameter-search routine. The search algorithm exhibited fast convergence (in the order of ten iterations) to reach recognition accuracies within several percent of optimal. We also incorporated feature weighting in the FCM algorithm.

Comparison of user-dependent and user-independent systems using a database of 13 gestures were made. When the system was tested with their own trainers, recognition accuracies of 98.9 and 98.2% were found for dependent and independent systems, respectively. These results are statistically significant at the 0.007 level. For experienced users testing systems they had not trained, the opposite was true, testing recognition accuracies were better for user-independent than user-dependent systems (98.2 over 96.0%). These results are statistically significant at the 0.00 level.

Further tests were made to assess the effectiveness of the NS algorithm for larger databases. Using 24 static postures from the ASL letter signs, the accuracy obtained was slightly lower than that of our BGU-R DB (only 13 gestures). Other researchers have also reported degradation in performance as the number of gestures is increased. Comparison of our system to that of the Gripsee system, resulted in 93.75%, which is comparable to Gripsee's published result of 93.8%. Also, Tele-Gest, a real-time implementation of our FCM gesture classifier, for teleoperated control of a remote fixed-robot manipulator was described. Usability tests on a simple pick-and-place task showed fast learning rates reaching standard times in five or six trials.

Our near-optimal parameter-search procedure is easily extended to systems with larger parameters and more complex hand-gesture recognition systems. The problem is not unique to hand-gesture-recognition systems, but is shared by other human–machine systems as well. Thus, the methodology presented here for automating system setups has far wider application. Ongoing research is aimed to improve the NS algorithm. For example, using a dynamic step-size rule to approach convergence slowly by decreasing the step size.

REFERENCES

[1] A. Katkere, E. Hunter, D. Kuramura, J. Schlenzig, S. Moezzi, and R. Jain, "ROBOGEST: Telepresence using hand gestures," Visual Computing Lab, California Univ., San Diego, Tech. Rep. VCL-94-104, Dec. 1994.

[2] D. Kortenkamp, E. Huber, and R. P. Bonasso, "Recognizing and interpreting gestures on a mobile robot," in *Proc. Artificial Intelligence (AAAI) Workshop*, Portland, OR, 1996, pp. 915–921.

[3] C. W. Ng and S. Ranganath, "Real-time gesture recognition system and application," *Image Vis. Comput.*, vol. 20, no. 13–14, pp. 993–1007, Dec. 2002.

[4] I. Y. Gu and T. Tjahjadi, "Multiresolution feature detection using a family of isotropic bandpass filters," *IEEE Trans. Syst., Man, Cybern. B*, vol. 32, no. 4, pp. 443–454, Aug. 2002.

[5] K. Abe, H. Saito, and S. Ozawa, "Virtual 3-D interface system via hand motion recognition from two cameras," *IEEE Trans. Syst., Man, Cybern. A*, vol. 32, no. 4, pp. 536–540, Jul. 2002.

[6] X. Yin and M. Xie, "Estimation of the fundamental matrix from uncalibrated stereo hand images for 3-D hand gesture recognition," *Pattern Recogn.*, vol. 36, no. 3, pp. 567–584, Mar. 2003.

[7] T. S. Huang and V. I. Pavlovic, "Hand gesture modeling, analysis, and synthesis," in *Proc. Int. Conf. Automatic Face and Gesture Recognition*, Zurich, Switzerland, 1995, pp. 73–79.

[8] J. Segen, "Controlling computers with gloveless gestures," in *Proc. Virtual Reality Systems Conf.*, New York, 1993, pp. 66–78.

[9] D. Franklin, R. E. Kahn, M. J. Swain, and R. J. Firby, "Happy patrons make better tippers—Creating a robot waiter using Perseus and the animate agent architecture," in *Proc. Int. Conf. Automatic Face and Gesture Recognition*, Killington, VT, 1996, pp. 14–16.

[10] M. Becker, E. Kefalea, E. Maël, C. V. D. Malsburg, M. Pagel, J. Triesch, J. C. Vorbrüggen, R. P. Würtz, and S. Zadel, "GripSee: A gesture-controlled robot for object perception and manipulation," *Auton. Robots*, vol. 6, no. 2, pp. 203–221, Apr. 1999.

[11] R. Cipolla and N. J. Hollinghurst, "Human–robot interface by pointing with uncalibrated stereo vision," *Image Vis. Comput.*, vol. 14, no. 3, pp. 171–178, Apr. 1996.

[12] D. Guo, Y. H. Yan, and M. Xie, "Vision-based hand gesture recognition for human–vehicle interaction," in *5th Int. Conf. Control, Automation, Robotics and Vision*, Singapore, 1998, pp. 151–155.

[13] S. Waldherr, R. Romero, and S. Thrun, "A gesture-based interface for human–robot interaction," *Auton. Robots*, vol. 9, no. 2, pp. 151–173, Sep. 2000.

[14] M. H. Yang, N. Ahuja, and M. Tabb, "Extraction of 2-D motion trajectories and its application to hand gesture recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1062–1074, Aug. 2002.

[15] X. Yin and M. Xie, "Finger identification in hand gesture based human–robot interaction," *J. Robot. and Auton. Syst.*, vol. 34, no. 4, pp. 235–250, 2001.

[16] J. Triesch and C. von der Malsburg, "A system for person-independent hand posture recognition against complex backgrounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 12, pp. 1449–1453, Dec. 2001.

[17] S. S. Ghidary, M. Hattori, S. Tadokoro, and T. Takamori, "Multi-modal human robot interaction for map generation," in *Proc. IEEE Intelligent Robots and Systems (IROS)*, Maui, HI, 2001, pp. 2246–2251.

[18] S. Sato and S. Sakane, "A human–robot interface using an interactive hand pointer that projects a mark in the real work space," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, 2000, pp. 589–595.

[19] G. Heidemann and H. Ritter, "Combining gestural and contact information for visual guidance of multi-finger grasps," in *Proc. European Symp. Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2002, pp. 301–306.

[20] J. C. Bezdek, "Cluster validity with fuzzy sets," *Cybernetics*, vol. 3, no. 3, pp. 58–73, 1973.

[21] J. Wachs, U. Kartoun, H. Stern, and Y. Edan, "Real-time hand gesture telerobotic system using fuzzy c-means clustering," in *Proc. 5th Biannual World Automation Congr.*, Orlando, FL, 2002, vol. 13, pp. 403–409.

[22] N. R. Pal and J. C. Bezdek, "Complexity reduction for 'large image' processing," *IEEE Trans. Syst., Man, Cybern. B*, vol. 32, no. 5, pp. 598–611, Oct. 2002.

[23] J. F. Yang, S. S. Hao, and P. C. Chung, "Color image segmentation using fuzzy C-means and eigensapce projections," *Signal Process.*, vol. 82, no. 3, pp. 461–472, Mar. 2002.

[24] H. Stern and B. Efros, "Adaptive color space switching for tracking under varying illumination," *J. Image Vis. Comput.*, vol. 23, no. 3, pp. 353–364, 2005.

[25] N. B. Karayiannis and P. I. Pai, "Fuzzy vector quantization algorithms and their application in image compression," *IEEE Trans. Image Process.*, vol. 4, no. 9, pp. 1193–1201, Sep. 1995.

[26] D. S. Modha and W. S. Spangler, "Feature weighting in k-means clustering," *Mach. Learn.*, vol. 52, no. 3, pp. 217–237, Sep. 2003.

[27] R. K. Ahuja, O. Ergun, J. B. Orlin, and A. P. Punnen, "A survey of very large scale neighborhood search techniques," *Discrete Appl. Math.*, vol. 123, no. 1–3, pp. 75–102, Nov. 2002.

[28] J. Wachs, H. Stern, and Y. Edan, "Parameter search for an image processing—Fuzzy c-means hand gesture recognition system," in *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, 2003, vol. 3, pp. 341–344.

[29] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 370–379, Aug. 1995.

[30] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

[31] J. Wachs, U. Karoun, H. Stern, and Y. Edan. (2002). Tele-Gest Project. [Online]. Available: http://www.ie.bgu.ac.il/juantesa/tele_gest/ index.htm

[32] E. Micheli-Tzanakou, *Supervised and Unsupervised Pattern Recognition: Feature Extraction and Computational Intelligence.* Boca Raton, FL: CRC Press, 2000.

[33] Michigan State University. (2002). American Sign Language Browser, East Lansing. [Online]. Available: http://commtechlab.msu.edu/sites/ aslweb/browser.htm

[34] *Sebastien Marcel's Gesture Database Web Page.* http://www.idiap.ch/ ~marcel/Databases/main.html

[35] F. C. M. Kjeldsen, "Visual interpretation of hand gestures as a practical interface modality," Ph.D. dissertation, Dept. Comput. Sci., Columbia Univ., New York, 1997.

[36] J. Triesch and C. von der Malsburg, "Robust classification of hand postures against complex backgrounds," in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition*, Killington, VT, 1996, pp. 170–175.

Juan P. Wachs (M'03) was born in Argentina in 1971. He received the B.Ed.Tech. degree in electrical education from the ORT Academic College in Jerusalem, Israel, in 1995. He received the M.Sc. degree in information systems, at the Department of Industrial Engineering and Management, Ben-Gurion University, in 2003, and is currently working toward the Ph.D. degree in intelligent systems at Ben-Gurion University, Be'er-Sheeva, Israel.

Since 2000, he has been a critical member of a team of researchers working on virtual reality tele-robotic operations through the use of a hand-gesture computer interface at the Ben-Gurion University of the Negev, Israel. His current research interests include machine vision, telerobotics, and virtual reality.

Mr. Wachs is a Member of the Operation Research Society of Israel (ORSIS).

Helman Stern (M'98) received the B.Sc. degree in electrical engineering from Drexel University, Philadelphia, PA, in 1957, the M.Sc. degree in engineering administration from George Washington University, Washington DC, in 1961, and the Ph.D. degree in operations research from the University of California, Berkeley, in 1969.

He has taught at Rensselear Polytechnic Institute, St. Troy, NY, the University of California, Berkeley, in 1961 State University of New York at Albany, and San Francisco State University, San Francisco, CA. He has been with Ben-Gurion University of the Negev since 1974, where he heads the Intelligent Systems Program and teaches courses in intelligent systems and machine vision. He has publications in such journals as *Pattern Recognition Letters*, *Journal of Image and Graphics*, *Journal of Image and Vision Computing*, *Networks*, *Computers & OR*, *Management Science*, *Operations Research*, *The Journal of Optimization Theory and Applications*, *Imaging Science*, *Human Factors*, and *Journal of Image and Vision Computing*. His current research interests are in image processing, machine and computer vision, intelligent multimedia systems, video multiplexing, hand-gesture recognition, face tracking, teleoperations, and scheduling.

Prof. Stern is a member of IEEE Computer Society, SigMultimedia, and INFORMS.

Yael Edan (M'88) received the B.Sc. degree in computer engineering and the M.Sc. degree in agricultural engineering from Technion–Israel Institute of Technology, Technion City, Haifa, Israel, in 1984 and 1988, respectively, and the Ph.D. degree in engineering from Purdue University, West Lafayette, IN, in 1990.

She has performed research in robotics, sensors, system architectures, simulation, and decision-making systems. In addition, she has made major contributions in the introduction and application of intelligent automation and robotic systems to the field of agriculture with several patents. She is currently the Chair of the Department of Industrial Engineering and Management at Ben-Gurion University of the Negev, Israel, and was the Chair of the Paul Ivanier Center for Robotics and Production Management at Ben-Gurion University, Israel. She has been the Chair of the Flexible Automation and Robotics/Mechatronics and BioRobotics committees, ASAE. She is active on several editorial boards (*International Journal of Industrial Engineering—Applications and Practice*; *IIE Transactions*) and has been an Associate Editor of *Transactions of the ASAE*.

Prof. Edan is a member of the ASAE, Society for Engineering in Agricultural, Food, and Biological Systems.