A Robust Competitive Clustering Algorithm with Applications in Computer Vision

Hichem Frigui

Department of Electrical Engineering
University of Memphis
Memphis, TN 38152
hfrigui@memphis.edu

Raghu Krishnapuram

Department of Mathematical and Computer Sciences
Colorado School of Mines
Golden, CO 80401
rkrishna@mines.edu

December 3, 1998

Acknowledgment: This work was partially supported by a grant from the Office of Naval Research (N00014-96-1-0439).

Abstract

This paper addresses three major issues associated with conventional partitional clustering, namely, sensitivity to initialization, difficulty in determining the number of clusters, and sensitivity to noise and outliers. The proposed Robust Competitive Agglomeration (RCA) algorithm starts with a large number of clusters to reduce the sensitivity to initialization, and determines the actual number of clusters by a process of competitive agglomeration. Noise immunity is achieved by incorporating concepts from robust statistics into the algorithm. RCA assigns two different sets of weights for each data point: the first set of constrained weights represents degrees of sharing, and is used to create a competitive environment and to generate a fuzzy partition of the data set. The second set corresponds to robust weights, and is used to obtain robust estimates of the cluster prototypes. By choosing an appropriate distance measure in the objective function, RCA can be used to find an *unknown* number of clusters of various shapes in noisy data sets, as well as to fit an *unknown* number of parametric models *simultaneously*. Several examples, such as clustering/mixture decomposition, line/plane fitting, segmentation of range images, and estimation of motion parameters of multiple objects, are shown.

Index Terms: Robust clustering, fuzzy clustering, competitive clustering, robust statistics, optimal number of clusters, linear regression, range image segmentation, motion estimation.

1 Introduction

Traditional clustering algorithms can be classified into two main categories [1]: hierarchical and partitional. In hierarchical clustering, the number of clusters need not be specified *a priori*, and problems due to initialization and local minima do not arise. However, since hierarchical methods consider only local neighbors in each step, they cannot incorporate *a priori* knowledge about the global shape or size of clusters. As a result, they cannot always separate overlapping clusters. Moreover, hierarchical clustering is static, and points committed to a given cluster in the early stages cannot move to a different cluster.

Prototype-based partitional clustering algorithms can be divided into two classes: crisp (or hard) clustering where each data point belongs to only one cluster, and fuzzy clustering where every data point belongs to every cluster to a certain degree. Fuzzy clustering algorithms can deal with overlapping cluster boundaries. Partitional algorithms are dynamic, and points can move from one cluster to another. They can incorporate knowledge about the shape or size of clusters by using appropriate prototypes and distance measures. These algorithms have been extended to detect lines, planes, circles, ellipses, curves and surfaces [2, 3, 4, 5]. Most partitional approaches use the alternating optimization technique, whose iterative nature makes them sensitive to initialization and susceptible to local minima. Two other major drawbacks of the partitional approach are the difficulty in determining the number of clusters, and the sensitivity to noise and outliers.

In this paper, we describe a new approach called Robust Competitive Agglomeration (RCA), which combines the advantages of hierarchical and partitional clustering techniques [6]. RCA determines the "optimum" number of clusters via a process of competitive agglomeration [7], while knowledge about the global shape of clusters is incorporated via the use of prototypes. To overcome the sensitivity to outliers, we incorporate concepts from robust statistics. Overlapping clusters are handled by the use of fuzzy memberships. The algorithm starts by partitioning the data set into a large number of small clusters which reduces its sensitivity to initialization. As the algorithm progresses, adjacent clusters com-

pete for points, and clusters that lose the competition gradually vanish. However, unlike in traditional hierarchical clustering, points can move from one cluster to another. RCA uses two different sets of weights (or memberships) for each data point: the first one is a set of probabilistically constrained memberships that represent degrees of sharing among the clusters. The constraint generates a good partition and introduces competition among clusters. The second set of memberships is unconstrained or possibilistic [8, 9, 10], and represents degrees of "typicality" of the points with respect to the clusters. These memberships are used to obtain robust estimates of the cluster prototypes.

The organization of the rest of the paper is as follows. In section 2, we briefly review other related approaches. In section 3, we present the RCA algorithm. In section 4, we illustrate the power and flexibility of RCA to incorporate various distance measures. In section 5, we describe the application of RCA to segmentation of range images. In section 6, we formulate a multiple model general linear regression algorithm based on RCA and apply it to simultaneous estimation of motion parameters of multiple objects. Finally, section 7 contains the conclusions.

2 Related Work

Most prototype-based partitional clustering algorithms such as K-Means and Fuzzy C-Means (FCM) [2] assume that the number of clusters, C, is known. Moreoever, since they use a least squares criterion, they break down easily (i. e., the prototype parameter estimates can be arbitrarily wrong [11]) in the presence of noise. The goal of clustering is to identify clusters in the data set. This implicitly assumes that we have a definition for a valid cluster. Thus, the idea of break down [11] can be extended to the clustering domain via the use of validity [12]. When the number of clusters, C, is known, the ideal cluster breaks down only when the outliers form a valid cluster with a cardinality higher than the cardinality, N_{min}, of the smallest good cluster. This gives us the theoretical breakdown point of N_{min}/N, where N is the number of points in the data set. Recent solutions to robust clustering when

C is known can be divided into two categories. In the first category are algorithms that are derived by modifying the objective function of FCM [13, 14, 10]. These algorithms are still sensitive to initialization and other parameters [12]. The algorithms in second category incorporate techniques from robust statistics explicitly into their objective functions. A notable non-fuzzy clustering algorithms in this category is the K-Medoids algorithm [15]. Bobrowski and Bezdek [16] proposed an L_1-norm-based fuzzy clustering algorithm which also falls into this category. However, there is no mention of robustness in this paper. A variation of this algorithm that is motivated by robustness can be found in [17]. Another early fuzzy clustering algorithm (on which RCA is based) is the Robust C-Prototypes (RCP) algorithm [18], which uses the M-estimator [19]. The Fuzzy Trimmed C Prototypes (FTCP) algorithm [20] uses the least trimmed squares estimator [21], the Robust Fuzzy C Means (RFCM) algorithm [22] again uses the M-estimator in a different way, and the Fuzzy C Least Median of Squares (FCLMS) algorithm [23] uses the least median of squares estimator [21]. FTCP and FCLMS can achieve the theoretical breakdown point of N_{min}/N with a trivial modification to their objective functions. However, in theory, they both require an exhaustive search. To reduce the computational complexty, a heuristic search is used in [20]and a genetic search is used in [23].

When C is unknown, one way to state the clustering problem is: find all the valid clusters in the data set (see [12] for a more precise definition). In this case, the ideal algorithm will not break down because it will identify all the "good" clusters correctly (say by exhaustive search), in addtion to some spurious ones. An alternative way to state the problem is: identify only all the valid clusters formed by the *good* data. In this case, the ideal algorithm will break down when the outliers form a valid cluster, giving us the breakdown point of N_{minval}/N, where N_{minval} is the minimum number of points required to form a valid cluster. Note that a given clustering algorithm may not achieve these theoretical breakdown points.

The traditional approach to determining C is to evaluate a certain global validity measure of the C-partition for a range of C values, and then pick the value of C that optimizes

the validity measure [25, 1, 26, 27]. An alternative is to perform progressive clustering [28, 27, 5], where clustering is initially performed with an overspecified number of clusters. After convergence, spurious clusters are eliminated, compatible clusters are merged, and "good" clusters are identified. Another variation of progressive clustering extracts one cluster at a time [29, 30]. These approaches are either computationally expensive, or rely on validity measures (global or individual) which can be difficult to devise. Robust approaches to clustering when C is unknown treat the data as a mixture of components, and use a robust estimator to estimate the parameters of each component. The Generalized MVE (GMVE) [29] which is based on the Minimum Volume Ellipsoid estimator [21], the Model Fitting (MF) algorithm [31], and the Possibilistic Gaussian Mixture Decomposition (PGMD) algorithm [30] are some examples. In the above approaches, the data set is classified into a set of "inliers", i.e., points belonging to a cluster, and a set of "outliers". Since the set of outliers includes points from other clusters, the proportion of outliers can be very high. Therefore, even the use of a robust estimaor with the theoretical-best breakdown point of 50% is not sufficient to make these algorithms highly robust. To overcome this problem, these algorithms consider the "validity" of the cluster formed by the inliers, and try to extract every valid cluster in the data set. In order to guarantee a good solution, the GMVE and PGMD use many random initializations. Cooperative Robust Estimation (CRE) [32] and MINPRAN [33] are two other robust model-fitting approaches that fall into this category. The CRE algorithm attempts to overcome the low breakdown point of M-estimators by initializing a large number of hypotheses and then selecting a subset of the initial hypotheses based on the Minimum Description Length (MDL) criterion. The CRE technique assumes that the scale (θ in [32]) is known. MINPRAN assumes that the outliers are randomly distributed within the dynamic range of the sensor, and the noise (outlier) distribution is known. Because of these assumptions, CRE and MINPRAN do not easily extend to the clustering domain. If the data is expected to have multiple curves, MINPRAN seeks one curve/surface at a time. In [12] the relation between the above progressive approaches and other robust clustering

algorithms are explored.

When the clusters overlap, the idea of extracting them in a serial fashion will not work. Removing one cluster may partially destroy the structure of other clusters, or we might get "bridging fits" [33]. Fig. 2(a) shows one such noisy data set with two crossing clusters. The algorithm we propose is designed to overcome this drawback. Moreover, all the current algorithms use hard finite rejection [34], i.e., points within an inlier bound are given a weight of 1, and points outside the bound are given a weight of zero. This means that these algorithms do not handle the "region of doubt" [21] very well. To overcome this problem, we use smooth [34, 21] or fuzzy rejection, where the weight function drops to zero gradually.

3 The Robust Competitive Agglomeration (RCA) algorithm

3.1 Algorithm Development

Let $\mathcal{X} = \{\mathbf{x}_j \mid j = 1, \ldots, N\}$ be a set of N vectors in an n-dimensional feature space with coordinate axis labels $(x_1, \cdots, x_n)$. Let $\mathbf{B} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_C)$ represent a C-tuple of prototypes each of which characterizes one of the C clusters. Each $\boldsymbol{\beta}_i$ consists of a set of parameters. The Fuzzy C-Means algorithm [2] minimizes:

$$J_m(\mathbf{B}, \mathbf{U}; \mathcal{X}) = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^m d_{ij}^2 \tag{1}$$

subject to

$$\sum_{i=1}^{C} u_{ij} = 1, \qquad \text{for } 1 \leq j \leq N. \tag{2}$$

In (1), d_{ij}^2 represents the distance of feature vector $\mathbf{x}_j$ from prototype $\boldsymbol{\beta}_i$, u_{ij} represents the degree to which $\mathbf{x}_j$ belongs to cluster i, $\mathbf{U} = [u_{ij}]$ is a $C \times N$ matrix called the constrained fuzzy C-partition matrix, and $m \in [0, \infty)$ is known as the fuzzifier. J_m, which is essentially the sum of (fuzzy) intra-cluster distances, has a monotonic tendency, and has the minimum value of zero when $C{=}N$. Therefore, it is not useful for the automatic determination of C. To overcome this drawback, we add a second regularization term to prevent overfitting the

data set with too many prototypes. The resulting objective function J_A is:

$$J_A(\mathbf{B}, \mathbf{U}; \mathcal{X}) = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 d_{ij}^2 \;\; - \;\; \alpha \sum_{i=1}^{C} \Big[\sum_{j=1}^{N} u_{ij} \Big]^2, \tag{3}$$

which is minimized subject to the constraint in (2). In (3), the second term is the negative of the sum of the squares of the cardinalities of the clusters, and is minimized when the cardinality of one of the clusters is N and the rest of the clusters are empty. With a proper choice of α, we can balance the two terms to find a solution for C. J_A is still not robust, since the first term is a Least Squares objective function. Therefore, we robustify J_A to yield the objective function for the proposed RCA algorithm as follows:

$$J_R(\mathbf{B}, \mathbf{U}; \mathcal{X}) = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 \rho_i(d_{ij}^2) \;\; - \;\; \alpha \sum_{i=1}^{C} \Big[\sum_{j=1}^{N} w_{ij} u_{ij} \Big]^2 \tag{4}$$

In (4), $\rho_i()$ is a robust loss function associated with cluster i, and $w_{ij} = w_i(d_{ij}^2) = \partial \rho_i(d_{ij}^2)/\partial d_{ij}^2$ represents the "typicality" of point $\mathbf{x}_j$ with respect to cluster i. The function $\rho_i()$ corresponds to the loss function used in M-estimators of robust statistics and $w_i()$ represents the weight function of an equivalent W-estimator (see, [11], for example). This particular choice for robustification is motivated by the need to keep the computational complexity low. The loss function reduces the effect of outliers on the first term, and the weight function discounts outliers while computing the cardinalities. By selecting d_{ij} and the α prudently, J_R can be used to find compact clusters of various types while partitioning the data set into a minimal number of clusters.

To minimize J_R with respect to the prototype parameters, we fix $\mathbf{U}$ and set the derivative of J_R with respect to $\boldsymbol{\beta}_i$ to zero, i.e.,

$$\sum_{j=1}^{N} (u_{ij})^m w_{ij} \frac{\partial d_{ij}^2}{\partial \boldsymbol{\beta}_i} = 0. \tag{5}$$

Further simplification of (5) depends on $\rho_i()$ and d_{ij}. Since the distance measure is application dependent, we will return to this issue in Section 4. To minimize (4) with respect to $\mathbf{U}$ subject to (2), we apply Lagrange multipliers and obtain

$$J(\mathbf{B}, \mathbf{U}; \mathcal{X}) = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 \rho_i(d_{ij}^2) \;\; - \;\; \alpha \sum_{i=1}^{C} \Big[\sum_{j=1}^{N} w_{ij} u_{ij} \Big]^2 \;\; - \;\; \sum_{j=1}^{N} \lambda_j \Big(\sum_{i=1}^{C} u_{ij} - 1 \Big). \tag{6}$$

We then fix $\mathbf{B}$ and solve

$$\frac{\partial J}{\partial u_{st}} = 2u_{st}\rho_s(d_{st}^2) \; - \; 2\alpha \sum_{j=1}^{N} w_{sj}u_{sj} - \lambda_t = 0 \quad \text{for } 1 \leq s \leq C, \text{and } 1 \leq t \leq N. \tag{7}$$

Equations (7) and (2) represent a set of of $N\times C+N$ linear equations with $N\times C+N$ unknowns (u_{st}, and λ_t). A computationally simple solution can be obtained by computing the term $\sum_{j=1}^{N} w_{sj}u_{sj}$ in (7) using the memberships from the previous iteration. This yields:

$$u_{st} = \frac{2\alpha \times (\sum_{j=1}^{N} w_{sj}u_{sj}) + \lambda_t}{2\rho_s(d_{st}^2)}. \tag{8}$$

Solving for λ_t using (8) and (2), and substituting in (8), we obtain the following update equation for the membership u_{st} of feature point $\mathbf{x}_t$ in cluster $\boldsymbol{\beta}_s$:

$$u_{st} = \frac{1/\rho_s(d_{st}^2)}{\sum_{k=1}^{C} 1/\rho_k(d_{kt}^2)} + \frac{\alpha}{\rho_s(d_{st}^2)}(N_s - \overline{N}_t) = u_{st}^{\text{RR}} + u_{st}^{\text{Bias}}, \tag{9}$$

where u_{st}^{RR} is the degree to which cluster s shares $\mathbf{x}_t$ (computed using robust distances), and u_{st}^{Bias} is a signed bias term which depends on the difference between the robust cardinality, $N_s = \sum_{j=1}^{N} w_{sj}u_{sj}$, of the cluster of interest and the weighted average of cardinalities

$$\overline{N}_t = \sum_{k=1}^{C} \frac{1}{\rho_k(d_{kt}^2)} N_k \Big/ \sum_{k=1}^{C} 1/\rho_k(d_{kt}^2).$$

The bias term, u_{st}^{Bias}, is positive(negative) for clusters with cardinality higher(lower) than average, and hence the membership of $\mathbf{x}_t$ in such clusters will appreciate(depreciate). When a feature point $\mathbf{x}_j$ is close to only one cluster (say cluster i), and far from other clusters, we have $N_i \approx \overline{N}_j$, or $u_{ij}^{\text{Bias}} \approx 0$, implying no competition. On the other hand, if a point is roughly equidistant from several clusters, these clusters will compete for this point based on cardinality. When the cardinality of a cluster drops below a threshold, we discard the cluster, and update the number of clusters.

It is possible for u_{ij} to become negative if N_i is very small and point $\mathbf{x}_j$ is close to other dense clusters. In this case, it is safe to set u_{ij} to zero. It is also possible for u_{ij} to become larger than 1 if N_i is very large and feature point $\mathbf{x}_j$ is close to other low cardinality clusters. In this case it is clipped to 1. This practice is customary in optimization theory.

The process of agglomeration, controlled by α, should be slow in the beginning to encourage the formation of small clusters. Then it should be increased gradually to promote agglomeration. After a few iterations, when the number of clusters becomes close to the "optimum", the value of α should again decay slowly to allow the algorithm to converge. Therefore an appropriate choice of α in iteration k is.

$$\alpha(k) = \eta(k) \frac{\sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij}^{(k-1)})^2 \rho_i (d_{ij}^2)^{(k-1)}}{\sum_{i=1}^{C} \left[\sum_{j=1}^{N} w_{ij}^{(k-1)} u_{ij}^{(k-1)} \right]^2}. \tag{10}$$

In (10), α and η are functions of the iteration number k, and the superscript $(k-1)$ is used on u_{ij}, d_{ij}^2, and w_{ij} to denote their values in iteration $k-1$. A good choice for η is

$$\eta(k) = \begin{cases} \eta_0 \ e^{-|k_0 - k|/\tau} & \text{if } k > 0 \\ 0 & \text{if } k = 0 \end{cases} \tag{11}$$

where η_0 is the initial value, τ is the time constant, and k_0 is the iteration number at which η starts to decrease. In all examples presented in this paper (except in section 5 where these parameters were fine-tuned for best performance), we choose $\eta_0 = 1$, $k_0 = 5$, and $\tau = 10$. With proper initialization, these values are reasonable regardless of the application. Initialization issues are discussed in section 7.

3.2 Choice of the weight function

In curve/surface fitting or linear regression, it is reasonable to assume that the residuals have a symmetric distribution about zero. Therefore, we choose Tukey's biweight function [11] given by

$$\rho_i((r_{ij}^*)^2) = \begin{cases} \frac{1}{3}[1 - (1 - (r_{ij}^*)^2)^3] & \text{if } |r_{ij}^*| \le 1, \\ \frac{1}{3} & \text{if } |r_{ij}^*| > 1 \end{cases} \tag{12}$$

$$w_i((r_{ij}^*)^2) = \begin{cases} [1 - (1 - (r_{ij}^*)^2)^2] & \text{if } |r_{ij}^*| \le 1, \\ 0 & \text{if } |r_{ij}^*| > 1 \end{cases} \tag{13}$$

where r_{ij}^* stands for the normalized residual defined as:

$$r_{ij}^* = \frac{r_{ij} - Med_i}{c \times MAD_i}. \tag{14}$$

In (12)-(14), r_{ij} is the residual of the j^{th} point with respect to the i^{th} cluster, Med_i is the median of the residuals of the i^{th} cluster, and MAD is the median of absolute deviations [11]

of the i^{th} cluster. In other words, in each iteration, the data set $\mathcal{X}$ is crisply partitioned into C components $\mathcal{X}_i$, for $i = 1, \cdots, C$, and Med_i and MAD_i are estimated for each cluster.

When distances (rather than residuals) are used, the symmetric distribution assumption does not hold. We suggest a monotonically non-increasing weight function $w_i(d^2) : \mathcal{R}^+ \rightarrow [0, 1]$ such that $w_i(d^2) = 0$ for $d^2 > T_i + cS_i$, where c is a constant, and T_i and S_i are given by

$$T_i = \underset{i}{Med}\ (d_{ij}^2) \quad \text{and} \quad S_i = \underset{i}{MAD}\ (d_{ij}^2) \qquad \text{for } i = 1, \cdots, C. \tag{15}$$

Choosing $w_i(0) = 1$, $w_i(T_i) = 0.5$, and $w_i\prime(0) = 0$, results in the following weight function:

$$w_i(d^2) = \begin{cases} 1 - \frac{d^4}{2T_i^2} & \text{if } d^2 \in [0, T_i], \\ \frac{\left[d^2 - (T_i + cS_i)\right]^2}{2c^2 S_i^2} & \text{if } d^2 \in (T_i, T_i + cS_i], \\ 0 & \text{if } d^2 > T_i + cS_i. \end{cases} \tag{16}$$

The corresponding loss function can be shown to be

$$\rho_i(d^2) = \begin{cases} d^2 - \frac{d^6}{6T_i^2} & \text{if } d^2 \in [0, T_i], \\ \frac{\left[d^2 - (T_i + cS_i)\right]^3}{6c^2 S_i^2} + \frac{5T_i + cS_i}{6} & \text{if } d^2 \in (T_i, T_i + cS_i], \\ \frac{5T_i + cS_i}{6} + K_i & \text{if } d^2 > T_i + cS_i. \end{cases} \tag{17}$$

In (17) K_i is an integration constant used to make all $\rho_i()$ reach the same maximum value.

$$K_i = \max_{1 \le j \le C} \left\{ \frac{5T_j + cS_j}{6} \right\} - \frac{5T_i + cS_i}{6} \qquad \text{for } i = 1, \cdots, C.$$

This choice ensures that all noise points will have the same membership value in all clusters. Fig 1 shows the plot of the weight function and the corresponding loss function.

In (14), (16), and (17), c is a tuning constant [11] which is normally chosen to be between 4 and 12. When c is large, many outliers will have small nonzero weights, thus affecting the parameter estimates. On the other hand, if c is small, only a subset of the data points will be visible to the estimation process, making convergence to a local minimum more likely. As a compromise, we start the estimation process with a large value of c, and then decrease it gradually as function of the iteration number (k), i.e.,

$$c_k = max(c_{min}, c_{k-1} - \Delta c) \tag{18}$$

with c_0=12, c_{min}=4, and Δc=1.

The RCA algorithm is summarized below.

Fix the maximum number of clusters $C = C_{max}$*;*
Initialize the prototype parameters, and set $k = 0$ *;*
Set w_{ij}*=1* $\forall\ i, j$*;*
Repeat
    Compute d_{ij}^2 *for* $1 \leq i \leq C$ *and* $1 \leq j \leq N$*;*
    Estimate T_i *and* S_i *by using (15);*
    Update the weights w_{ij} *by using (13) or (16);*
    Update $\alpha(k)$ *by using (10);*
    Update the partition matrix $\mathbf{U}^{(k)}$ *by using (9);*
    Compute the robust cardinality N_i*;*
    If ($N_i < \epsilon_1$) *discard cluster* $\boldsymbol{\beta}_i$*;*
    Update the number of clusters C*;*
    $k = k + 1$ *;*
    Update the tuning factor c *by using (18);*
    Update the prototype parameters;
Until (*prototype parameters stabilize*)*;*

4 Examples of Distance Measures

As mentioned in section 3.1, RCA can be used with a variety of distance measures depending on the nature of the application. In this section, we discuss distance measures suitable for ellipsoidal clusters and hyperplanes.

4.1 Detection of Ellipsoidal Clusters

To detect ellipsoidal clusters in a data set, we use the following distance measure [35, 36].

$$d_{Cij}^2 = |\mathbf{C}_i|^{1/n}(\mathbf{x}_j - \mathbf{c}_i)^T \mathbf{C}_i^{-1}(\mathbf{x}_j - \mathbf{c}_i). \tag{19}$$

In (19), $\mathbf{c}_i$ is the center of cluster $\boldsymbol{\beta}_i$, and $\mathbf{C}_i$ is its covariance matrix. (See [37] for an interpretation of d_{Cij}^2.) Using (5), it can be shown that the update equations for the centers $\mathbf{c}_i$ and the covariance matrices $\mathbf{C}_i$ are

$$\mathbf{c}_i = \frac{\sum_{j=1}^{N}(u_{ij})^2 w_{ij}\mathbf{x}_j}{\sum_{j=1}^{N}(u_{ij})^2 w_{ij}}, \tag{20}$$

$$\mathbf{C}_i = \frac{\sum_{j=1}^{N} (u_{ij})^2 w_{ij} (\mathbf{x}_j - \mathbf{c}_i)(\mathbf{x}_j - \mathbf{c}_i)^T}{\sum_{j=1}^{N} (u_{ij})^2 w_{ij}}. \tag{21}$$

If we assume $\mathbf{C}_i = \sigma^2 \mathbf{I}_n$, then (19) reduces to the Euclidean distance. This simplified version can be used when the clusters are expected to be spherical.

Fig. 3 illustrates RCA using d^2_{Cij}. Fig. 3(a) shows a synthetic Gaussian mixture consisting of 4 clusters of various sizes and orientations. Uniformly distributed noise constituting 40% of the total points was added to the data set. Fig. 3(b) shows the initial 20 prototypes superimposed on the data set, where "+" signs indicate the cluster centers, and the ellipses enclose points with a Mahalanobis distance less than 9. These prototypes were obtained by running the G-K algorithm [36] for 5 iterations. After 2 iterations of RCA, 9 empty clusters are discarded (see Fig. 3(c)). The number of clusters is reduced to 6 after 3 iterations, and to 4 after 4 iterations. The final result after a total of 10 iterations is shown in Fig. 3(d).

To illustrate the ability of RCA to handle non-uniform noise, Fig. 4 shows the result of RCA on a data set containing Gaussian clusters with roughly 25% noise. To illustrate the ability of the RCA algorithm to detect overlapping clusters, in Fig. 2(b) we show the result of RCA on the data set in Fig. 2(a). The algorithm converged in 10 iterations.

4.2 Detection of Linear Clusters

To detect clusters that resemble lines or planes, we use a generalization of the distance measure proposed in [3, 2]. This distance is given by

$$d^2_{Lij} = \sum_{k=1}^{n} \nu_{ik} ((\mathbf{x}_j - \mathbf{c}_i) \bullet \mathbf{e}_{ik})^2, \tag{22}$$

where $\mathbf{e}_{ik}$ is the k^{th} unit eigenvector of the covariance matrix $\mathbf{C}_i$. The eigenvectors are assumed to be arranged in ascending order of the corresponding eigenvalues. The value of ν_{ik} in (22) is chosen dynamically in every iteration to be $\nu_{ik} = \lambda_{in}/\lambda_{ik}$, where ν_{ik} is the k^{th} eigenvalue of $\mathbf{C}_i$. It can be shown that for the distance measure in (22), the update equations for $\mathbf{c}_i$ and $\mathbf{C}_i$ are given by (20) and (21) respectively.

Fig. 5(a) shows an image consisting of 10 line segments in a noisy background. Fig. 5(b)

shows the 20 initial prototypes obtained by running the AFC algorithm [3] for 5 iterations. After 2 iterations of RCA, the number of clusters drops to 15 as shown in Fig. 5(c). After 9 iterations, the number of clusters reduces to the "optimal" number and the algorithm converges after a total of 12 iterations. The final result is shown in Fig. 5(d).

5 Application to Range Image Segmentation

5.1 Planar Range Image Segmentation

Since planar surface patches can be modeled by flat ellipsoids, the distance measure d^2_{Cij} in (19) can also be used to find the optimal number of planar patches. To avoid missing tiny surfaces, we start by dividing the image into non-overlapping windows of sizes $W_s \times W_s$. Then, we apply RCA in each window with $C = C_{max}$ to estimate the optimal number of planar patches within the window. Finally, we pool the resulting (say M) prototypes to initialize the RCA algorithm with $C{=}M$. Because of the nature of d^2_{Cij}, planar surfaces with non-convex shapes may be approximated by several planar patches, or several spatially disconnected planar patches may be approximated by a single cluster. Therefore, after RCA converges, we merge compatible clusters [27] that are adjacent. We then perform connected component labeling on each cluster, and assign different labels to disjoint regions.

The above RCA-based algorithm was tested on two standard data sets, ABW data set and perceptron data set, that were created for bench-marking range image segmentation algorithms [38]. Each set contains 40 images of size 512×512, and has been randomly divided into a 10-image training set and a 30-image testing set. We use the performance measures developed by Hoover et al. [38] to evaluate the performance of RCA. These measures rely on comparing the Machine Segmented (MS) image and the Ground Truth (GT) image, and classify the regions into one of the 5 categories: correct detection, over-segmentation, under-segmentation, missed, and noise. The accuracy of the segmentation is quatified by computing the average and standard deviation of the differences between the angles made by all pairs of adjacent regions that are instances of correct detection in the MS and GT images. The

above data sets and performance measures have been used in [38] to compare the University of South Florida (USF), University of Edinburgh (UE), Washington State University (WSU), and University of Bern (UB) segmentation algorithms. Here, we will reproduce the same set of experiments and include the RCA algorithm in the comparison.

In the training phase, we fine-tuned the parameters of RCA as follows: window size used in the initialization $W_s = 128$; initial number of prototypes in each window $C_{max} = 15$; $(\eta_0, \tau) = (2, 20)$ (see (10)). These parameters are optimal for both ABW and Perceptron data sets. Since the Perceptron data is more noisy, we use $c_{min} = 4$, and for the ABW data, $c_{min} = 8$. Also, to reduce computations, all images were subsampled in the x and y directions by a factor of 3. These parameters are all then fixed in the testing phase.

Fig 6(a) shows the intensity image of one of the ABW test images. The segmented range image is shown in Fig 6(b). The shaded gray regions correspond to background points that are ignored during segmentation. Fig. 7 shows an example from the Perceptron data set. As in [38], we compute the performance metrics of the five segmentation algorithms while varying the compare tool tolerance from 51% to 95%. Due to space limitation, we only show plots of the correct detection measure (Fig 8). The performance measures using an 80% compare tolerance for all five segmenters are listed in Table 1 for the ABW data and Table 2 for the Perceptron data. RCA compares very well with the best segmenters.

Among the 5 planar surface segmenters in the comparison, UE, WSU, and RCA have the capability to segment curved surfaces. RCA has the additional advantage that it can handle irregularly spaced sparse data as well (e.g. range data computed from stereo methods).

5.2 Quadric Range Image Segmentation

Let the i^{th} prototype $\boldsymbol{\beta}_i$, represented by the parameter vector $\mathbf{p}_i$, define the equation of a quadric surface as $\mathbf{p}_i^T \mathbf{q} = 0$, where $\mathbf{p}_i^T = [p_{i1}, p_{i2}, \ldots, p_{i10}]$, $\mathbf{q}^T = [x^2, y^2, z^2, xy, xz, yz, x, y, z, 1]$, and $\mathbf{x} = (x, y, z)$ is a 3-D point. Since the exact distance from a point $\mathbf{x}_j$ to a quadric surface

$\boldsymbol{\beta}_i$ has no closed-form expression, we use the approximate distance [39, 40] given by

$$d_{Aij} = \frac{\mathbf{p}_i^T \mathbf{q}}{||\nabla \mathbf{p}_i^T \mathbf{q}||} = \frac{\mathbf{p}_i^T \mathbf{q}}{\sqrt{\mathbf{p}_i^T \mathbf{D}(\mathbf{q}_j)\mathbf{D}(\mathbf{q}_j)^T \mathbf{p}_i}}, \tag{23}$$

where $D(\mathbf{q}_j)$ is the Jacobian of $\mathbf{q}$ evaluated at $\mathbf{x}_j$. To avoid the all-zero trivial solution for $\mathbf{p}_i$, the following constraint may be chosen [39]

$$\mathbf{p}_i^T \Big[\sum_{j=1}^{N} (u_{ij})^2 w_{ij} [\mathbf{D}(\mathbf{q}_j)\mathbf{D}(\mathbf{q}_j)^T] \Big] \mathbf{p}_i = \sum_{j=1}^{N} (u_{ij})^2 w_{ij},$$

Starting from (5), it can be shown that the use of d_{Aij} leads to a solution of $\mathbf{p}_i$ based on the following generalized eigenvector problem: $\mathbf{F}_i \mathbf{p}_i = \lambda_i \mathbf{G}_i \mathbf{p}_i$, where $\mathbf{F}_i = \sum_{j=1}^{N} (u_{ij})^2 w_{ij} \mathbf{q}_j \mathbf{q}_j^T$, and $\mathbf{G}_i = \sum_{j=1}^{N} (u_{ij})^2 w_{ij} [\mathbf{D}(\mathbf{q}_j)\mathbf{D}(\mathbf{q}_j)^T]$.

To obtain a reliable initialization, we divide the image into small non-overlapping windows, and apply RCA in each window with C=1. Finally, we pool the resulting prototype parameters to initialize the RCA algorithm. Initially, there might be several initial prototypes corresponding to the same surface. However, due to competition, only one of these surfaces will survive.

The examples used in this section consist of some 240×240 real and some synthetic range images[1]. A sampling rate of 3 in the x and y directions was used to reduce computations. 30×30 windows were used to estimate the initial prototypes. Fig. 9(a) shows a synthetic range image of a plastic pipe. Fig. 9(b) shows the initial 36 surface patches. These patches were generated after assigning each point to the nearest prototype. Fig. 9(c) shows the final results, where each each surface is displayed with a different gray value, and the boundaries are shown in black. Fig. 10(a) shows a real range image of three plastic pipes of different sizes and orientations. The final results of the RCA algorithm consisting of the correctly identified surfaces are shown in Fig. 10(b).

To test the robustness of RCA, Gaussian noise (with σ=4) was added to the image in Fig. 9(a), and about 10% of the data points were randomly altered to become outliers. The

[1] These images were obtained from Michigan State University and Washington State University via anonymous ftp.

results are shown in Fig. 11, where noise points (i.e. points with zero weight (w_{ij}) in all clusters) are shown in black.

6 Estimation of Multiple Motion Groups and Segmentation

In this section, we show how RCA can be used to perform multiple model linear regression, and apply it to estimation of the motion parameters of multiple motion groups.

6.1 General Linear Regression

The General Linear Regression (GLR) [41] for solving a set of homogeneous equations for motion parameters can be written as: $\mathbf{X}\boldsymbol{\beta} = \mathbf{r}$, where $\mathbf{X}^T = (\mathbf{x}_1|\cdots|\mathbf{x}_N)$ is the design matrix with $\mathbf{x}_i = (1, x_{i1}, \cdots, x_{ip})^T$, $\boldsymbol{\beta} = [\beta_0, \beta_1, \cdots, \beta_p]^T$ is the parameter vector, and $\mathbf{r} = [r_0, r_1, \cdots, r_p]^T$ is the residual vector. Since the system is homogeneous, we can fix $\beta_0 = -1$, and reformulate the GLR model as: $-\mathbf{1} + \mathbf{X}^*\boldsymbol{\beta}^* = \mathbf{r}$, where $\mathbf{1}$ denotes a N-dimensional vector with every component equal to 1, $\mathbf{X} = [\mathbf{1}|\mathbf{X}^*]$, and $\boldsymbol{\beta}^T = [-1, \boldsymbol{\beta}^{*T}]$. GLR can be solved by the least squares minimization: $\min_{\boldsymbol{\beta}^*} \sum_i r_i^2 = \min_{\boldsymbol{\beta}^*} \|\mathbf{r}\|^2$, with the solution: $\boldsymbol{\beta}^* = (\mathbf{X}^{*T}\mathbf{X}^T)^{-1}\mathbf{X}^{*T}\mathbf{1}$. However, least squares is very sensitive to noise. An alternative is the weighted least squares: $\min_{\boldsymbol{\beta}^*} \sum_i w_i r_i^2$, with the solution: $\boldsymbol{\beta}^* = (\mathbf{X}^{*T}\mathbf{W}\mathbf{X}^T)^{-1}\mathbf{X}^{*T}\mathbf{W}\mathbf{1}$, where $\mathbf{W} = \text{diag}(w_1, \cdots, w_N)$.

If a data set contains multiple models, the GLR model must be applied repetetively to extract one model at a time. This approach is computationally expensive, requires models to be well separated, needs a high breakdown estimator (since while extracting the i^{th} model, all other models are considered as outliers), and is sensitive to initialization. To deal with these problems, we propose the Multiple-Model General Linear Regression (MMGLR) method, which allows the simultaneous estimation of an unknown number of models.

6.2 Multiple-Model General Linear Regression

Let the i^{th} model with the parameter vector $\boldsymbol{\beta}_i = [\beta_{i0}, \beta_{i1}, \cdots, \beta_{ip}]^T$, be represented by

$$\beta_{i0} + \beta_{i1}x_{j1} + \beta_{i2}x_{j2} + \cdots + \beta_{ip}x_{jp} = r_{ij}, \quad \text{for } 1 \leq j \leq N,$$

where r_{ij} is the residual corresponding to the j^{th} data vector in the i^{th} model. MMGRL minimizes (4) (where d_{ij}^2 is replaced by r_{ij}^2) subject to the constraint in (2). Solving (5) corresponding to this situation leads to $\frac{\partial}{\partial \boldsymbol{\beta}_i}\left\|\mathbf{U}_i\mathbf{W}_i^{1/2}(\mathbf{-1} + \mathbf{X}^*\boldsymbol{\beta}_i^*)\right\|^2 = 0$, where $\mathbf{U}_i = \text{diag}(u_{i1}, \cdots, u_{iN})$, and $\mathbf{W}_i^{1/2} = \text{diag}(w_{i1}^{1/2}, \cdots, w_{iN}^{1/2})$. The resulting update equation for the parameters is:

$$\boldsymbol{\beta}_i^* = (\mathbf{X}^{*T}\mathbf{U}_i^2\mathbf{W}_i\mathbf{X}^T)^{-1}\mathbf{X}^{*T}\mathbf{U}_i^2\mathbf{W}_i\mathbf{1}. \tag{24}$$

In linear regression, it is customary to use the studentized residuals $r_j^* = r_j/\sqrt{1-h_{jj}}$, where h_{jj} is the j^{th} diagonal element of the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. Huang et al. [41] showed that the corresponding hat matrix for GLR is $\mathbf{H}^* = \mathbf{X}^*(\mathbf{X}^{*T}\mathbf{X}^*)^{-1}\mathbf{X}^{*T}$. To extend this principle to the MMGLR, we compute C hat matrices (i. e., one per model), as

$$\mathbf{H}_i^* = \mathbf{W}_i\mathbf{U}_i^2\mathbf{X}^*\left(\mathbf{W}_i^2\mathbf{U}_i^4\mathbf{X}^{*T}\mathbf{X}^*\right)^{-1}\mathbf{W}_i\mathbf{U}_i^2\mathbf{X}^{*T}. \tag{25}$$

The residuals can be normalized as $r_{ij}^* = r_{ij}/\sqrt{1-h_{jj}^{*(i)}}$. However, this normalization introduces a bias towards noise points ($w_{ij} \approx 0$) or points belonging to other models ($u_{ij} \approx 0$). In this case $h_{jj}^{*(i)} \approx 0$, and hence no normalization takes place. Also, residuals will be inflated for points which are typical of the i^{th} model since they are divided by a factor smaller than one. Therefore, we modify the normalization process as follows:

$$r_{ij}^* = \begin{cases} \frac{r_{ij}}{\sqrt{1-h_{jj}^{*(i)}}} & \text{if } u_{ij}w_{ij} > \epsilon, \\ \frac{r_{ij}}{\sqrt{1-h_{max}^*}} & \text{otherwise} \end{cases} \tag{26}$$

where $h_{max}^* = \max_{i,j}\ h_{jj}^{*(i)}$. In other words, points which are known to be atypical of the i^{th}model, are forced to receive the maximum possible inflation factor.

MMGLR can be used to estimate the motion parameters of multiple objects in the same scene. The instantaneous velocity $\dot{\mathbf{p}}(t)$ of a point $\mathbf{p} = (x, y, z)^T$ located on the surface of

a translating object rotating with an instantaneous angular velocity $\boldsymbol{\omega}(t) = (\omega_1, \omega_2, \omega_3)^T$, is characterized by $\dot{\mathbf{p}}(t) = \boldsymbol{\omega}(t) \times \mathbf{p}(t) + \mathbf{k}(t)$, where $\mathbf{k}(t) = (k_1, k_2, k_3)^T$ is a vector involving translation. Let $(X(t), Y(t))$ be the 2-D prespective projection of $\mathbf{p}(t)$ onto the image plane at Z=1, and let $(u(t), v(t))$ denote its projective instantaneous velocity. Motion estimation consists of solving for $\boldsymbol{\omega}$ and $\mathbf{k}$ using a set of N observations $(X_j, Y_j)^T$ and their corresponding $(u_j, v_j)^T$ for $j = 1 \cdots N$. This can be done by solving $\mathbf{Ah} = 0$, where $\mathbf{A} = [a_1^T, a_2^T, \cdots, a_N^T]$, $\mathbf{a}_j = [1, X_j^2, Y_j^2, 2X_jY_j, 2X_j, 2Y_j, -v_j, u_j, v_jX_j - u_jY_j]^T$, and $\mathbf{h} = [h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8]^T$. Once $\mathbf{h}$ has been determined, the motion parameters $\boldsymbol{\omega}$ and $\mathbf{k}$ can be easily obtained [42]. Since $\mathbf{h}$ is 9-dimensional and $\mathbf{Ah} = 0$ represents a set of homogeneous equations, we need only 8 observations to solve for the optical flow [42].

When a scene consists of C independently moving objects, the motion of each object can be characterized by a different vector $\mathbf{h}_i$. In this situation, we need to solve $\mathbf{Ah}_i = \mathbf{0}$ for $i = 1, \cdots, C$. MMGLR solves this set of equations where $\mathbf{X}$ and β_i correspond to $\mathbf{A}$ and $\mathbf{h}_i$ respectively. It finds C automatically.

MMGLR requires an overspecified number (C) of initial parameter estimates. We obtain each one of these estimates by solving $\mathbf{Ah} = 0$ on a randomly selected subset of 8 observations. These C estimates are then pooled together to initialize the MMGLR algorithm. To ensure a reliable result, the initial number of models C needs to be high. However, since C decreases drastically in the subsequent iterations, this method is still efficient. Since MMGLR allows points to move from one model to another, and since fuzzy rejection allows points to change from inliers to outliers and vice versa smoothly, we can afford to use a smaller number of initializations than algorithms based on hard rejection. In both experiments described in this subsection, we use C=50.

Fig. 12(a) shows a synthetic 3-D scene consisting of 4 touching rigid objects, each undergoing a motion with different rotational and translational velocities. Fig. 12(a) displays the subsampled and scaled true optic flow field. We contaminated this optic flow field with Gaussian noise (SNR=70), and additionally altered 20% of the observations randomly to

make them outliers. The resulting optic flow field is shown in Fig. 12(b). MMGLR succeeds in determining the correct number of motion groups in the scene. It also estimates their motion parameters accurately, as shown in Table 3. Fig. 12(c) shows the segmented optic flow field where each motion group is represented by a different symbol. The correctly identified outliers (points having zero weight w_{ij} in all models) are shown as black dots in Fig. 12(c). The recovered optic flow field is shown in Fig. 12(d).

Figs. 13(a) and 13(b) show two 512×512 subimages of the 13^{th} and 14^{th} frames in a motion sequence [43] containing a moving truck. In this experiment, the background motion (due to camera panning) is treated as another motion group to create a multiple motion scenario. We selected 30 target points on the vehicle, and another 30 points from the background. The matches of these 60 points were computed using a robust matching algorithm [44] and verified manually. To illustrate the robustness of MMGLR, we added another 10 target points with erroneous matches. All 70 points are marked '+' in Figs. 13(a) and 13(b). The target points and their matches were first converted from pixel coordinates to image coordinates, and then calibrated [43]. Finally, all target points were integrated to form the mixture data set $\{(X_i, Y_i), (u_i, v_i)\}, i = 1, \cdots, 70$, where (X_i, Y_i) is the image coordinates of the i^{th} target point in the 13^{th} frame, and (u_i, v_i) is its displacement vector.

The "ground truth" for the vehicle motion is unknown. Also, since the rotation angle of the truck is too small (about 5^o), it could not be estimated reliably using two-view point correspondence and three-view line correspondence algorithms [45]. Since we are testing the robustness of MMGLR and its ability to detect multiple models, and not the performance of the linear optic flow algorithm, we compare our results with those obtained when the linear optic flow algorithm is supplied with the correct data subset for each motion (see Table 4). MMGLR was first run with only the 60 good target points, and then with the added outliers. In both cases, the algorithm was able to detect the correct number of motion groups (=2) and estimate their parameters correctly. Fig. 14 shows the partition of the optic flow field where the two motion groups and the detected outliers are denoted by different symbols.

7 Discussion and Conclusions

7.1 General Comments

RCA is an attempt at addressing the three main issues of partitional clustering algorithms (the difficulty in determining the number of clusters, sensitivity to initialization, and sensitivity to outliers), without sacrificing computational efficiency. RCA minimizes a fuzzy objective function in order to handle overlapping clusters. Constrained fuzzy memberships are used to create a competitive environment that promotes the growth of "good" clusters. Possibilistic memberships [10]are used to obtain robust estimates of the prototype parameters. Concepts from robust statistics have been incorporated into RCA to make it insensitive to outliers. To handle the region of doubt, and to reduce the sensitivity to initialization, RCA uses soft finite rejection. The agglomerative property makes it relatively insensitive to initialization and local minima effects. By using suitable distance measures, we can apply this algorithm to solve many computer vision problems. The choice of α in (10) is quite critical to the algorithm. However, α can be chosen by trial and error to produce stable results for a given application. The variety of examples presented in this paper show that this is possible, and that RCA can provide robust estimates of the prototype parameters even when the clusters vary significantly in size and shape, and the data set is contaminated.

7.2 Computational Complexity

The RCA algorithm has a computational complexity similar to that of FCM [2], which is $\mathcal{O}(NC)$ in each iteration. Here, N is the number of data points, and C is the number of clusters. However, additional time is required to estimate the weight function $w(d^2)$ which requires us to compute the median of the squared distances twice (first to compute the median, then to compute the MAD). The median of a data set can be computed iteratively using

$$x_{med} = \frac{\sum_{j=1}^{N} \frac{x_j}{|x_j - x_{med}|}}{\sum_{j=1}^{N} \frac{1}{|x_j - x_{med}|}}.$$

This procedure converges in $\mathcal{O}(\log N)$ passes through the data set. Since the distribution of the squared distances does not change significantly in one iteration, this procedure converges even faster when the median of the previous iteration is used to initialize the computation of the median of the current iteration. Thus, the overall complexity can be estimated as $\mathcal{O}(N \log N + NC)$ per iteration, or $\mathcal{O}(NK(\log N + C))$, where K is the number of iterations. It is to be noted that the value of C varies from C_{max} to C_{final}. Except for the application to motion analysis, in all other cases we use a standard algorithm such as FCM to initialize RCA. Therefore, the initialization overhead is $\mathcal{O}(NkC_{max})$, where k is a small (≈ 5) integer.

7.3 Breakdown Issues

As discussed in section 2, when C is known, the breakdown point is N_{min}/N, and when C is unknown, the breakdown is either undefined or $N_{min_{val}}/N$. These results were derived by the use of validity, and an ideal clustering algorithm would use a validity measure and an expensive exhaustive search to achieve this level of robustness [12]. However, validity measures are hard to define in practice unless the distribution of the good points is known. Moreover, deviations from the assumed distribution can occur with widely varying degrees in real applications, and it is hard to choose thresholds when their optimal values can vary widely among different data sets, and even among clusters in the same data set.

RCA is a *general purpose* algorithm that attempts to achieve robustness with reasonable computational complexity. This is the rationale behind the choice of the M-estimator to robustify RCA. This choice limits the breakdown point of RCA to $\frac{1}{p+1}$, where p is the dimensionality of the parameter vector to be estimated. However, since RCA starts with a large mber of initial prototypes, it is possible to increase its robustness under certain conditions. RCA uses the initial prototypes to generate a partition. The algorithm consists of updating the weight function for each component of the partition, then updating the memberships, and then finally updating the prototypes. This process is repeated until convergence. Since the weight function uses the median and MAD, it can tolerate up to 50% noise points (within

the component) provided it starts with a good initialization.

Let there be C actual clusters. Let the good points from the k^{th} actual cluster be given the label "k", $k = 1, 2, \cdots, C$, and let the noise points be labeled "0". Let the (hard) partition corresponding to the C_{max} initial prototypes be labeled as follows. If a given component has only noise points, it is labeled "0", otherwise it is labeled "i", where i is the label of the majority of good points in the component. Let P_i^{max} denote the largest component with the label i. For the RCA algorithm to give robust results, we require an initialization that satisfies the following conditions. (i) There exists at least one component that has the label i, for all $i = 1, \cdots, C$. (ii) The prototype corresponding to P_i^{max} is a good point of the i^{th} actual cluster. (iii) The largest component labeled "0" is smaller than P_i^{max}, $i = 1, \cdots, C$. (iv) P_i^{max} contains more than 50% of points labeled "i". Since the cluster region by definition is denser than the noise region, by using a sufficiently large number of prototypes, it is usually possible to achieve an initialization to meet these conditions in practice. Initial prototypes placed in the cluster region will naturally have larger cardinalities and those in the noise region will have smaller ones. Conditions (i)-(iv) need to be satisfied in the following iterations as well, to guarantee that the algorithm will converge to a correct result. However, since cardinalities are replaced by robust cardinalities in the subsequent iterations, it becomes easier to satisfy these conditions. When the components coalesce and form the final result, each noise point will be crisply assigned to one of the components while computing the weight function. In the worst case, all noise points can be assigned to the smallest cluster. Therefore, conditions (iii) and (iv) above translate to the requirement that the number of noise points be smaller than the cardinality of the smallest cluster. Thus, when (i)-(iv) are satisfied, RCA can achieve the theoretical breakdown point. A similar discussion applies to non-point prototypes as well, with minor modifications. In this case, each initial prototype can be generated with n data points, where n is the number of parameters in the prototype.

7.4 Initialization Issues

From the above discussion, it is clear that initialization plays a very important role in the RCA algorithm. The initialization procedure necessarily varies with the type of prototypes used, the distance measure used, the type of data, and finally the application. We now outline some guidelines for initialization.

We can compute a theoretical value for the initial number of clusters, C_{max}, as follows. Let there be C_{exp} number of actual clusters expected in the data set, let N_i denote the cardinality of cluster i, and let n be the number of points required to generate a prototype. If we randomly pick n points to generate a prototype, then the probability p that we pick C_{exp} good prototypes, one from each cluster, is given by $p = \prod_{i=1}^{C_{exp}} \left\{ \frac{C(N_i,n)}{C(N,n)} \right\}$. If this selection is repeated K times, the probability that one of these selections generates good prototypes for all C_{exp} clusters is given by $P_g = 1 - (1-p)^K$. For a given value of P_g, we can compute the value of K as, $K = \lceil \frac{\log(1-Pg)}{\log(1-p)} \rceil$, and C_{max} can be estimated as $C_{max} = K \times C_{exp}$. This value of C_{max} grows exponentially with C_{exp} and n, and therefore is unrealistic.

In practice, an existing clustering algorithm (such as FCM [2], GK [36], AFC [3]) can be used for initialization. At the end of such an initialization, although not all C_{max} prototypes are expected to be good, we can assume that each of the C_{exp} clusters has a fairly high probability, P_i^{init}, of being represented by one of the C_{max} initial prototypes. For example, consider the case of finding lines in a 2-D data set, i.e. $n = 2$. If there are N total points, there are $N(N+1)/2$ possible ways to pick a pair of points, and hence $N(N+1)/2$ possible random initializations for a line. However, most of these initializations involve points that are far away from each other and constitute poor initializations. On the other hand, an algorithm such as AFC will use only nearby points, and the probability that two nearby points belong to the same line is high. If the data set is an image, then by dividing the image into small windows and applying a conventional clustering algorithm with a suitable number of clusters in each window can dramatically increase the value of P^{init}. The probability that all C_{exp} clusters are represented by the initialization is given by $p = \Pi_{i=1}^{C_{exp}} P_i^{init}$. In this case, a much

smaller number of initial clusters will suffice.

Based on the above discussion, we suggest the following rules of thumb. For general clustering, choose $C_{max} \approx \frac{N}{10*n}$, and use a simple clustering algorithm (such as FCM) to generate the initial prototypes. Since good points are by definition in dense regions, this initialization can be expected to meet the conditions discussed in the previous sub-section. The case of plane and surface fitting can be handled by dividing the image into small windows and applying a suitable clustering algorithm in each window. In the case of regression, the above initialization techniques are no longer applicable. Hence, we use a random sampling procedure to generate the prototypes. Because of this randomness, we require a larger value for C_{max}. In our applications, we set $C_{max} \approx \frac{N}{n}$.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was partially supported by a grant from the Office of Naval Research (N00014-96-1-0439).

References

[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[2] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[3] R. N. Davé, "Use of the adaptive fuzzy clustering algorithm to detect lines in digital images," *Intelligent Robots and Computer Vision VIII*, vol. 1192, pp. 600–611, 1989.

[4] R. N. Davé and K. Bhaswan, "Adaptive fuzzy c-shells clustering and detection of ellipses," *IEEE Trans. Neural Network*, vol. 3, no. 5, pp. 643–662, May 1992.

[5] R. Krishnapuram, H. Frigui, and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation," *IEEE Trans. Fuzzy Systems*, vol. 3, no. 1, pp. 29–60, Feb. 1995.

[6] H. Frigui and R. Krishnapuram, "A robust clustering algorithm based on competitive agglomeration and soft rejection of outliers," in *Proceeding of the IEEE Conf. on Computer Vision and Pattern Recognition*, San Fransisco, California, 1996, pp. 550–555.

[7] H. Frigui and R. Krishnapuram, "Clustering by competitive agglomeration," *Pattern Recognition*, vol. 30, no. 7, pp. 1223–1232, 1997.

[8] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

[9] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum Press, New York, 1988.

[10] R. Krishnapuram and J. Keller, "A possibilistic approach to clustering," *IEEE Trans. Fuzzy Systems*, vol. 1, no. 2, pp. 98–110, May 1993.

[11] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust Statistics the Approach Based on Influence Functions*, John Wiley & Sons, New York, 1986.

[12] R. N. Davé and R. Krishnapuram, "Robust clustering methods: A unified view," *IEEE Trans. Fuzzy Systems*, vol. 5, no. 2, pp. 270–293, 1997.

[13] Y. Ohashi, "Fuzzy clustering and robust estimation," in 9^{th} *Meeting of SAS Users Group International*, Hollywood Beach, Florida, 1984.

[14] R. N. Davé, "Characterization and detection of noise in clustering," *Pattern Recognition Letters*, vol. 12, no. 11, pp. 657–664, Nov. 1991.

[15] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Interscience, New York, 1990.

[16] L. Bobrowski and J. C. Bezdek, "c-means clustering with the l_1 and l_∞ norms," *IEEE Trans. SMC*, vol. 21, no. 3, pp. 545–554, 1991.

[17] P. R. Kersten, "The fuzzy median and the fuzzy mad," in *Proc. of ISUMA/NAFIPS*, College Park, September 1995, pp. 85–88.

[18] H. Frigui and R. Krishnapuram, "A robust clustering algorithm based on the m-estimator," in *Proceedings of First Intl. Conf. on Neural, Parallel and Scientific Computations*, Atlanta, May 1995, vol. 1, pp. 163–166.

[19] P. J. Huber, *Robust Statistics*, John Wiley & Sons, New York, 1981.

[20] J. Kim, R. Krishnapuram, and R. N. Davé, "On robustifying the c-means algorithms," in *Proceedings of ISUMA/NAFIPS*, College Park, MD, September 1995, pp. 630–635.

[21] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, John Wiley & Sons, New York, 1987.

[22] Y. Choi and R. Krishnapuram, "Fuzzy and robust formulations of maximum- likelihood-based gaussian mixture decomposition," in *Proceedings of the Intl. Conf. on Fuzzy Systems*, New Orleans, September 1996, pp. 1899–1905.

[23] O. Nasraoui and R. Krishnapuram, "A genetic algorithm for robust clustering based on a fuzzy least median of squares criterion," in *Proceedings of the North American Fuzzy Information Processing Society Conf.*, Syracuse, NY, September 1997, pp. 217–221.

[24] R. N. Davé, "Generalized noise clustering as a robust fuzzy c-m-estimators model," in *Proceedings of the North American Fuzzy Information Society Conference*, Pensacola, August 1998, pp. 256–260.

[25] J. C. Bezdek and N. R. Pal, "Some new indices for cluster validity," *IEEE Trans. SMC, Part: B*, vol. 28, no. 3, pp. 301–315, 1993.

[26] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 11, no. 7, pp. 773–781, July 1989.

[27] R. Krishnapuram and C. P. Freg, "Fitting an unknown number of lines and planes to image data through compatible cluster merging," *Pattern Recognition*, vol. 25, no. 4, pp. 385–400, 1992.

[28] R. N. Davé and K. J. Patel, "Progressive fuzzy clustering algorithms for characteristic shape recognition," in *North American Fuzzy Information Processing Society*, Toronto, 1990, pp. 121–124.

[29] J. M. Jolion, P. Meer, and S. Bataouche, "Robust clustering with applications in computer vision," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 13, no. 8, pp. 791–802, Aug. 1991.

[30] X. Zhuang, Y. Huang, K. Palaniappan, and J. S. Lee, "Gaussian mixture density modeling, decomposition and applications," *IEEE Trans. Image Processing*, vol. 5, pp. 1293–1302, Sept. 1996.

[31] X. Zhuang, T. Wang, and P. Zhang, "A highly robust estimator through partially likelihood function modeling and its application in computer vision," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 14, no. 1, pp. 19–34, Jan. 1992.

[32] T. Darrell and P. Pentland, "Cooperative robust estimation using layers of support," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 17, no. 5, pp. 474–487, May 1995.

[33] C. V. Stewart, "Minpran: A new robust estimator for computer vision," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 17, no. 10, pp. 925–938, Oct. 1995.

[34] D. C. Hoaglin, F. Mosteller, and Ed. J. W. Tukey, *Understanding Robust and Exploratory Data Analysis*, Wiley, New York, 1983.

[35] G. S. Sebestyen and Roemder, *Decision-Making Process in Pattern Recognition*, Macmillan Company, New York, 1962.

[36] E. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *IEEE CDC*, San Diego, California, 1979, pp. 761–766.

[37] R. Krishnapuram and J. Keller, "Fuzzy and possibilistic clustering methods for computer vision," in *Neural and Fuzzy Systems*, S. Mitra, M. Gupta, and W. Kraske, Eds., vol. IS 12, pp. 135–159. SPIE Institute Series, 1994.

[38] A. Hoover, G. J. Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher, "An experimental comparison of range image segmentation algorithms," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 18, no. 7, pp. 673–689, July 1996.

[39] G. Taubin, "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with application to edge and range image segmentation," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 13, no. 11, pp. 1115–1138, Nov. 1991.

[40] H. Frigui and R. Krishnapuram, "A comparison of fuzzy shell-clustering methods for the detection of ellipses," *IEEE Trans. Fuzzy Systems*, vol. 4, no. 2, pp. 193–199, May 1996.

[41] Y. Huang, K. Palaniappan, X. Zhuang, and J. E. Cavanaugh, "Optic flow field segmentation and motion estimation using a robust genetic partitioning algorithm," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 17, no. 12, pp. 1177–1190, Dec. 1995.

[42] X. Zhuang, T. S. Huang, and R. M. Haralick, "A simplified linear optic flow-motion algorithm," *Computer Vision, Graphics, and Image Processing*, vol. 42, pp. 334–344, 1988.

[43] Y. Liu and T. S. Huang, "A sequence of stereo image data of a moving vehicle in an outdoor scene," Tech. Rep., Beckman Institute, University of Illinois at Urbana-Champaign, 1990.

[44] Z. Zhang, R. Deriche, O. Faugeras, and Q. T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial Intelligence Journal*, vol. 78, pp. 87–119, Oct. 1995.

[45] Y. Liu and T. S. Huang, "Vehicle-type motion estimation from multi-frame images," *IEEE Trans. Patt. Analysis Mach. Intell.*, vol. 15, no. 8, pp. 802–808, Aug. 1993.

LIST OF FIGURES

LIST OF TABLES

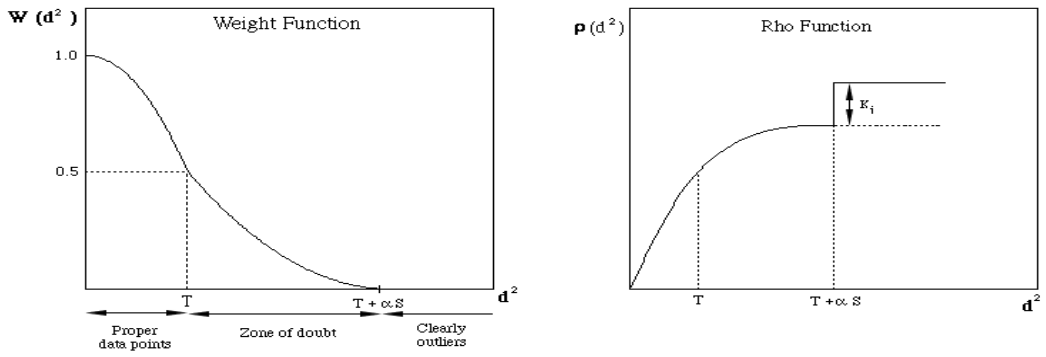
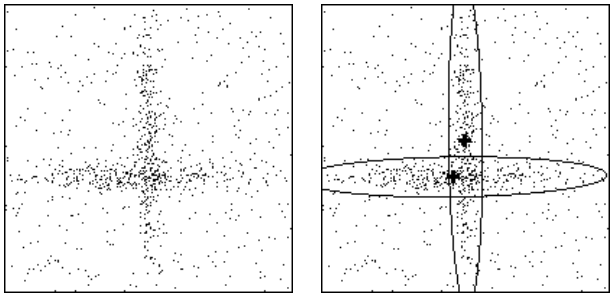


Figure 1:

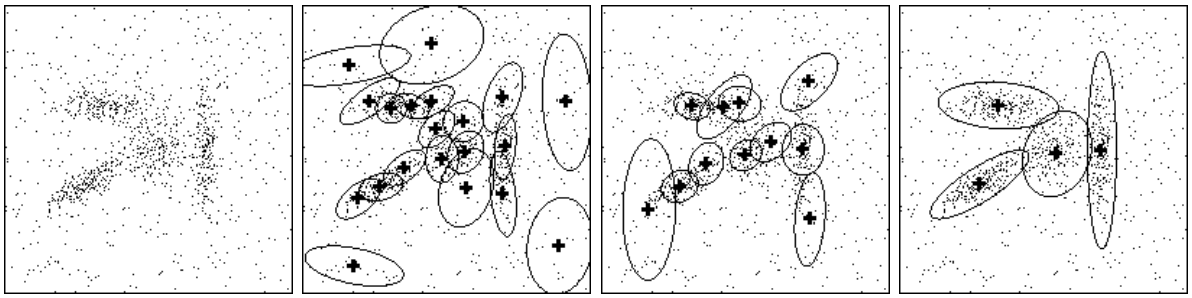


(a) (b)

Figure 2:

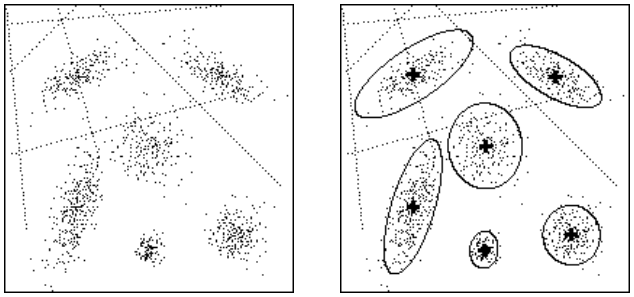
Table 1:

Alg.	**GT regions**	**correct detection**	**ang diff. (std. dev.)**	**over-seg.**	**under-seg.**	**missed**	**noise**
UE	15.2	13.4	1.6^o (0.9)	0.4	0.2	1.1	0.8
RCA	15.2	13.0	1.5^o (0.8)	0.8	0.1	1.3	2.1
UB	15.2	12.8	1.3^o (0.8)	0.5	0.1	1.7	2.1
USF	15.2	12.7	1.6^o (0.8)	0.2	0.1	2.1	1.2
WSU	15.2	9.7	1.6^o (0.7)	0.5	0.2	4.5	2.2



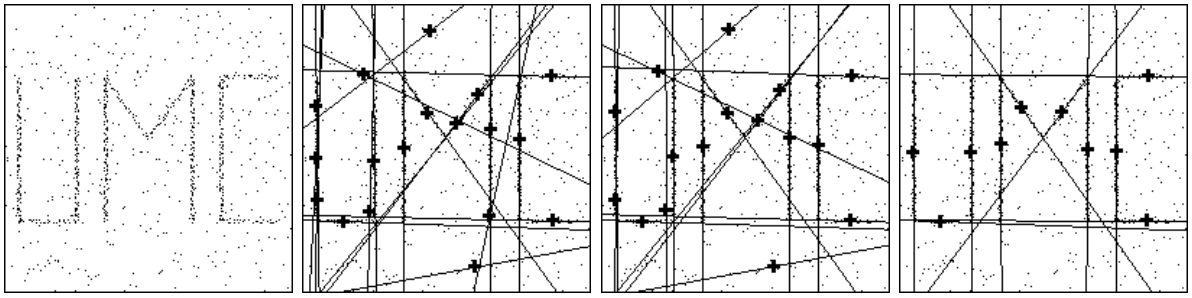
(a) (b) (c) (d)

Figure 3:



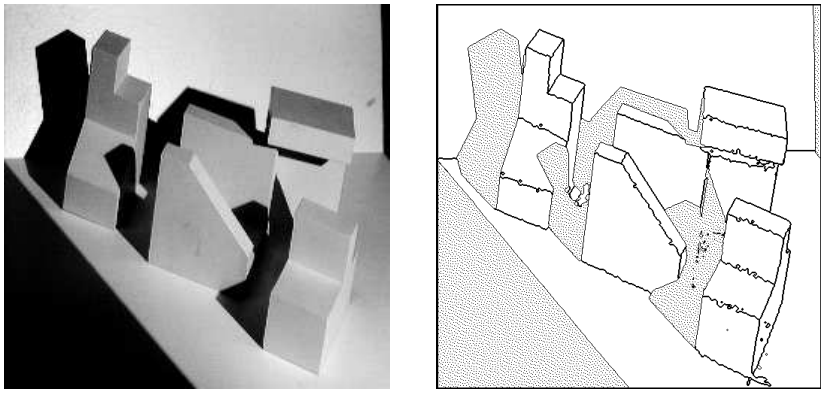
(a) (b)

Figure 4:



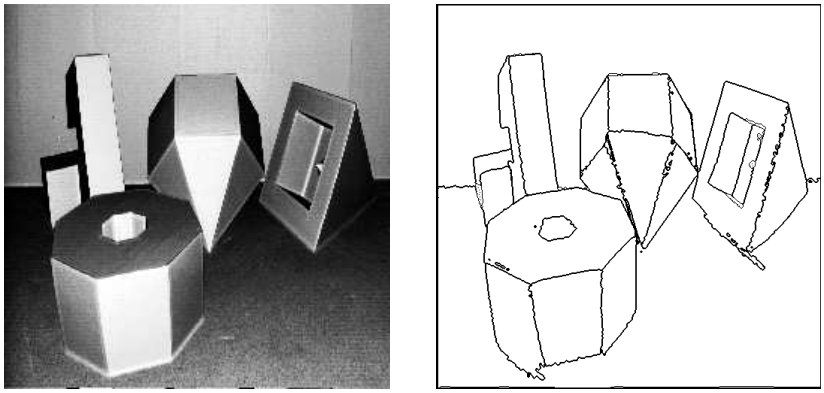
(a) (b) (c) (d)

Figure 5:



(a) (b)

Figure 6:

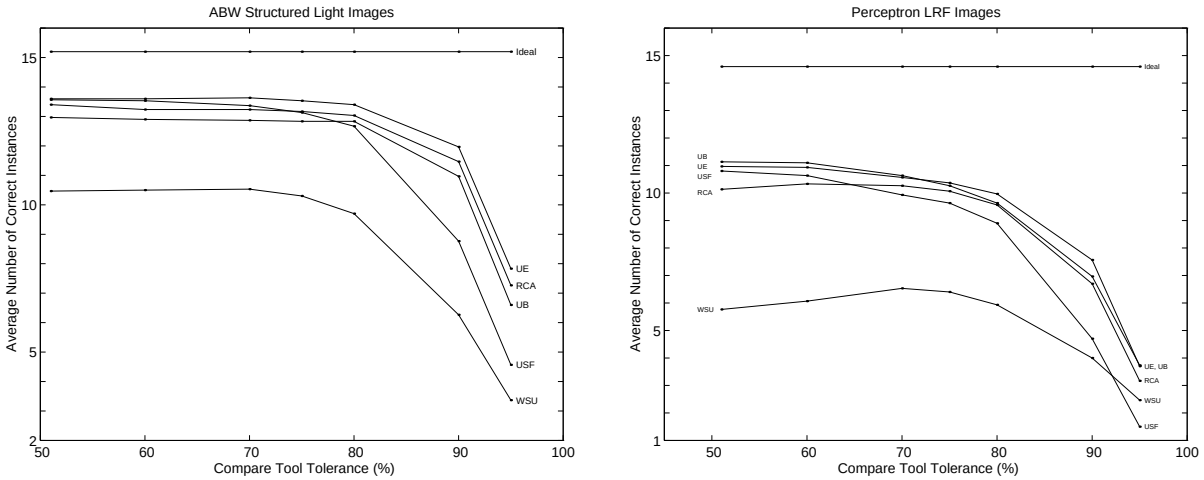


(a) (b)

Figure 7:

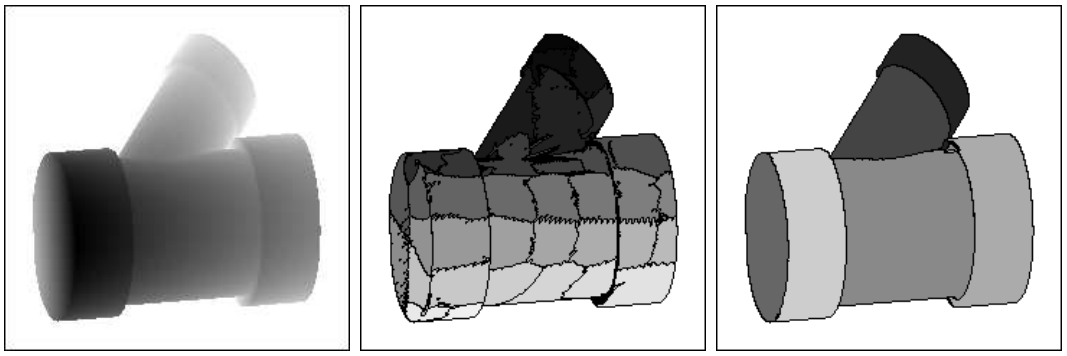
Table 2:

Alg.	**GT regions**	**correct detection**	**ang diff. (std. dev.)**	**over-seg.**	**under-seg.**	**missed**	**noise**
UE	14.6	10.0	2.6^o (1.5)	0.2	0.3	3.8	2.1
UB	14.6	9.6	3.1^o (1.7)	0.6	0.1	4.2	2.8
USF	14.6	8.9	2.7^o (1.8)	0.4	0.0	5.3	3.6
WSU	14.6	5.9	3.3^o (1.6)	0.5	0.6	6.7	4.8
RCA	14.6	9.6	2.6^o (1.6)	0.7	0.2	3.7	3.6



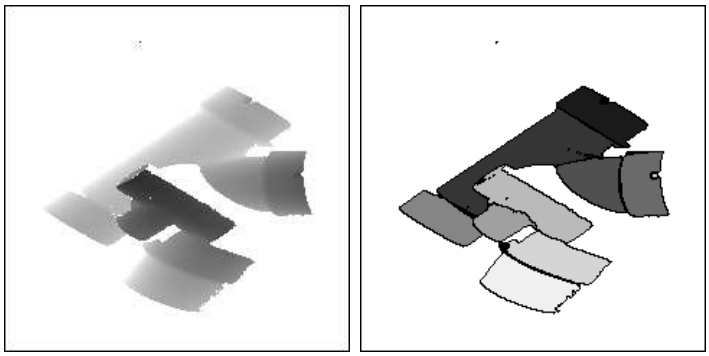
(a) (b)

Figure 8:



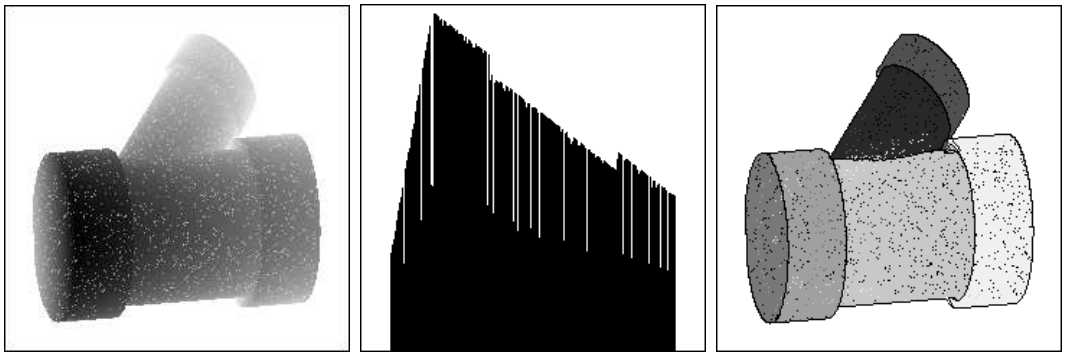
(a) (b) (c)

Figure 9:



(a) (b)

Figure 10:



(a) (b) (c)

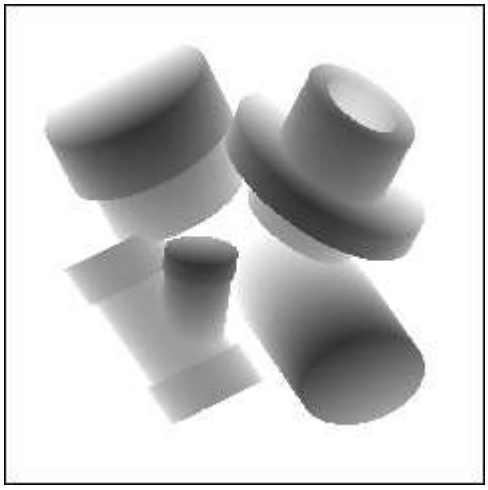
Figure 11:

Table 3:

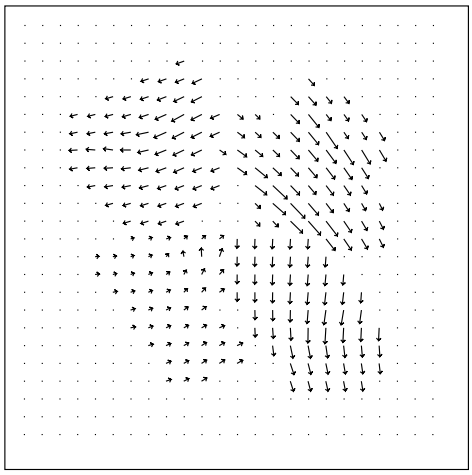
Object #	**Parameters**	**Angular Velocity**			**Translational Velocity**		
		ω_1	ω_2	ω_3	k_x	k_y	k_z
1	Actual	0.200	-0.800	-0.200	20.00	50.00	-100.00
	Estimated	0.198	-0.801	-0.201	20.04	49.97	-100.01
2	Actual	0.300	-0.100	1.000	10.00	-75.00	-50.00
	Estimated	0.298	-0.099	1.001	10.03	-74.98	-50.01
3	Actual	0.100	0.800	-0.300	0.00	-50.00	75.00
	Estimated	0.101	0.799	-0.302	0.01	-49.97	75.02
4	Actual	1.000	-0.100	0.200	-20.00	20.00	-75.00
	Estimated	0.997	-0.102	0.198	-20.06	19.93	-75.07

Table 4:

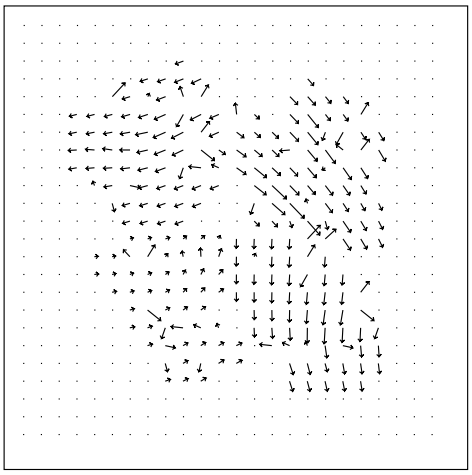
Algorithm	**Motion Group**	**Rotation Axis**			**Angle (deg.)**	**Translation**		
		n_1	n_2	n_3		k_x	k_y	k_z
Linear	*Truck*	-0.988	0.107	-0.107	2.137	-0.158	-0.015	0.987
	background	0.893	-0.413	-0.176	0.140	0.476	-0.069	0.877
MMGLR (No outliers)	*Truck*	-0.988	0.107	-0.107	2.137	-0.157	-0.015	0.988
	background	0.895	-0.404	-0.189	0.143	0.517	-0.059	0.854
MMGLR (10 outliers)	*Truck*	-0.988	0.111	-0.106	2.097	-0.158	-0.021	0.988
	background	0.895	-0.404	-0.189	0.143	0.517	-0.059	0.854



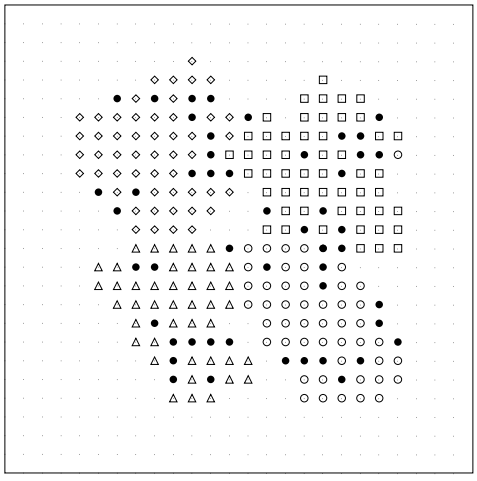
(a)



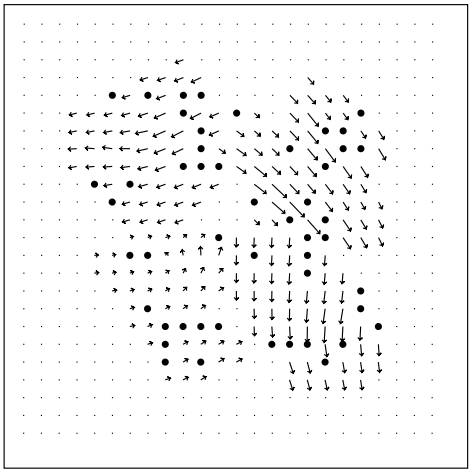
(b)



(c)

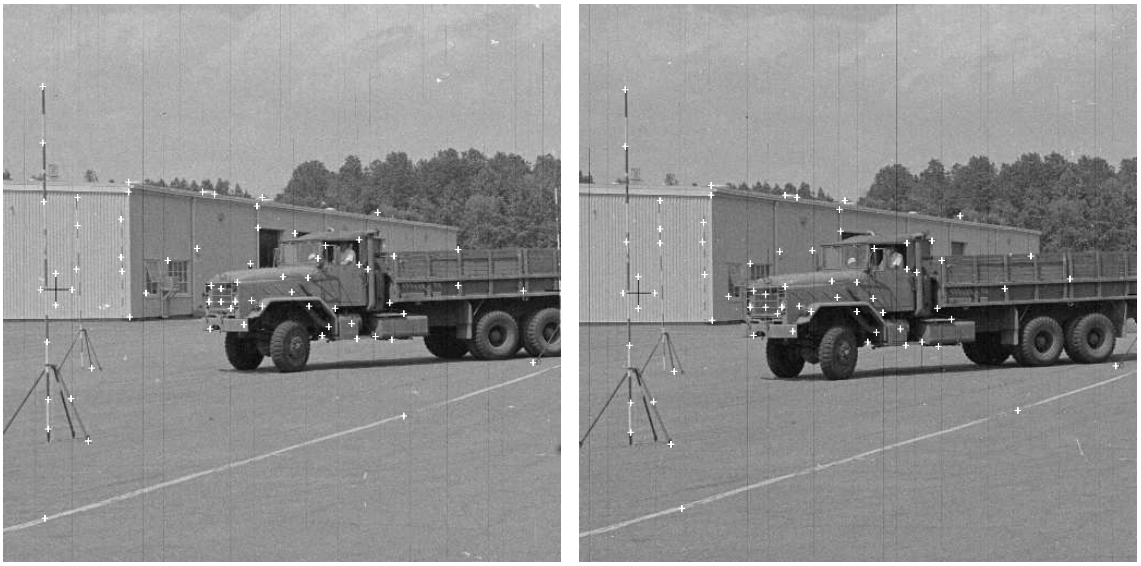


(d)



(e)

Figure 12:



(a) (b)

Figure 13:

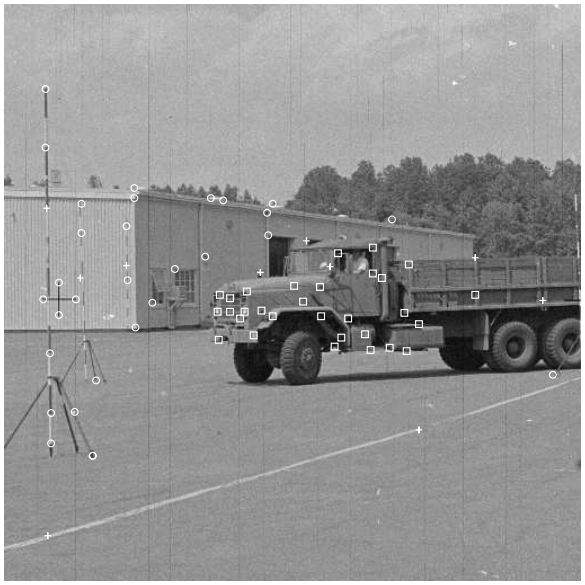


Figure 14: