When Does a Robot Perceive a Dynamic Object?

K. Madhava Krishna*[*] and Prem K. Kalra
Department of Electrical Engineering
Indian Institute of Technology
Kanpur, India 208 016

Received 11 April 2001; accepted 5 September 2001

Robots operating in a real-time environment encounter both stationary and moving objects that need to be negotiated using different schemes in general. Motion planning in a dynamic environment entails tracking moving objects and predicting their future positions. However, this requires as a first step the classification of the objects present in the environment as static or dynamic objects, a step that somehow seems to have been overlooked in the literature dealing with navigation in dynamic environments. Presented here are four schemes for perceiving the presence of dynamic objects in the robot's neighborhood. The first approach incorporates a network architecture that classifies the robot's experience of the environment in terms of spatio-temporal sensor patterns as an experience of a static or dynamic object. The second method detects motion by observing changes in the map of the environment it builds and updates. The remaining two approaches use a strategy for representing the objects in the environment through clusters; inspecting the characteristics of the clusters reveals the dynamic objects. These methods are denoted as STA (spatio-temporal approach), MBA (model-based approach), and CBAI and CBAII (cluster-based approach I and II), respectively. The methods have been tested in environments that contain multiple dynamic objects amidst static ones and their efficacy established. A brief comparison of these approaches in terms of criteria critical for real-time collision avoidance has also been presented. © 2002 John Wiley & Sons, Inc.

1. INTRODUCTION

Literature abounds with approaches for dealing with collision avoidance in a dynamic environment. Some of these approaches require prior knowledge of the position and trajectories of the objects.[1,2] Other approaches compute the robot's path in a static environment using global strategies.[3–5] The computed path is replanned when dynamic objects are introduced that crisscross the planned path. Later approaches that involve fuzzy inferencing or neuro-fuzzy based control have also been reported.[6,7] Recent strategies have extended this to situations that involve

[*]To whom all correspondence should be addressed; e-mail: kkrishna@iitk.ac.in.

cooperative collision avoidance of multiple robots that plan and execute a task.[8]

In all these approaches, either the exact position or velocity information of the dynamic objects are assumed to be available either before the path is planned or during real time. However, it is difficult to provide such information to the robot in real time. Based on this information, the algorithms predict the future position of the dynamic object and the collision time. But the initial processing stage of classifying an object as static or dynamic based on sensor readings is not mentioned in these approaches. Only after the robot classifies an object as static or dynamic does the question arise of predicting its future position. Hence, even with algorithms that perform real-time collision avoidance in which no prior information regarding the objects is available, there is a tacit assumption regarding whether the real-time data belongs to a static or dynamic object. Recently, Song and Chang have circumvented this problem implicitly by predicting the future sensor readings based on the present and past readings.[9,10] Their algorithm makes estimates of the expected sensor readings in the subsequent sample based on prior observations. The estimates are made for both static and dynamic objects and hence there is no explicit classification regarding the nature of the object with respect to its static or dynamic attributes.

Presented in this paper are four explicit approaches for detecting the presence of dynamic objects in a robot's vicinity. Such explicit detection obviates the need for tracking and predicting the future positions of all the objects in the neighborhood, as only the dynamic objects need to be tracked. In some sense it is also more consistent with human intuition, as humans are generally aware whether they are cognizing static or dynamic objects and plan their paths accordingly. The first approach (STA) uses a network architecture that classifies a stream of sensor patterns as that obtained from a static or dynamic object. It operates directly on the range data obtained from sonar without the necessity for a map-building affair. The other three approaches involve representing the contents of the environment on a map. In the first of these approaches (MBA), a dynamic object is perceived when significant changes in the features of an object (as represented in the map) are observed over a time window. The remaining two approaches (CBA-I and CBA-II) partition the data into clusters. The clusters that are formed indicate the number of objects present in the neighborhood. The properties of the cluster determine the nature of the object in terms of its static or dynamic attribute. The first algorithm (CBA-I) self-organizes and determines the number K of clusters or objects in the neighborhood of the robot. The algorithm starts with the initial value of K at 1. For the second approach we have used the Gustafson-Kessel (GK) algorithm[11] with the initial number of clusters fixed to a value higher than the number of objects a robot generally encounters in five samples. The exact number of clusters is then found through cluster merging procedures.

The rest of the paper is organized as follows: Section 2 deals in detail with the four approaches employed for perceiving the existence of dynamic objects; Section 3 analyzes the performance of these approaches through simulations; and Section 4 winds up the paper with concluding remarks.

2. PERCEIVING THE DYNAMIC OBJECT

The problem of ascertaining the presence of dynamic objects in a robot's vicinity becomes all the more obscure when information about the environment is obtained from range sensors. In vision-based detection and tracking systems a single snapshot can furnish the essential details and a holistic representation of the environment can be obtained. Data obtained from range sensors on the other hand are nothing but discretized spatial samples, which represent those parts of the local environment that have reflected the beam emitted by these sensors. To obtain a unified picture of the environment based on a temporal sequence of such spatially discrete samples becomes problematic, especially if the environment is nonstationary and the sensors are themselves subjected to translation and rotation. In spite of these difficulties, range sensors (especially lasers) have been popular and found suitable for real-time navigation purposes. In the subsequent sections we present four algorithms that offers a feasible solution for perceiving dynamic objects based on range data.

2.1. Spatiotemporal Approach (STA)

Figure 1 depicts the sensor arrangement used while navigating in a dynamic environment. The

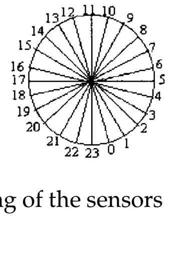

Figure 1. Numbering of the sensors placed on the circumference of the robot.

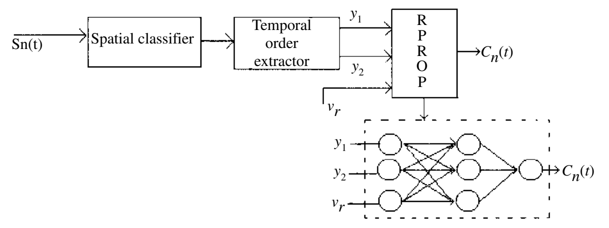

Figure 2. Network architecture for dynamic object perception.

arrangement consists of a ring of 24 sensors placed on the circumference of the robot. The robot is modeled as a circle and the sensors are 15 degrees apart with respect to the robot's center. Each sensor transmits a beam with a conical spread of 10 degrees. It is assumed that the objects are of such size that none of them falls completely within the blind zone of a sensor. In other words, an object that is within the detecting range of the sensor ring shall not go undetected.

The network architecture used for dynamic object perception is shown in Figure 2. This architecture is an extension of the architecture introduced in an earlier paper for detecting local minimum traps.[12] The first block in Figure 2 is the fuzzy spatial classifier and the second block is termed the temporal order extractor. The outputs of the second block feed to a resilient propagation network (RPROP),[13] a variant of the ubiquitous back propagation algorithm.[14] The functions of these blocks will be clarified shortly.

The problem is one of classifying an experience of a stream of sensor patterns as an experience of either a static or dynamic object. The difficulties mentioned earlier can be resolved to a considerable extent through an artificial neural network (ANN) model. ANN is specially suited for instances where the development of mathematical models is difficult, uncertain, and error prone. The objective is, given the average robot velocity v_r and a temporal sequence of range readings obtained by a sensor, for the RPROP network to learn to classify the sequence as an experience of a static or a dynamic object. However, the number of range measurements that are required for an appropriate classification cannot be ascertained. This depends on the angular separation of the object's direction of approach and the robot's heading direction. For objects whose direction of approach is more or less parallel to the robot's direction of motion, a history of three or four measurements would be sufficient to discern the presence of a dynamic object. For objects whose motion direction is more transverse, a longer temporal sequence is required. The differences are due to the decrease in the relative velocity of the object with respect to the robot as its direction of approach varies from parallel to transverse. Objects whose motion direction is almost perpendicular to the robot's own direction (Fig. 3(a)) are classified as dynamic, not based on the philosophy of relative velocity but through a sudden decrease in the range measurements when a sensor suddenly detects the object at *near* ranges. These are depicted in Figures 4(a) and (b), where sensor 11 detects an object suddenly at a *near* range. In order to train the network with a fixed number of range readings as input, the following procedure is adopted.

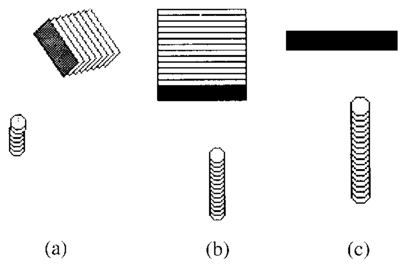

Figure 3. (a) Acquiring patterns from a transverse dynamic object. (b) Acquiring patterns from a parallel dynamic object. (c) Acquiring patterns from a static obstacle.

The robot is made to approach static and dynamic objects at various orientations and velocities, as shown in Figures 3(a–c). The range data is processed as follows. For any sample of the environment, the range data acquired by each sensor forms an input to the fuzzy spatial classifier. The spatial classifier classifies the reading into one of the four classes (*far, medium, near,* and *very near*). The fuzzy membership function that does this classification is shown in Figure 5. A temporal sequence of classes is formed for each sensor with every sample, the fuzzy classifier appending the class for that sensor. A typical sequence for a sensor

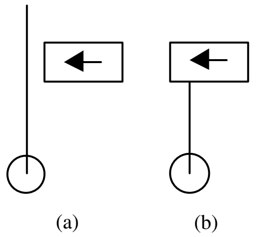

Figure 4. (a) At an instant prior to detection, sensor 11 measures a free space. (b) In the next instant, sensor 11 detects the object suddenly at near range.

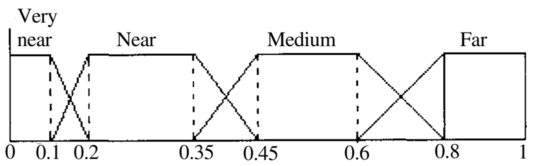

Figure 5. Spatial classification of a sensor reading through fuzzy membership functions in dynamic environments.

looks like [*bbbfffmm*...]. This sequence can be interpreted as three experiences of blanks (the sensor did not detect an object) followed by three experiences of an object at *far* range succeeded by two experiences at *medium* range. However, what is important is not the temporal order of classes but the temporal order of number of occurrences of each class in the sequence, termed VOC (vector of occurrences of classes). The VOC is extracted until the first range reading classified as *near* by the spatial classifier is encountered. The VOC is then tagged as static or dynamic, depending on whether it was acquired while the robot was approaching a static or a dynamic object. In Figure 3(c) the VOC for the eleventh sensor is given by the vector [9, 9], indicating that the robot's experience of the static object through sensor 6 consisted of 9 experiences at *far* ranges, followed by another 9 experiences at a *medium* range, whereas the experience of the dynamic object in Figure 3(b) was a consolidated experience of 4 at far ranges and 3 at medium range, represented by the VOC [4, 3]. The experience of the transverse dynamic object in Figure 4(b) gets coded as [0, 0], indicating that sensor 11 suddenly experienced the object at *near* range with no prior experiences at *far* or *medium* ranges. The robot's average velocity during both the simulations was 3.2 pixels/sample. With the robot's velocity at 3.2 pixels/sample, a VOC of [9, 9] is tagged as 0, indicating it as an experience of a static object, whereas [4, 3] and [0, 0] are tagged as 1, indicating a dynamic object. The range patterns are acquired for varying robot and object velocities. The acquired data is processed as described, and tags are affixed. A data set is thus formed, with the input vector consisting of the robot's velocity and the VOC and the output being the static or dynamic tag associated with the input vector. A resilient back-propagation algorithm was used for learning the input-output mapping, thereby learning to classify a consolidated experience of a particular sensor as an experience of a static versus a dynamic object by that sensor. Figure 6 shows the error decay curve during the training process. The inputs to the RPROP network are the robot's average velocity and the VOC denoted by the vector [y_1, y_2]

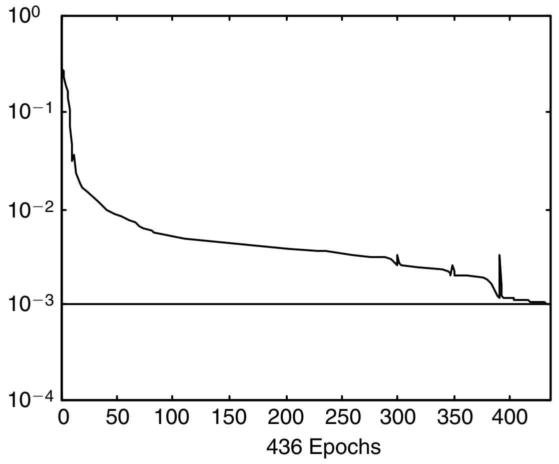

Figure 6. The error decay curve while training experiences of static and dynamic objects.

in the lower half of Figure 2, where the structure of the RPROP network used is shown. The temporal order extractor (Fig. 2) extracts the VOC from a given sequence of classes. The robot's velocity was varied between 2 to 5 pixels per sample while the velocities of the dynamic object ranged between 3 and 6 pixels per sample during data acquisition.

2.1.1. Postprocessing Module

Once the presence of dynamic objects is identified, their features are extracted. The features extracted are the approximate endpoints of the object, which are required for tracking and avoiding these objects. At this stage the requirement of a map becomes indispensable. The postprocessing module makes use of a feature extractor (FE) that extracts the features as follows.

Demarcating Distinctly Separable Objects. Consider two objects A and B in Figure 7, which were detected as dynamic by sensors 10 and 13, respectively. The FE extracts the remaining parts of both the objects by a recursive least-squares fit from the range readings of sensors 10, 9, 8, and 7 for A and sensors 12, 13,

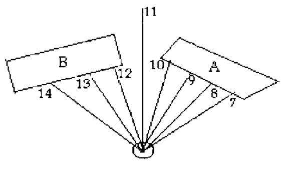

Figure 7. Object A detected by sensors 7–10 and object B by 12–14, with 11 indicating free space.

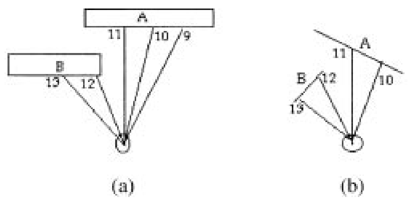

Figure 8. (a) Objects demarcated by a sudden jump between the readings of sensors 11 and 12. (b) Objects demarcated by a pattern of increase-decrease in a counterclockwise scan.

and 14 for B. It demarcates A and B as distinct objects by virtue of the free space reading due to sensor 11 between the readings of 10 and 12.

Demarcating Partly Occluded Objects. Consider a dynamic object A occluded by another dynamic object B in Figure 8(a). Since there is no free space between the two, the FE initially considers all the sensor readings arising due to sensors 9–13. These objects are demarcated by observing a sudden jump within a contiguous set of sensor readings (Fig. 8(a)), or through a pattern of increasing range readings followed by a decrease (Fig. 8(b)) during a counterclockwise scan. Further the FE resorts to a recursive least-squares (RLS) fit as a final procedure for separating the visible edges of occluding objects that do not get demarcated while looking for above changes in patterns. The RLS also parameterizes the visible edges in terms of their slopes and intercepts.

Handling Wrong Classifications. Prior to extracting the features of the dynamic object, a check is required for handling some of the inaccuracies in classification. There are situations when the consolidated experience of a particular sensor can get classified as dynamic experience although it was the experience of a static object. Figures 9(a) and (b) indicate an example of such a situation. Figure 9(a) depicts the instant just prior to the instant when sensor 5 detects an object.

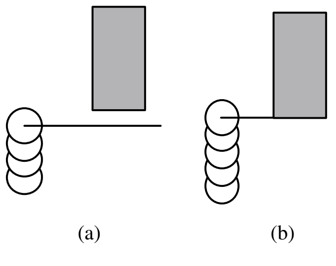

Figure 9. (a) The instant just prior to the detection of the object on the right by sensor 5. (b) Sensor 5 begins detecting the obstacle at near range.

Figure 9(b) depicts the instant when sensor 5 detects an object suddenly at a near range. In these circumstances the VOC would be [0, 0], which gets classified as an experience of a dynamic object irrespective of the robot's velocity. This is due to a sudden decrease in readings of the sensor. Whenever such a sudden decrease in a sensor reading occurs, a check is made for the readings of that sensor adjacent to it but closer to the center sensor (sensor 11). If the most recent reading is comparable, then the result of classification of the experience of the *adjacent sensor* is assigned to the sensor that experienced the sudden decrease. If the readings are not comparable, then the experience of the sensor that detected the sudden decrease is classified as an experience of a dynamic object. By "comparable" we mean that adjacent readings must have a similar classification by the fuzzy rule base, or can classified as belonging to adjacent fuzzy sets. The readings are noncomparable if they are classified as belonging to fuzzy categories that are not adjacent or same, such as near and far, or very near and medium. (It is admitted that this is not the best way to compare adjacent sensor readings and more precise possibilities could exist). This part of the postprocessing scheme is depicted through the truth table shown in Table I. The third column denoted by C takes a value 1 if the adjacent sensor readings are comparable and 0 otherwise. The first and second columns represent the output for the RPROP network for the adjacent sensors m and n, denoted in the table as C_m and C_n. Here $m = n-1$ or $n+1$ depending on whether the sensor is to the left or the right of sensor 11, the center sensor with respect to the robot's heading direction. The realization of the truth table is shown in Figure 10. The figure also

Table I. Truth table for the postprocessing stage of STA. C is the comparator output that compares adjacent sensor readings, C_n is the experience of sensor S_n classified as static ($C_n = 0$) or dynamic ($C_n = 1$) by the RPROP network. C_m is the experience of sensor S_m adjacent to S_n and closer to the center sensor. With respect to the sensor arrangement of Figure 1, $m = n+1$ if $n < 11$ and $m = n-1$ otherwise. Inputs are C_m, C_n, and C and output is Y_n.

C_m	C_n	C	Y_n
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

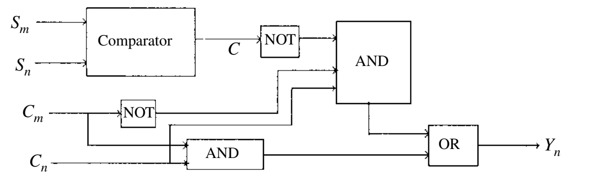

Figure 10. Realization of the postprocessing module of STA.

indicates a comparator that compares the most recent reading of the sensor, s_n, and that of the sensor adjacent to it and closer to center sensor, denoted as s_m. The essence here is whenever a sensor detects an experience that corroborates with a transverse object, the classification of that object by the sensor adjacent to it and closer to the center is given preference, provided of course that the adjacent sensor had experienced the object earlier.

The overall STA is depicted in Figure 11 with the preprocessing block consisting of the fuzzy classifier and temporal order extractor followed by the RPROP network and the postprocessing module of Figure 10. The net output is binary, indicating an experience of static (0) or dynamic (1) objects by sensor s_n.

As far as objects approaching the robot from behind, an elaborate classification is not required. Here the concern is only with those objects whose velocity is greater than the velocity of the robot. Objects whose velocities are less than that of the robot do not pose a threat, since the relative distance of separation only increases with every sample. Hence, for classifying objects that approach the robot from behind as dynamic, one need only check for an average decrease in distance of separation over a time window. They are so classified when the second component in the VOC of a rear sensor takes a finite value, indicating a decrease in the experience of the object from *far* to *medium* range.

2.2. The Model-Based Approach

The STA detects motion in the proximity of the robot solely based on the changes in range data over a time window. When the robot rotates, the sensors can see an entirely different environment from what they had seen an instant before, though there has been no drastic change in the environment itself. This can result in wrong classifications. The STA overcomes these erroneous classifications to a reasonable extent by an extrapolation scheme. Based on the range samples acquired until the instant of rotation, the STA completes the unfinished VOC by this scheme, and the RPROP network through the extrapolated VOC perceives the dynamic objects. Nonetheless, the scheme is still unreliable when the robot must undergo rotations over many consecutive samples to avoid successively encountered objects. To perceive motion based on the samples of the environment during rotational displacements of the robot entails, as a first step, correlating the range data of those samples to its actual regions in space. The correlation is thus a representation of the robot's environment in a reference frame and is a map-building affair. Motion can then be perceived by detecting changes in the map over a temporal window. The MBA builds a map of the robot's environment, extracts the features of the objects, tracks the features over the samples, and detects motion by detecting changes in the representation of these features. The various preprocessing modules involved before an object is deemed fit for classification by the MBA are discussed briefly in the following.

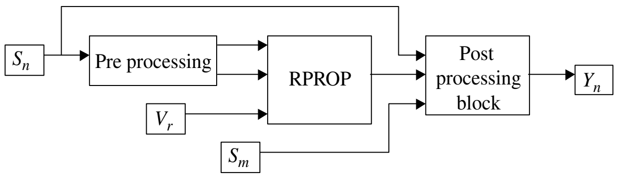

Figure 11. Overall STA.

2.2.1. Preprocessing Modules in MBA

The preprocessing modules consist of a feature extractor (FE) and an object tracker (OT). The visible edges of the objects are extracted and represented through the coordinates of their endpoints and the center. The procedure adopted for feature extraction is listed very briefly.

Feature Extraction. The FE operates in much the same vein as the one described in the postprocessing module of the STA—the only difference being that it extracts the features of all the objects in the neighborhood, while in STA it extracts the features of only those objects that were classified as dynamic. Thus the FE appears in the postprocessing phase of the STA but in the preprocessing phase of the MBA.

Object Tracking. The extracted objects are tracked through their centers in the subsequent samples by a nearest-neighbor classification algorithm based on Euclidean metric. An interesting case that arises while tracking must be mentioned. Shown in Figure 12 is a case where an object that was fully visible in the previous instant gets partially occluded, due to which the

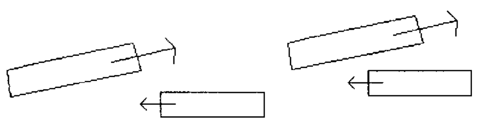

Figure 12. Between two successive scans of the environment the object gets partially occluded; this can cause a subsequent shift in the center, causing it to be classified as a new object.

shift in its center is considerable, and it can be detected as a new object in the subsequent instant. The correspondence between the apparent new object at t is done with the older object at $t-1$ by seeing whether their gradients and intercepts are compatible at both instants. Other issues—such as how the tracking algorithm maintains a list of objects tracked, identifies new objects that appear within the vicinity of the robot, and discards objects that are no longer visible—are not mentioned here, for they are not pertinent to the main theme of the discussion.

2.2.2. Object Classifier

The tracked objects are classified in two tracks. One track is relevant for objects whose extracted edges are parallel to their own motion direction, and the other for objects whose edges are perpendicular to their direction of motion. Since the direction of motion of the object with respect to its edges cannot be determined until the motion is detected, both the schemes are employed and the object is certified as dynamic if one or both the schemes detect motion in it.

Parallel Edge Motion Detection (PEMD). The PEMD scheme detects motion in an object by looking for the rear (front) end of its visible edge to get past the location that was earlier occupied by the front (rear) end within a threshold number of samples, n_{th}. This is portrayed in Figure 13(a), where the rear end of the object gets past the location occupied by the front end at t_0 in 5 samples. The threshold can be varied according to the speed and size of the object considered. For longer objects, the MBA looks for one of the endpoints to get past a certain length of the object within the required number of samples. The ratio of the length of the edge to its entire visible length, which must be crossed over by the endpoint for the object to get classified as dynamic, is obtained through the plot of the relation shown in Figure 13(b). Denoting the front coordinates of the visible edge as (x_f, y_f) and the rear coordinates as (x_r, y_r), and when the direction of motion is from the rear to the front, the following procedure classifies the attribute of the object:

Procedure MBA-PEMD

```
If $(x_f(t) > x_r(t))$ and $(y_f(t) > y_r(t))$ for an object k
    If in $\varepsilon < n_{th}$ number of samples
        If $x_r(t+\varepsilon) > x_r(t) + \frac{x_f(t) - x_r(t)}{n}$ and
           $y_r(t+\varepsilon) > y_r(t) + \frac{y_f(t) - y_r(t)}{n}$
        then k is a dynamic object
        else k is a static object.
```

Similar reasoning extends for other combinations of the front and rear edge, such as $(x_f(t) < x_r(t))$, and so on.

Perpendicular Edge Motion Detection (PDMD). The PDMD scheme detects motion in an object by considering the rate of decrease in the distances to the object's center over a time window. Consider Figure 14, where a static object is tracked in two successive samples through its center. The robot undergoes a net rotational displacement of α between the two scans. The robot actually moves along an arc from position a at t to b at $t+1$, such that the net rotational displacement with respect to the reference frame is α. The angles made by the robot's heading direction with respect

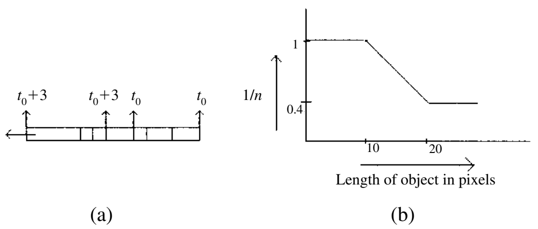

Figure 13. (a) PEMD. The rear edge of the object gets past the location of the front edge at t_0 after 3 samples at t_0+3. (b) Plot of length of object vs the ratio $1/n$.

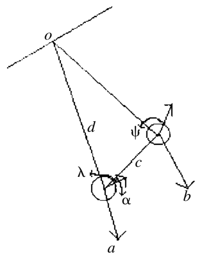

Figure 14. Static object tracked in two successive samples by its center marked as o.

to the object's center at the two instants are λ and ψ (see Fig. 14), respectively. In actual simulations and implementations there would be shifts in the object's center between the two instants. The shifts are not considered in obtaining a relation for the decrease in displacement, for what is required is a general notion of the decrease to formulate a threshold that can classify the attribute of the object. The amount of shift in the center between two successive scans is in general difficult to predict or estimate. Let the distance from the object's center, o, to the robot's center at a be d, and the displacement of the robot's center between the two samples be c. Then the expected value of the distance from the object's center to the robot's current localization at b is given by

$$\begin{aligned}\hat{d}_t &= -c\cos\psi \pm \sqrt{(d_{t-1} - c\sin\psi)(d_{t-1} + c\sin\psi)} \\ &= d_{t-1} - c\cos\psi \quad \text{for } d_{t-1} \gg c \end{aligned} \tag{1}$$

The expected rate of decrease between two samples of the environment can be written as $\hat{s}_i = \hat{d}_t - d_{t-1}$. Considering an ensemble of N such samples of the environment, and denoting the observed average rate of decrease as $\bar{s}$ and the expected average rate of decrease as $\bar{\hat{s}}$, the PDMD routine classifies the object as dynamic if the following relation holds:

$$\bar{s} > \bar{\hat{s}} + \Delta \tag{2}$$

where Δ is a threshold fixed at 1.6 pixels/sample,

$$\bar{s} = \frac{d_{t+N-1} - d_t}{N-1}$$

and

$$\bar{\hat{s}} = \frac{\hat{d}_{t+N-1} - d_t}{N-1} = \frac{\sum_{i=1}^{N-1} c_i \cos\psi_i}{N-1} \tag{3}$$

It is to be noted that Δ is *not* a problem-specific parameter, it can be considered as a kind of tolerance, given in lieu of the fact that the sensors do not detect the same locations on an object during successive instants. In other words, we may say to a reasonable certainty that if relation (2) holds, the extracted feature corresponds to a dynamic object. However, in cases where $\bar{\hat{s}} < \bar{s} < \bar{\hat{s}} + \Delta$, the extracted feature is considered to be dynamic with a certainty $p = (\bar{s} - \bar{\hat{s}})/\Delta$. If the same inequality $\bar{\hat{s}} < \bar{s} < \bar{\hat{s}} + \Delta$ holds over successive instants, the certainty values are incremented. The certainty value reaches unity in general within the subsequent three samples.

The main advantage of the MBA is its robustness to rotational displacements of the robot. It has been seen that the MBA can track objects accurately even during large rotational movements of the robot. An obvious demerit of the MBA is the necessity to build a model of the environment in every scan, update it, and track all the objects in the neighborhood irrespective of their attribute.

2.3. Clustering Based Approach I (CBA-I)

In the MBA, the feature vectors employed for feature extraction were those obtained during the most recent sample. In both CBA schemes, however, the feature vectors obtained during the recent five samples of the environment are used for demarcating the objects. The advantages are that the objects are more easily demarcated as the density of feature vectors populating a cluster increases, for the object has been repeatedly scanned. This facilitates more accurate partitioning of the data into clusters. The differences are highlighted in Figures 15(b) and (c). Figure 15(b) shows the point cloud of the workspace of Figure 15(a) as seen by the sensors in one sample, while Figure 15(c) is the accumulated point cloud over the last 5 samples and the clusters are more discernible than in Figure 15(b). The preprocessing stage consists of partitioning the point cloud of feature vectors (FV), where each vector is a triplet $[x, y, t]$, into clusters representing the objects that own the vectors. For clustering, the time component of the vector is not considered.

2.3.1. Preprocessing Module

Before describing the clustering algorithm, some of its salient features are listed:

- The algorithm is self-organizing and determines the number of clusters.

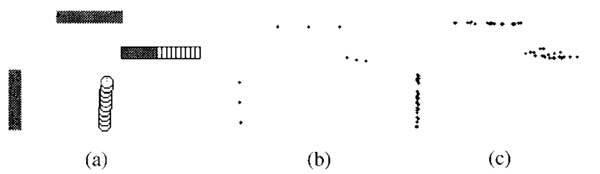

Figure 15. (a) Robot scanning an environment with two static and one dynamic object. (b) Features extracted based on the last scan alone. (c) Features extracted over the last five samples.

- It makes use of two distance measures, center to feature vector distance (CVD) and vector to vector distance (VVD), whose thresholds are denoted as c_{th} and v_{th} respectively.
- The thresholds decide the formation of a new cluster. The thresholds are adaptive, however, and change according to the distribution of the feature vectors.
- The VVD uses the standard Euclidean metric while the CVD employs the Mahanolobis distance.
- The partitioning is achieved in one shot, in the sense that the feature vectors must be presented only once to the algorithm.

The algorithm can be considered novel in the sense that an exactly similar algorithm does not seem to appear in the literature, but our aim here is not to stake claims for its novelty or originality. Certain features make it suitable for the problem under consideration in the context of real-time learning, namely, the ability to determine the number of clusters, and one-shot partitioning. The contribution of the two distance measures and the adaptive thresholds adds to the reliability of the algorithm, in finding the number of objects in the vicinity and their centers to sufficient accuracy.

The basic clustering algorithm follows.

Inputs. Feature vectors $S = \{X_1, X_2, \dots, X_N\}$, where each $X_i = \{x_i, y_i\}$; initial threshold values for c_{th} and v_{th}, forgetting factor β and boundary parameter η. In the algorithm given, parameter $\alpha = 1 - \beta$.

Outputs. number of clusters K, each cluster designated as $C_i, i = \{0, 1, \dots, K-1\}$; number of feature vectors $n(k)$ in a cluster $k = \{0, 1, \dots, K-1\}$; cluster centers $z_k = \{x_{ck}, y_{ck}\}$; covariance matrix for each cluster; eigenvalues and eigenvectors for each cluster.

Procedure CBA1_CLUS

```
1. Select an arbitrary feature vector, $X_i$, assign it to cluster $C_0$, or equivalently cluster $(X_i) \leftarrow C_0$, $z_0 = X_i$. Tag $X_i$ as considered.
2. Find the distance of the nearest, *unconsidered* feature vector (FV) $X_j$ to the currently considered FV $X_i$. Denote this distance as vvd. $vvd = \min_{j \neq 1} d_e(X_i, X_j),\ j = \{0, 1, \dots, i-1, i+1, \dots, N\}$ where $d_e$ is the Euclidean norm.
3. *If* $vvd \leq v_{th}$
   a. cluster $(X_j) \leftarrow C_{cur}$, where, $C_{cur}$ = cluster $(X_i)$
   b. $n(C_{cur}) = n(C_{cur}) + 1$; increment the number of vectors in a class
   c. Update center and covariance matrices for the cluster
   d. Update VVD threshold as $v_{th} = \alpha(v_{th}) + (1-\alpha)vvd$
   e. Find $cvd = d_m(X_j, z_{cur})$, where $d_m$ stands for the Mahanalobis norm
   f. If $cvd > c_{th} - \eta$ then $c_{th} = cvd + \eta$; updating the CVD threshold
4. *If* $vvd > v_{th}$
   a. Find $z_k = \min_i d_m(X_j, z_i),\ i = \{0, 1, \dots, K\}$, the nearest center of a cluster to $X_j$
   b. $cvd = d_m(X_j, z_k)$
   c. *If* $cvd \leq c_{th}$
      i. cluster $(X_j) \leftarrow C_k$, $C_k$ = cluster $(z_k)$
      ii. $n(C_k) = n(C_k) + 1$
      iii. Update centers, covariance matrices for $C_k$
   d. *Else if* $cvd > c_{th}$
      i. $K = K + 1$; forms a new cluster
      ii. $z_K = X_j$; the center of the new cluster takes the value of the FV $X_j$
      iii. $n(K) = 1$; number of FV in the new cluster is initialized to 1
      iv. Set the distance thresholds to their starting values
5. Tag $X_j$ as considered and assign $X_i = X_j$ so that $X_j$ becomes the FV over which steps 2–4 are applied for the next pass of the algorithm
6. Repeat steps 3–5 until all the feature vectors have been tagged as considered.
7. Compute eigenvalues and eigenvectors for the clusters that got formed during the partition process.
```

The pivot of the algorithm is the vector to vector distance measure, denoted as *vvd* in the algorithm. Starting from an arbitrary FV X_i owned by an object O_k in the range space, the algorithm attempts a search to extract the remaining feature vectors of O_k before branching to the FV of another object.

The algorithm can be best understood through Figures 16(a) and (b), which are the point cloud representation of a certain environment. The arrows in the figure indicate the direction in which the search proceeded. S in the figures denotes the starting feature vector considered (step 1 of the algorithm). The initial threshold, v_{th}, is generally of a higher value. The algorithm searches for the nearest *unconsidered* vector to S (step 3). If the nearest vector is within the threshold distance to the vector considered, it

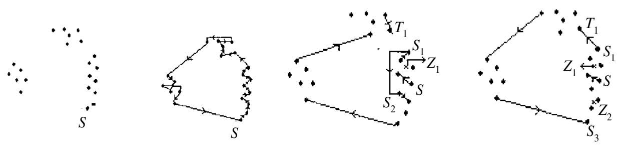

Figure 16. (a) The point cloud representation of an environment is shown on the left. The right figure represents the direction of the search starting at S. (b) The search branches from FV S_1 to S_2 of the same cluster. When the search reaches S_1 the center has shifted to Z_1. Center Z_1 is shown as a cross. (c) The search proceeds to T_1 from S_1 and reenters at S_3. The centers of the split cluster Z_1, Z_2 are shown using a cross.

gets added as an FV of the same object; the center, variances, and thresholds for that object get updated (steps 4a–f). The newly added FV becomes the next FV to be considered, to which step 3 is again applied. In this way the algorithm proceeds to extract the vectors of the same object. The algorithm exploits the principle that the vectors owned by the same object should be spatially more proximal amongst themselves than the vectors belonging to another object. Simultaneously the threshold v_{th} shrinks and settles at a value indicative of the average intervector partition within a cluster. Figure 17(a) shows the typical plot of v_{th} for a cluster as the search proceeds. The forgetting factor $\beta = 1 - \alpha$ is responsible for adapting v_{th} to the intervector partition by this factor while retaining the previous threshold value by a factor $\alpha = 1 - \beta$.

When the nearest FV to X_i, X_j, does not fall within the threshold v_{th}, the algorithm finds the center of a cluster that is closest to X_j Mahanalobically (step 5a, where d_m is the Mahanalobis norm). If the closest center does not satisfy the threshold criterion of step 5c, a new cluster is formed with its center and thresholds initialized according to steps 5d(i–iv). Step 5c is crucial for those situations when the initial FV to be considered was somewhere near the center of the object, like the FV S in Figure 16(b). When the search

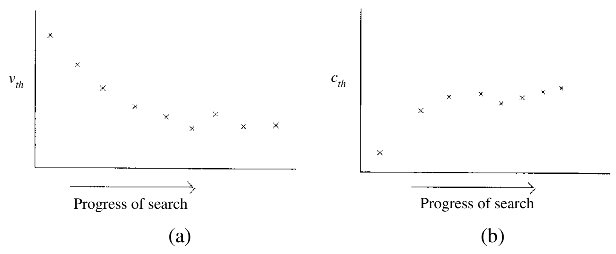

Figure 17. (a) Typical variation of threshold v_{th} for a cluster. (b) Typical variation of threshold c_{th} for a cluster.

reaches an end of the cluster with FV S_1 (Fig. 16(b)) the closest *unconsidered* FV to S_1, which is S_2, shall fail the criterion of step 4. However, since it lies within the threshold c_{th} with respect to center Z_1 (Fig. 16(b)) of the same cluster at that instant, it (S_2) gets assigned as an FV owned by the same object that owns S and S_1. The search proceeds from S_2. The arrows in Figure 16(b) also indicate the branching of the search from S_1 to S_2. While v_{th} is adapted to capture the average intervector partition distance within a cluster, c_{th} is adapted to delineate the boundary of the cluster and to grab those vectors that do not satisfy criterion of step 4 but lie near the periphery of the cluster. The c_{th} adapts itself to trace the boundary of the cluster when the Mahanalobis norm is employed. The typical variation of c_{th} for a cluster with the progress of search is shown in Figure 17(b). The boundary parameter n of step 4f is instrumental in achieving this adaptation.

On the other hand, if T_1 becomes the closest FV with respect to S_1 of Figure 16(b), then T_1 initiates the formation of a new cluster as it fails the criterion of steps 4 and 5c. In this case the search progresses as shown in Figure 16(c) and the starting cluster is again reentered through the feature vector S_3. If S_3 passes the criterion of step 5c, then it gets assigned to the object that owns S and S_1. If it fails, it splits the object into two clusters with centers Z_1 and Z_2, indicated by crosses in Figure 16(c). Such split clusters are merged through a compatible cluster-merging scheme discussed very briefly in the subsequent subsection.

2.3.2. Object Classification

The eigenvalues and the eigenvectors of the partitioned clusters are computed. A cluster representative of a dynamic object can be discerned from those signifying a static object through its elliptical or rectangular shape. Static clusters are linear and enclose negligible area while the visible edges of dynamic objects tracked over samples enclose a finite area.

The objects are classified according to a simple procedure as follows:

```
Procedure CBA-I_CLAS
for $i = 1$ to $K$
    if $(\sqrt{\lambda_{2i}}/\sqrt{\lambda_{1i}}) > \kappa$) then $C_i$ represents a dynamic object
    else it indicates a static object
end;
```

Here λ_{1i} is the eigenvalue corresponding to the first principal component and κ is set to 0.3. The ratio is the ratio of the standard deviations along the principal axes of the cluster. The ratio indicates the shape of the cluster, and for linear clusters or lines the ratio is closer to zero.

Prior to object classification, a compatible cluster merging (CCM) method needs to be invoked. CCM is used for merging two categories of decomposed clusters. The first category is the split cluster described towards the end of Section 2.3.1. The other category refers to two or more parallel line like clusters representing the same dynamic object. This occurs when the sensors detect more or less the same locations on a moving object in successive samples. In such a case the dynamic object appears to be composed of two or more linear clusters that are more or less parallel. The CBA-I may decompose such an object into its linear clusters that need to be merged. The CCM algorithm does this. The algorithm is ommitted here for brevity. The criteria for CCM are similar to those employed by Krishnapuram[15] with certain modifications, and the clusters are merged pair-wise transitively.[16]

Compatible Cluster Merging. It is the nature of most of the clustering algorithms that they do not partition the data into the expected number of clusters unless the number of clusters is specified beforehand. This is especially so if the algorithm has to find the number of clusters on its own from the given data. It has been found with the problem at hand that two categories of decomposed cluster tend to get formed, that need to be merged. One category is the split cluster discussed previously and the other category is the decomposition of a dynamic object into two or more parallel line like clusters.

The split clusters are merged if the following conditions hold:

1. The closest interpixel distance between any two feature vectors owned by the individual clusters is comparable with the steady state threshold v_{th} obtained for each of the two clusters.
2. The eigenvectors corresponding to the smallest eigenvalue for either of the clusters under consideration are nearly parallel.

The decomposed clusters of a dynamic object get merged based on conditions suggested by Krishnapuram and Freg.[15] Let the centers of the two clusters be ν_i and ν_j, the eigenvalues of the two clusters be $\{\lambda_{i1}, \dots, \lambda_{in}\}$ and $\{\lambda_{j1}, \dots, \lambda_{jn}\}$, and the normalized eigenvectors be $\{\phi_{i1}, \dots, \phi_{in}\}$ and $\{\phi_{j1}, \dots, \phi_{jn}\}$. The eigenvalues and eigenvectors are arranged in descending order of the eigenvalues. The criteria proposed are stated as follows:

1. $$\frac{\|\nu_i - \nu_j\|}{\sqrt{\lambda_i} + \sqrt{\lambda_j}} \le k_3, \quad k_3 \text{ lies between 2 and 4} \tag{4}$$
2. $$\left|\phi_{in}^T \phi_{jn} \ge k_1\right|, \quad k_1 \text{ close to 1} \tag{5}$$
3. $$\frac{\phi_{in}^T + \phi_{jn}^T}{2} \frac{\nu_i - \nu_j}{\|\nu_i + \nu_j\|} \le k_2 \tag{6}$$

The first condition states that the cluster centers should be sufficiently close, and the second states that the clusters should be almost parallel. The third requires that the normal to the hyperplanes be orthogonal to the line connecting the two centers. In other words, the clusters should lie in the same hyperplane. The third condition is redundant for the current application, as all the feature vectors are represented on the same Cartesian plane. Though there exist more sophisticated merging criteria,[16] this method has been adopted (considering their simplicity as well as their suitability for the problem at hand). The compatible clusters are merged transitively in pairwise fashion.[16]

2.3.3. Parallel Edge Motion Detection

The preceding classification scheme works well for dynamic objects whose visible edges are perpendicular to their own direction of motion. Objects whose visible edges are parallel to their motion direction get represented as line-like clusters, as was the case with static objects. For such objects the time component of the FV is considered explicitly and motion is detected in the same manner as described through procedure MBA-PEMD.

2.4. Clustering Based Approach II (CBA-II)

Finally, we present the GK algorithm as a possible clustering approach for dynamic object perception. The GK algorithm requires the number of clusters to be specified initially. Since the number of objects in the robot's vicinity is unknown, the number of clusters, K, is assigned a value higher than the number of objects the robot generally encounters over five sampling instants. The data obtained is partitioned into clusters through the GK algorithm and compatible clusters are merged through the CCM scheme mentioned in Section 2.3.3. Prior to this is another processing step, the reassignment of wrongly assigned clusters to feature vectors. It is one of the inherent traits of the fuzzy clustering family of algorithms that smaller clusters tend to grab FVs owned by larger clusters to minimize the cost function.[17] Hence a module that reassigns the wrongly assigned vectors to their actual owner clusters is introduced. This module identifies an FV, X_i, to be wrongly assigned to an object O_i, when its closest FV belongs to a different object O_j and X_i is also Mahanalobically closer to the center of the object O_j than the object to which it is actually assigned (O_i). After reassignment, compatible clusters are merged and the final clusters classified in the same manner as described in Sections 2.3.2–3.

While it is accepted that the CBA-II differs from the CBA-I only in the preprocessing stage of cluster formation, the main motivation for employing this approach has been to present a possible application of the fuzzy clustering family of algorithms[18] in the area of mobile robot navigation. Literature indicating such applications has been very rare. Apart from that, the nature of the sensor data seem amenable for a clustering application. Fuzzy clustering algorithms have an advantage in that the data is partitioned based on an explicit minimization of an optimization function and the final convergence is independent of the order in which the inputs are presented. The parameters need not be varied from one data set to another. The GK algorithms seem specifically suited for this purpose due to their ability to detect elliptical and linear clusters of different shape and orientation. The constraint to specify the initial number of clusters can be compensated by clustering with a higher number than required and coalescing the compatible ones. A better approach to determining the optimal partition of data can be the one given by Xie and Beni.[19] However, this algorithm is unsuitable for real-time applications, as it requires clustering over a range of possible partitions starting from $K = 2$ before reaching a value of K that indicates optimal partition. A comparative summary of the four approaches is presented towards the end of the subsequent section, with respect to criteria critical for robotic navigation.

3. SIMULATION RESULTS AND COMPARISON

To test the efficacy of the algorithm, a graphical simulator has been developed on a Pentium machine. The robot is modeled as a circle of radius 5 pixels with a ring of 24 sensors placed along the circumference, with an angular separation of 15 degrees with respect to the robot's center. In simulations, the trajectory of the robot until it identifies the dynamic object is shown. To portray the efficacy of the classification strategy for further application, simulation graphs of how this classification actuates collision avoidance of dynamic objects is occasionally shown.

3.1. Simulation Analysis

Figures 18–20 depict identification of dynamic objects through the STA. The weight vectors of the resilient propagation (RPROP) network obtained after the training process converged are used while navigating in a real-time environment. Input data is preprocessed by the spatial classifier followed by the

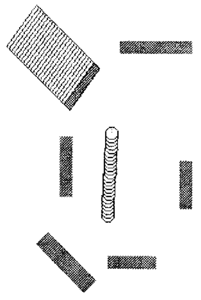
Figure 18. Instant of perception of the dynamic object on the left through sensors 13 and 14 with VOCs [3, 3] and [2, 3], respectively.

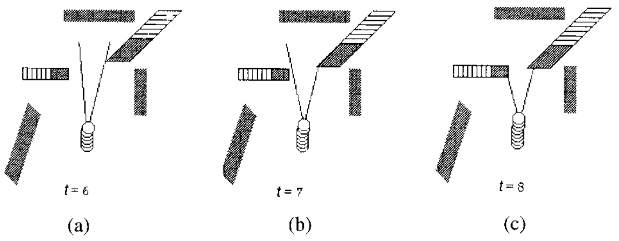

Figure 19. (a) Instant prior to cognition of the dynamic object on the right. (b) Sensor 10 cognizes the dynamic object on the right due to VOC [7, 0]. (c) Sensor 12 cognizes the dynamic object on its left through VOC [8, 0].

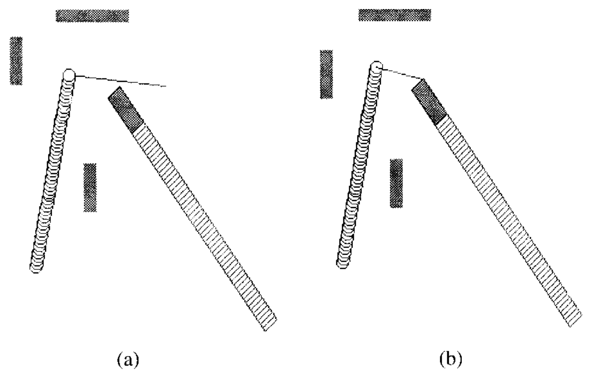

Figure 20. (a) Detection of free space by sensor 5 an instant prior to motion detection. (b) Sensor 5 perceives motion in the environment through the VOC [0, 0].

order extractor before being fed to the RPROP network. The network outputs the temporal sequence of classes as an experience of static or dynamic objects.

Figure 18 shows the instant of perception of the dynamic object on the robot's left amidst five static objects through sensors 13 and 14. The RPROP network classifies the VOC [3, 3] obtained through sensor 13 and VOC [2, 3] obtained through sensor 14 as experiences of a dynamic object. The postprocessing module compares the range readings of both the sensors, and finding them to be comparable, infers that the sensors have cognized the same dynamic object. Figures 19(a–c) portray the perception of two dynamic objects amidst three static ones. Figure 19(a) shows the instant just prior to perception. Figure 19(b) signifies the situation when the dynamic object on the right is perceived through sensor 10, and Figure 19(c) is when the dynamic object on the left is detected by sensor 12. The average velocity of the robot was 2.7 pixels per sample. The experience of the VOC by the sensors that perceived motion is also given in the captions to the figures. Figure 20(a) indicates the situation just prior to motion detection, when sensor 5 detects free space. Figure 20(b) is the instant when motion is perceived through sensor 5 as it detects an object suddenly at near ranges. This experience gets captured through VOC [0, 0] for sensor 5, which indicates an experience of a dynamic object. The postprocessing module checks for the reading of sensor 6 that is adjacent to 5 but closer to the center sensor 11. Since the readings are not comparable, the STA classifies the experience of the spatiotemporal patterns by sensor 5 as an experience of a dynamic object.

Figure 21 is a snapshot of the instant when the MBA through the PDMD track detects the dynamic object on the rear while the robot rotates to avoid the static object. This we feel to be an involved case for detection, for the object is on the rear and approaches the robot at an obscure angle that is likely to go unnoticed. Figures 22(a–c) depict an environment that contains three dynamic and two stationary objects. The MBA could clearly distinguish the static and dynamic objects. The figures also show the instances when the objects labeled 1, 2, and 3 got identified as dynamic. Here objects 1 and 2 got identified through the MBA-PEMD scheme and object 3 through the MBA-PDMD method.

Figures 23–25 delineate the classification of dynamic objects through CBA-I. Figure 23(b) is the point

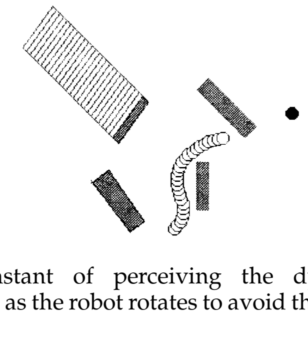
Figure 21. Instant of perceiving the dynamic object through MBA as the robot rotates to avoid the static objects.

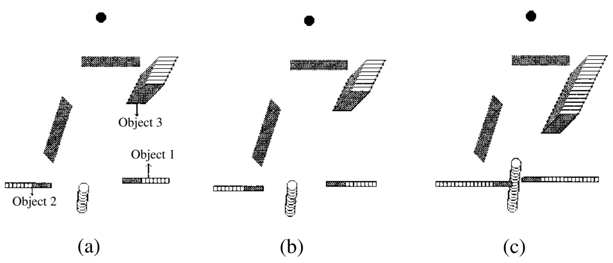

Figure 22. (a) Instant of perceiving object 1 by MBA. (b) Perception of object 2. (c) Perception of object 3.

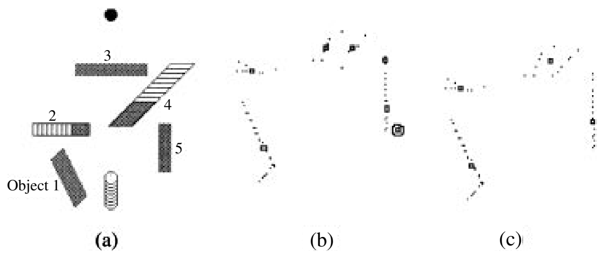

Figure 23. (a) Three static and two dynamic objects. (b) The point cloud representation of part (a) with the centers of clusters formed by CBA-I. A outlier cluster beside object 5 is shown circled. (c) Cluster centers after CCM.

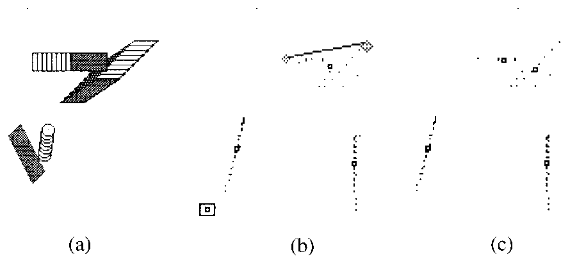

Figure 24. (a) An environment where one dynamic object crisscrosses the path of another within a few samples. (b) Clustering results of CBA. The FV of the merged cluster of the two dynamic objects for $t = 8$ are disparate and shown connected by a line. A noisy cluster is also shown within a box. (c) The final clusters and their centers. The coalesced dynamic object gets decoupled into two.

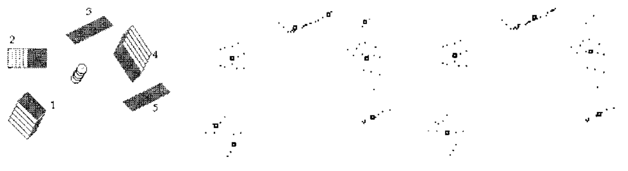

Figure 25. (a) Navigation amidst three dynamic and two static objects labelled 1–5 in clockwise direction. (b) Initial clustering results. Dynamic objects 1 and 4 are clustered as two clusters each by the CBA-I. Static object 3 forms a split cluster. (c) Final cluster centers after CCM. Objects 1, 3, and 4 are represented by a single cluster now.

Table II. Ratios of the standard deviations of the objects of Figure 23(a) along the principal axes of the clusters.

Object Index	$\sqrt{\lambda_{i1}}$	$\sqrt{\lambda_{i2}}$	$\sqrt{\lambda_{i1}}/\sqrt{\lambda_{i2}}$
1	12.11	2.24	0.18
2	6.08	1.09	0.18
4	10.00	3.90	0.39
5	12.42	1.25	0.10

cloud representation of the environment of Figure 23(a), based on the data acquired during the last five samples. The figure also shows the centers of the clusters denoted by small circles or squares found by the CBA-I algorithm. The static object on the robot's right (object 5) gets represented as a split cluster (described in Section 2.3.2) while the dynamic object on the robot's front-right (object 4) is decomposed into two linear clusters. The prominent advantage of this approach is that noisy readings and outliers get identified as individual noise clusters that can be easily discarded. This is especially crucial from the point of view of the collision-avoidance scheme that would follow the classification. Since collision avoidance involves some kind of future prediction of the object's motion based on its endpoints or centers, presence of outliers can distort the centers or endpoints from their actual locations and result in prediction errors that can affect the collision-avoidance strategy. In Figure 23(b) the outlier beside the cluster indicating object 5, shown enclosed in a circle, gets identified as a distinct noise cluster. These clusters can then be quarantined from affecting the estimates of the motion predictor that would follow. Figure 23(c) shows the centers of the clusters after CCM. The dynamic object now gets represented as a rectangular/elliptical cluster, the split cluster on the right gets merged, and the outlier discarded. The classification strategy identified both the dynamic objects, the one on the front-right (4) through the ratios of the eigenvalues and the one on the front left (2) through the same scheme as MBA-PEMD. Table II shows the ratios of the square roots of the eigenvalues of the clusters of Figure 23(c). The labeling of the objects in Table II is consistent with the labeling indicated in Figure 23(a).

Figure 24(a) depicts another interesting situation, where one dynamic object crisscrosses the path of another in quick succession. As a result the point cloud representation of both the dynamic objects (Fig. 24(b)) are merged, and the CBA-I clusters both the objects together as a single entity. Such merged clusters get differentiated through a dynamic decoupling procedure. This procedure makes use of the time component of the feature vectors to separate the clusters. In a nutshell, the FV of the cluster at a particular timestamp is considered. For example, considering the FV of the merged cluster with timestamp $t = 8$ shows two distinct clusters marked with circles and connected by a line in Figure 24(b). Similarly, two distinct clusters get formed at $t = 9$ and $t = 11$. If over a sequence of five samples a cluster can be subdivided in more than two samples, the algorithm decouples the cluster, as follows. The distinct clusters that got formed at successive time stamps are clubbed through a nearest-neighbor criterion. In other words, a distinct cluster at $t = 8$ gets merged with the closest distinct cluster at $t = 9$. Thus we have two distinct sets of clusters considering the timestamps of $t = 8$ and $t = 9$ from the four that got formed (two each for each timestamp). For each of the distinct sets the eigenvectors are computed. The remaining FV gets assigned to that cluster to which it is Mahanalobically closer. Thus the coupled cluster of Figure 24(b) gets decoupled in Figure 24(c) into two. The preprocessed clusters were then classified accurately as two dynamic and two static clusters. A noisy cluster shown enclosed within a square in Figure 24(b) also gets discarded in Figure 24(c).

Figure 25(a) shows navigation amidst two static and three dynamic objects. The CBA-I clusters the five objects into eight clusters. The clusters along with the centers are shown in Figure 25(b) with the dynamic objects labeled 1 and 4 getting decomposed into two clusters each. Figure 25(c) represents the centers after CCM. Once again the classification strategy could accurately identify the dynamic objects amidst static ones. Here the eigenvalue method mentioned in Section 2.3.2 classified all three dynamic objects.

Finally, indicated in Figure 26(a) is the clustering of the environment of Figure 25(a) through the GK algorithm. The initial number of clusters had been fixed to 8. Also shown is an FV indicated by an arrow that

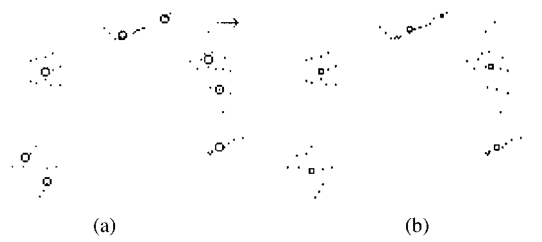

Figure 26. (a) GK algorithm forms eight clusters in much the same way as CBA-I except for a misassigned FV indicated by an arrow. (b) Final centers after CCM.

is owned by object 4 (Fig. 25(a)) but gets wrongly assigned to object 3. Figure 26(b) shows the centers after sorting out the wrongly assigned FV, followed by CCM.

As a possible illustration of the efficacy of the detection schemes for the purpose of real-time collision avoidance of dynamic objects, the graphs of Figure 27 show the various stages in collision avoidance of two dynamic objects encountered in quick succession. The objects got classified as dynamic through the MBA.

3.2. Comparative Analysis

The four approaches are compared on the basis of their suitability for real-time implementation, accuracy of classification, nature of the internal representation required, and performance during rotational displacements of the robot. The following limitations are common across all four approaches, some of which can be made out from the simulation graphs.

1. The approaches can discern an object as stationary or dynamic based on its translational displacements. The approaches are not designed to identify rotating objects as dynamic (such as a rod that rotates about its center).
2. The length of the object can also affect the performance of the classification. This happens when the detection is based on the visible edge that is parallel to the object's own motion direction and is very long. Here "very long" can be interpreted as those edges whose length spans the conical envelope of all those sensors that could have detected the object over more than half the time window number of samples. Such very long edges would get identified as stationary though dynamic. For example, an edge that can be detected at a particular instant by sensors 10 to 13 and is detected by all of them and continues to be detected by all of the sensors that can possibly detect it for two more samples is considered very long.
3. The distance moved by the robot during a scan of the environment is negligible when compared with the distance moved by it between successive scans of an environment. In other words the algorithm treats the robot to be stationary when the sensors probe the environment. This does not necessarily mean that during real time the robot must stop every time it evaluates its neighborhood. The assumption is only to facilitate simplified computations.

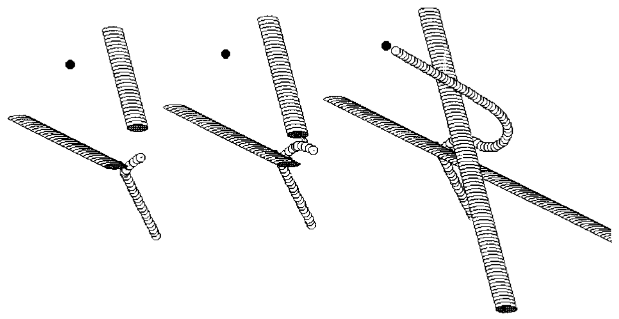

Figure 27. Various stages in detection, tracking and avoidance of two dynamic objects. Detection through MBA.

3.2.1. Analysis of STA

The STA appears to be the best suited for real-time implementations, as it can classify the objects faster than the remaining approaches. The only preprocessing is the extraction of the temporal order of classes. Upon being presented with the temporal order, the network outputs the classification through the weight vectors learned offline by a single feed-forward pass. It also does not require any internal representation in terms of feature maps of the objects, and classifies purely based on range data. The scheme is however susceptible to rotation. Though its variance to rotation is tolerated through the extrapolation scheme mentioned in Section 2.2, it is unreliable when the robot has to encounter dynamic objects in succession. In the absence of successive rotations, the scheme gives the best accuracy amongst all the approaches. This can be attributed to the use of artificial neural networks, which is especially useful for nonlinear mappings as well as noisy data. Further, the architecture employed (Figs. 2, 8) is inherently elegant and can be realized through hardware.

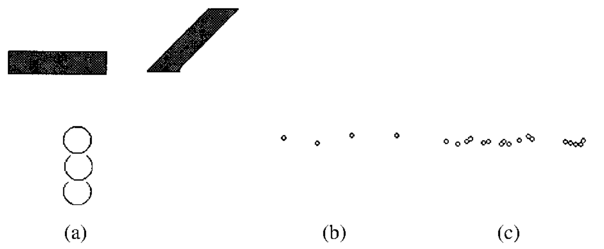

Figure 28. (a) A difficult case for discernment for the MBA. (b) FV extracted from one sample. (c) Feature vectors extracted over last five scans.

3.2.2. Analysis of MBA

In terms of real-time suitability, the MBA should rank second after the STA, as features are extracted through a single pass of the feature vectors obtained from the most recent scan. It does not involve an iterative procedure for clustering the partitioned data as in CBA-II or a search for the nearest FV as in CBA-I. Storage requirements are lower when compared with the clustering approaches, since data acquired from the last sample alone is considered. The algorithm is invariant to rotational displacements of the robot. However, in terms of accuracy of classification, it is the least accurate amongst all the approaches. The MBA is specifically vulnerable in situations such as that illustrated in Figure 28(a). Since the visible edges of both objects are along the same line and the objects are spatially proximal, the MBA may not be able to demarcate the two objects based on the FV extracted (shown in Fig. 28(b)).

3.2.3. Analysis of CBA-I

In terms of real-time suitability, CBA-I ranks third. Though the clusters are partitioned through a single pass of the feature vectors in the sense that iterative loops are not required, the partitioning process entails a search for the nearest FV for every FV considered. However the search space reduces with every FV considered, as only the remaining unconsidered FVs are searched. The scheme also requires a CCM procedure. In terms of accuracy, the scheme is most reliable when rotational displacements are considered. The scheme is also robust to presence of outliers. Outliers in general get partitioned as distinct noisy clusters that can be discarded. With FV accumulated over the last five samples and an intrinsic ability to detect free space through the threshold v_{th}, it can easily demarcate objects as shown in Figure 28(a), due to the presence of a discernible free space between the FVs of the two objects (Fig. 28(c)). It is invariant to rotation.

3.2.4. Analysis of CBA-II

This scheme ranks last in terms of real-time suitability amongst the approaches considered here. This is chiefly due to an iterative procedure for partitioning the clusters typical of a clustering algorithm based on optimizing an error function,[11,18] coupled with other procedures such as CCM. Nevertheless, for the kind of objects encountered in a typical indoor environment where the number of objects encountered over a span of a few samples is not very high, the performance seems reasonable for real-time application when simulated on high-speed processors. The accuracy is comparable with the CBA-I, however the tendency to grab FVs belonging to another cluster is prominent. The algorithm is rotation invariant.

3.3. Possible Objections and Limitations

It is acknowledged that the results presented here are graphical simulations and the utility of the approach for real-world implementations may be questioned. The prominent objections can be from the point of view of estimation of the robot's egomotion and the reliability of sensor data. It is argued here that the above methods can be employed legitimately for experimental robots also from the point of view of both these objections.

The range data provided by the range finders are with respect to a frame of reference fixed to the sensor. To compare range data obtained at different instants they need to be projected onto the same reference frame to facilitate maintaining internal representations in the form of maps. This entails an accurate knowledge of the robot's displacement in terms of rotational and translational motions between samples. The approaches presented in this paper are primarily concerned with identifying local spatial changes that occur within a short time window of observations and not with changes over temporally distant scans. Hence long-range position estimation is not the consequence of this paper. The changes in the robot's position over temporally proximal scans can be estimated to working accuracy based on odometric data itself. Recently Prassler and others have reported efficient real-time implementations on a similar problem through map building based on such odometric data.[20] It is also emphasized that it is not the objective of this paper to concern itself with robot localization or the map building problem, and the argument is that the present approach should be used in tandem with approaches that surmount dead-reckoning errors.

As far as range data is concerned, laser range finders have proven to be more precise and reliable than sonar. In this paper we have assumed that specular reflections do not arise from dynamic objects and this assumption has been condoned in other approaches.[9,10] The authors reported successful real-time implementations of dynamic collision avoidance through an REBP network, which was trained based on range data obtained from a sensor model that neglected these reflections. Another important consideration with range images is the ignorance regarding which part of the conical beam actually detected the object. Due to this, projections of the range data onto the reference frame will be in an average error of five degrees. Estimating which edge of the cone could have detected the object by considering readings obtained from adjacent sensors can reduce this error. For example, consider a contiguous set of readings obtained by sensors 7–10 detecting an object (Fig. 29). These readings can vary in three ways. They can monotonically increase from 7–10, which indicates that the object is closest to 7, or decrease monotonically, which indicates that it is closest to 10, or they can decrease and increase, which indicates that they are closest to a sensor between 7 and 10. For the monotonically increasing case, we consider the reading of sensors 10–8 to be given by that edge of the conical beam that is closest to sensor 7. These edges are shown in dark lines in Figure 29, where each sensor beam is marked by three rays, the center ray and the two extremities of the cone. However, for sensor 7 we consider the reading to have been obtained by the center ray, since it is not possible to certify whether the object came under the influence of the entire cone of sensor 7 or part of it. Hence the reading of sensor 7 is assumed to have been obtained through its central ray to keep the error bound within 5 degrees. Similar reasoning can be applied for the decreasing and increasing-decreasing patterns. In this way the errors for other sensors except sensor 7 would be negligible.

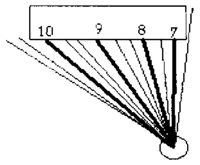

Figure 29. Sensor 7–10 detect the object with increasing ranges. That ray of the cone that is most likely to have detected the object is shown in a dark line. Each sensor is marked by three rays, a center ray and the two extremeties of the cone.

4. CONCLUDING REMARKS

To be useful in the real world, a mobile robot should be able to navigate in unstructured environments where changes are likely to occur. Though there exist many approaches in the literature that deal with navigation in nonstationary workspaces, they do not specify a strategy for demarcating dynamic objects amidst static ones. Awareness of the surroundings in terms of static and moving agents is a realistic first step for any navigating system. This occurs inherently in real-life systems. This paper has presented four possible approaches for classifying objects in a robot's vicinity as static or dynamic. These approaches have been tested through real-time simulations and found suitable. A comparative analysis of these approaches with criteria critical to real-time implementations has been presented, and it is also argued that these approaches can be adopted for indoor mobile robotic applications without much modification.

REFERENCES

1. K. Fujimura and H. Samet, A hierarchical strategy for path planning among moving objects, IEEE Trans Robot Automat 5:(1) (1989), 61–69.
2. C.L. Shih, T. Lee, and W.A. Gruver, A unified approach for robot motion planning with moving polyhedral objects, IEEE Trans Syst Man, Cybern 20:(4) (1990), 903–915.
3. N.C. Griswold and J. Eem, Control for mobile robot in presence of moving objects, IEEE Trans Robot Automat 6:(2) (1990), 263–268.
4. Y.S. Nam, B.H. Lee, and M.S. Kim, View time based moving object avoidance using stochastic prediction of object motion, Proc IEEE Int Conf Robot Automat, Minneapolis, MN, 1996, pp. 1081–1086.
5. Q. Zhu, Hidden Markov model for dynamic object avoidance of mobile robot navigation, IEEE Trans Robot Automat 7:(3) (1991), 390–397.
6. P. Srivastava, S. Satish, and P. Mitra, A distributed fuzzy logic based *n*-Body collision avoidance system, Proc Fourth International Symposium on Intelligent Rob Systems, Bangalore, India 1998, pp. 166–172.
7. I. Hiraga et al., An acquisition of operator's rules for collision avoidance using fuzzy neural networks, IEEE Trans Fuzzy Syst 3:(3) (1995), 280–287.
8. A. Fujimori, M. Teramoto, P.N. Nikiforuk, and M.M. Gupta, Cooperative collision avoidance between multiple mobile robots, J Robot Syst 17:(7) (2000), 347–363.
9. C.C. Chang and K.T. Song, Environment prediction for a mobile robot in a dynamic environment, IEEE Trans on Robot Automat 13:(6) (1997).
10. K.T. Song and C.C. Chang, Reactive navigation in dynamic environment using a multisensor predictor, IEEE Trans Syst, Man, and Cybern 29:(6) (1999).

11. D. Gustafson and W. Kessel, Fuzzy clustering with a fuzzy covariance matrix, Proc IEEE CDC, San Diego, CA, 1979, pp. 761–766.
12. K.M. Krishna and P.K. Kalra, Solving the local minima problem for a mobile robot by classification of spatio-temporal sensory sequences, J Robot Syst 17:(10) (2000), 549–564.
13. M. Riedmiller and H. Braun, A direct adaptive method for faster back propagation learning: The RPROP algorithm, Proc IEEE Int Conf on Neural Networks, 1993.
14. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning internal representations by error propagation, Parallel distributed processing, vol 1, D.E. Rumelhart and J.L. McClelland (Editors). MIT Press, Cambridge, MA, 1986.
15. R. Krishnapuram and C.P. Frieg, Fitting an unknown number of lines and planes to image data through compatible cluster merging, Pattern Recognition 25:(4) (1992), 385–400.
16. R. Babuska, Fuzzy Modeling for Control, Kluwer Academic Publishers, Boston, 1998, p. 101.
17. R. Krishnapuram and J. Kim, A note on the Gustafson-Kessel and adaptive fuzzy clustering algorithms, IEEE Trans on Fuzzy Syst 7:(4) (1999), 453–461.
18. J. C. Bezdeck, Pattern recognition with fuzzy objective function algorithms. New York: Plenum, 1981.
19. X.L. Xie and G. Beni, A validity measure for fuzzy clustering, IEEE PAMI 13:(8) (1991), 841–847.
20. E. Prassler, J. Scholz, and A. Elfes, Tracking multiple moving objects for real-time robot navigation, Autonomous Robots 8:(2) (2000), 105–116.