
Journal of Intelligent and Robotic Systems **33:** 371–408, 2002.

Detection, Tracking and Avoidance of Multiple Dynamic Objects

K. MADHAVA KRISHNA and PREM K. KALRA
Department of Electrical Engineering, Indian Institute of Technology at Kanpur, India
E-mail: {*kkrishna;kalra*}*@iitk.ac.in*

(Received: 13 November 2000; in final form: 10 July 2001)

Abstract. Real-time motion planning in an unknown environment involves collision avoidance of static as well as moving agents. Strategies suitable for navigation in a stationary environment cannot be translated as strategies per se for dynamic environments. In a purely stationary environment all that the sensor can detect can only be a static object is assumed implicitly. In a mixed environment such an assumption is no longer valid. For efficient collision avoidance identification of the attribute of the detected object as static or dynamic is probably inevitable. Presented here are two novel schemes for perceiving the presence of dynamic objects in the robot's neighborhood. One of them, called the Model-Based Approach (MBA) detects motion by observing changes in the features of the environment represented on a map. The other CBA (cluster-based approach) partitions the contents of the environment into clusters representative of the objects. Inspecting the characteristics of the partitioned clusters reveals the presence of dynamic agents. The extracted dynamic objects are tracked in consequent samples of the environment through a straightforward nearest neighbor rule based on the Euclidean metric. A distributed fuzzy controller avoids the tracked dynamic objects through direction and velocity control of the mobile robot. The collision avoidance scheme is extended to overcome multiple dynamic objects through a priority based averaging technique (PBA). Indicating the need for additional rules apart from the PBA to overcome conflicting decisions while tackling multiple dynamic objects can be considered as another contribution of this effort. The method has been tested through simulations by navigating a sensor-based mobile robot amidst multiple dynamic objects and its efficacy established.

Key words: sensor-based mobile robot, dynamic objects, real-time detection and tracking, clustering-based approach, model-based approach, collision avoidance, fuzzy rule-base.

1. Introduction

Proposed in this paper a methodology for detecting, tracking and avoiding many dynamic objects in the neighborhood of a navigating robot. A priori information regarding the objects' position and velocity are not available to the algorithm. It was felt that an initial classification of the neighborhood in terms of static and dynamic agents would simplify the strategy for collision avoidance that follows classification. As a consequence only the dynamic objects need to be tracked and their future positions predicted while the static objects need not be subjected to such treatment.

The classification of the environment into static and dynamic objects is accomplished through two approaches, both of which involve representing the contents of the environment on a map. The first approach (MBA) detects a dynamic object when significant changes in its features as represented on a map are observed over a time window. The other approach (CBA) partitions the contents of the environment into clusters where the partitioned clusters indicate the number of objects present in the neighborhood. The properties of the cluster determine the nature of the object as static or dynamic. The CBA is a self-organizing clustering algorithm capable of ascertaining the number of clusters present in a given data set. The detected dynamic objects are tracked in the subsequent scans of the environment through a straightforward nearest neighbor strategy.

A fuzzy rule-base scheme does the collision avoidance of the tracked objects through velocity and direction control. The control strategy can be considered distributive in the sense each object tracked is evaded by a separate fuzzy controller. This is different from the centralized controller used in stationary environments [12, 14] where the avoidance law is not applied with respect to each object but to the entire environment as a whole. It is to be noted that only the dynamic objects are tackled by the strategy discussed in this paper while the static objects are circumvented through an earlier approach of ours [14]. The fuzzy rule-base scheme is developed initially for a single dynamic object. When multiple dynamic objects are detected the avoidance strategy takes recourse to one of the following three alternatives:

(i) The collision avoidance strategy for a single dynamic object is extended to the case of multiple objects through a priority based averaging scheme (PBA), where each of the dynamic object is assigned a priority.
(ii) The top two objects with maximum priority are considered and collision avoidance strategy is applied to the two objects with the aid of additional heuristic rules as the actions generated by the fuzzy rules for the individual objects gives rise to conflicts.
(iii) The objects are avoided by letting the robot turn along the direction where maximum freespace between two adjacent objects is detected.

The three alternatives correspond to different possibilities in the configuration of dynamic objects that would be elaborated in Section 3 of the paper.

The main contribution of this effort we feel are the two strategies for detecting dynamic agents amidst static objects. Though literature abounds in work related with navigation in dynamic environments, not much exploration has been along the directions of an explicit classification of the detected objects as static or dynamic. Some of the earliest methods that incorporated dynamic object avoidance can be found in the works of Fujimura [6] and Shih [20]. Both these methods involved modeling the stationary and dynamic objects in a four-dimensional space–time graph. A typical graph search reveals the collision free path. Other methods computed the robot's path in a static environment using global strategies. As the

robot traces this path to reach its goal dynamic objects are further introduced that crisscross its path. Avoidance is achieved by re-planning the original path through a velocity control in some cases [7] and velocity and direction control in others [16, 23]. A similar approach that interleaves planning and execution with the robot following a pre defined path that can be subject to modifications was proposed by Lamadrid and Gini [11]. Later on some fuzzy rule based approaches [12, 22] were reported for dynamic object avoidance through a velocity control scheme. The problem with a purely velocity control scheme is its inability to avoid objects that approach the robot along a more or less head on direction. Hiraga and others proposed a neuro fuzzy based strategy [8] that incorporated both direction and velocity control but the algorithm had been dealt with from the context of avoiding a single dynamic object only. Recent papers have extended the basic philosophy of collision avoidance between a robot and dynamic objects to cooperative collision avoidance among multiple mobile robots that plan and execute a task [6].

However in none of the approaches listed above there is a scheme for detecting the presence of dynamic objects amidst the static ones. In case of global approaches such as the space–time graphs of Fujimora and Shih [5, 20] the need for such detection does not arise for the positions and velocities of all the objects are assumed to be known a priori and dynamic objects are those whose velocities are nonzero. In approaches where prior information about the objects are not known such as [8, 12] there still appears to be a tacit prior assumption regarding the static vis-à-vis dynamic attribute of the detected object. In other words even amongst algorithms that navigate in the absence of global information such as in sensor-based navigation, a priori information regarding the attribute of the sensed object does appear to sneak in. Recently Chang and Song [4, 21] have circumvented this problem implicitly by predicting the future sensor readings based on past and present readings. 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 attribute.

In this way the proposed schemes for explicit detection of dynamic agents renders novelty to the theme of this paper. The algorithm needs to track and predict the future positions of only the dynamic objects for further collision avoidance. This we claim to be more consistent with human intuition as humans are generally aware whether they are cognizing a static or dynamic object and accordingly plan their paths. Very recently a method that makes use of what is called the timestamp map [19] has been proposed for explicit classification of attributes. The timestamp approach considers only the two most recent samples of the environment for classification and hence expects noticeable changes within two successive scans and detects only very fast moving objects. The present approach considers a cumulative effect over a series of scans and is capable of discerning slow moving objects also. Apart from this there are also basic differences in the methodologies adopted in the two approaches.

Another contribution of the present effort as felt by the authors relating to collision avoidance is its indication that a collision avoidance strategy applicable for a single dynamic object cannot in general be extended over multiple dynamic objects when apriori information is unavailable. Many approaches appear to have bypassed this observation by extending their strategies over multiple objects through the commonly adopted priority-based averaging (PBA) technique [5, 22]. The PBA has inherent limitations for a certain configuration of objects where the avoidance measure adopted for individual objects gives rise to conflicts. This is somewhat akin to the local minimum problem [13, 18] where the goal orienting and obstacle repulsing tendencies of the robot cancel each other. To overcome such conflicts the normal extension through PBA needs to be overridden through a different strategy or rules. These set of rules come into play only when a conflict is detected.

As far as tracking of moving objects are concerned most of the methods are devoted to vision-based tracking predominant among them being the optical flow method [9], its improvements [15, 17] and motion-energy-based methods [3, 14]. The present approach deals with real-time data obtained from range sensors where detected objects are tracked in the subsequent samples based on the customary Euclidean metric scheme. This scheme becomes relevant in the absence of a priori information regarding the parameters of the moving objects and is suitable also from the point of view of real-time efficiency.

The rest of the paper has been organized as follows. Section 2 deals with the two schemes that classify an object as stationary or dynamic and the tracking of such classified dynamic objects. Section 3 discusses the various considerations involved in avoidance of multiple dynamic objects through a fuzzy rule based approach. Section 4 portrays the efficacy of the algorithm through simulation while Section 5 presents the conclusions and future scope of this work.

2. Real-Time Motion Perception

The problem of ascertaining the presence of dynamic objects in the 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 the range sensors on the other hand are nothing but discretized spatial samples, which represents 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 inscrutable especially if the environment is non-stationary and the sensors are themselves subjected to translation and rotation. Despite these difficulties ultrasonic sensors have been popular and found suitable for real-time navigation purposes.

It is also to be noted that the problem of motion detection is an involved one especially if the sensors are prone to motion. This leads to a situation where ap-

parent changes in the robot's perception of the environment may not only be due to the actual motion of the objects but also due to a change in point of view of the sensor. Distinguishing between actual moving objects and unknown stationary objects seen from a different viewpoint is not a trivial problem to solve in general.

2.1. MOTION PERCEPTION BY MBA

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 below.

2.1.1. *Pre-processing Modules in MBA*

The preprocessing modules consist of feature extractor (FE) and 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.

(i) Object demarcation. The FE initially demarcates those objects, which are distinctly separable, i.e., objects that are demarcated by virtue of one or more sensor reading that indicate a free-space between two sets of contiguous readings. For example, in Figure 1 object A is detected by sensors 7–10 and object B by sensors 12–14. The objects are demarcated through sensor 11 whose reading indicates a free-space.

The FE then looks for occluding objects that are not demarcated due to a free-space sensor reading between them, such as in Figure 2(a). These objects are demarcated by observing a sudden jump within a contiguous set of sensor readings (Figure 2(a)), or through a pattern of increasing range readings followed by a decrease (Figure 2(b)) during a counter-clockwise 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

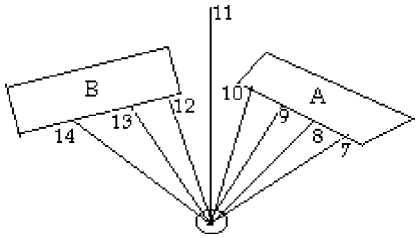
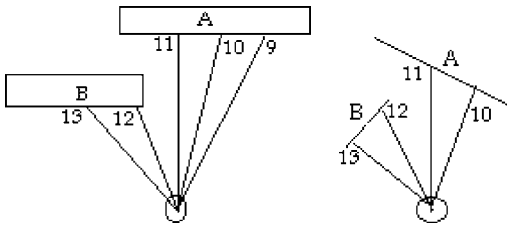


Figure 1. Object A detected by sensors 7–10, object B by 12–14 with 11 indicating free space.



(a) (b)

Figure 2. (a) Objects demarcated by a sudden jump between the reading of sensor 11 and 12; (b) objects demarcated by a pattern of increase–decrease in a counter clockwise scan.

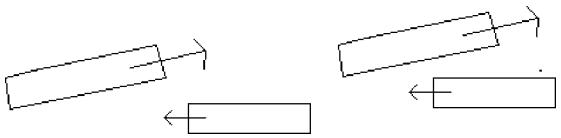


Figure 3. Between two success scans of the environment the object gets occluded partially that can cause a subsequent shift in the center and can get classified as a new object.

in patterns. The RLS also parameterizes the visible edges in terms of their slopes and intercepts.

(ii) Object tracking. The parameters of the dynamic object that are to be tracked are the two endpoints of the extracted edges and the center point. If a single point represents an object then there is only one parameter to be tracked. In MBA the range data set of every scan is reduced to its features and the correspondence between the features representing the same object in successive scans is determined by the nearest-neighbor criterion. For each visible edge representing an object at time t, the edge closest to it at $t - 1$ is determined in terms of the Euclidean metric norm applied between the edge centers at the two sampling instants. If this distance to its nearest neighbor is smaller than a neighborhood threshold, n_{th}, we assume that both the edges represent the same object. The parameters representing the object at $t - 1$ are updated by the parameters of the object at t that become the new parameters of the object. An interesting case that arises while tracking we feel needs mentioning. Shown in Figure 3 is a case where an object that was fully visible in the previous instant gets partially occluded due to which the shift in the center is considerable and can get detected as a new object in the subsequent instant. In other words the distance between the centers occupied by the object at t and at $t - 1$ exceeds nth leading to the classification of the occluded object at t as a new object. To avoid such a situation 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, discards objects that are no longer visible are not mentioned here for they are not pertinent to the main theme of discussion.

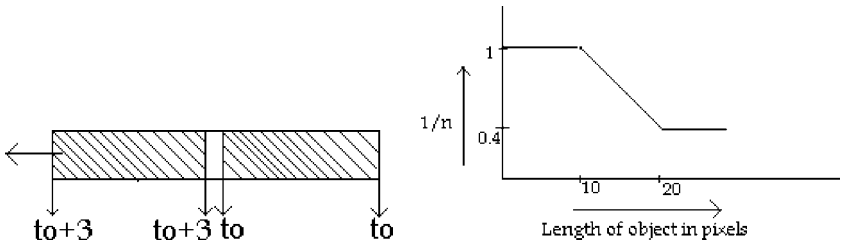
2.1.2. *Object Classifier*

The tracked objects are classified in two tracks. One track is relevant for objects whose extracted edges are parallel to *its* (object's) own motion direction and other with objects whose edges are perpendicular to its direction of motion. For objects whose edges have a parallel and a perpendicular motion component either of the scheme is applicable.

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, m_{th}. This is portrayed in Figure 4(a) where the rear end of the object gets past the location occupied by the front end at t_0 in 3 samples. 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 $1/n$ of length of the edge that 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 4(b). Fixing the value of n higher than 3 can cause misclassification for it is observed that even in static objects one of the endpoints gets past a portion of the length of the object over a temporal window. This is because the sensors detect different parts of the object at different instances and the two endpoints of the edge appear to vary with every sample even for a static object.

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 MBAPEMD.
If $(x_f(t) > x_r(t))$ **and** $(y_f(t) > y_r(t))$ *for an object k*
  If in $\varepsilon < n_{\text{th}}$ *number of samples*



(a) (b)

Figure 4. (a) PEMD. The rear edge of the object gets past the location of the front edge at to after 3 samples at to +3; (b) plot of length of object vs. the ration 1/n.

If $x_r(t+\varepsilon) > x_r(t) + \dfrac{x_f(t) - x_r(t)}{n}$ **and** $y_r(t+\varepsilon) > y_r(t) + \dfrac{y_f(t) - y_r(t)}{n}$
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. Evidently the assumption is that objects move along linear or piecewise linear trajectories. This assumption appears inevitable in the absence of apriori information about the motion characteristics of the objects. Nonetheless the above procedure can still detect objects that move along a nonlinear path such as a parabola as long as the condition mentioned in the above procedure gets satisfied.

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 5 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 angles made by the robot's heading direction with respect to the object's center at the two instants are λ and ψ (see Figure 5), respectively.

In actual simulations and implementations there would be shifts in the object's center as figured out by the MBA 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

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

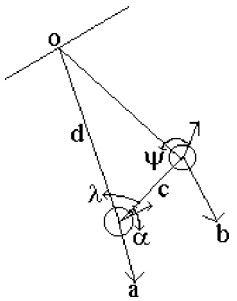


Figure 5. Static object tracked in two succesive samples by its center marked as o.

The expected rate of decrease between two samples of the environment can be written as

$$\hat{s}_i = \hat{d}_t - d_{t-1}. \tag{2}$$

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:

$$\bar{s} > \bar{\hat{s}} + \Delta, \tag{3}$$

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

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

and

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

It is to be noted that Δ is *not* a problem specific parameter, it can be considered as some of 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.

2.1.3. *Post-processing in MBA*

The following modules constitute the exception handler that forms the core of the post processing involved in MBA.

Dynamic object occlusion. This subroutine checks if a dynamic object gets occluded fully by another dynamic or static object. This is detected when a dynamic object that was well within the detection range of the sensors at $t-1$ was not tracked at t. The routine then checks if the occluded flag was set for that object at $t-1$. Detecting partially occluded objects and non-separable distinct objects as described in Section 2.1.1 sets the occluded flag. If the occluded flag was set at $t-1$ the algorithm deciphers that the object has become occluded completely at t. The algorithm then continues to update the parameters of the occluded object using a predictor though the sensors do not actually detect it. This is continued till the predicted values get beyond the detecting range of the sensors or the object becomes visible again.

Instantaneous object classification. The objective of this routine is to find out those situations where the algorithm detects a new face of a dynamic object while the old face is no more tracked. This happens when a cluster at $t - 1$ that was well within detection range does not get tracked at t while a new cluster is detected at less than far range. A new cluster is generally expected to get detected at far ranges. If this does not happen it can be that the sensors detect a new face of an already existing object. This is checked if the new cluster is closest to that cluster of $t - 1$ that has disappeared at t. If this were so it can be concluded that the new cluster is actually a new face of an already detected object. In such a case the new cluster is instantaneously classified as dynamic if the old cluster was so classified.

There exist other circumstances where the exception handler is invoked such as when a cluster of the previous instant gets associated with two clusters in the subsequent instant that are not discussed here for succinctness. It is admitted that establishing correspondence between the cluster centers through the nearest neighbor criterion is not a very accurate method for object tracking considering the many exceptions that arise above. Better methods may be to use a shape matching technique for tracking the clusters. But their suitability in fast tracking for real-time implementations has to be evaluated considering the time intensity involved. In our simulations however we have found that that tracking based on cluster centers along with the exception handler is adequately reliable.

2.2. CLUSTERING BASED APPROACH – I (CBA-I)

In MBA the feature vectors employed for feature extraction were those obtained during the most recent sample. In the CBA scheme 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 6(b) and (c). Figure 6(b) shows the point cloud of the workspace of Figure 6(a) as seen by the sensors in one sample while 6(c) is the accumulated point cloud over the last 5 samples and the clusters are more discernible than in 6(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.

2.2.1. *Preprocessing Module*

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

- Algorithm is self-organizing and determines the number of clusters.

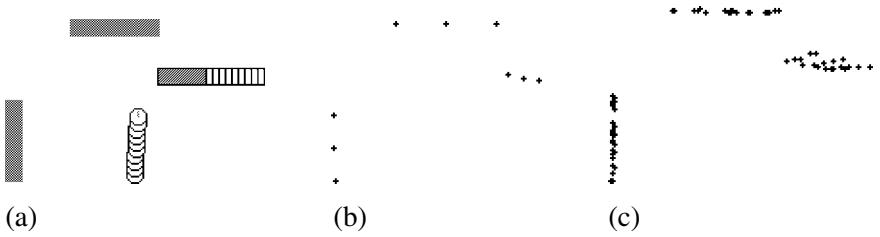


Figure 6. (a) Robot scanning an environment with two static and one dynamic object; (b) features extracted based on the last scan alone; (c) features extracted over the last five samples.

- Makes use of two distance measures, center to feature vector distance (CVD) and vector to vector distance (VVD). Both the distance measures have their thresholds denoted as cth and vth, respectively.
- The thresholds decide the formation of a new cluster. The thresholds are however adaptive and changes according to the distribution of the feature vectors.
- The VVD uses the standard Euclidean metric while CVD employs the Mahanolobis distance.
- The partitioning is achieved in one shot in the sense the feature vectors have to be presented only once to the algorithm.

The algorithm can be considered novel in the sense 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. However certain features of it makes it suitable for the problem under consideration in the context of real-time navigation without prior knowledge of the workspace, 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 $S = \{X_1, X_2, \ldots, X_N\}$, where each $X_i = \{x_i, y_i\}$, initial threshold values for cth and vth, forgetting factor β and boundary parameter η. In the algorithm given below parameter $\alpha = 1 - \beta$.

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

PROCEDURE CBA_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. Repeat steps 3 to 6 until all the feature vectors have been tagged as considered.
   3. Find the distance of the nearest, *unconsidered* feature vector (FV) X_j to the currently considered FV X_i. Denote this distance as *vvd*; $vvd = \min_{j \neq i} d_e(X_i, X_j)$, $j = \{0, 1, \ldots, i-1, i+1, \ldots, N\}$, where d_e is the Euclidean norm.
   4. *If* $vvd \leqslant v\text{th}$
      4a. $cluster(X_j) \leftarrow C_{cur}$, where $C_{cur} = cluster(X_i)$,
      4b. $n(C_{cur}) = n(C_{cur}) + 1$; increment the number of vectors in a class,
      4c. Update center and covariance matrices for the cluster,
      4d. Update VVD threshold as $v\text{th} = \alpha(v\text{th}) + (1-\alpha)vvd$,
      4e. Find $cvd = d_m(X_j, z_{cur})$, where d_m stands for the Mahanalobis norm,
      4f. If $cvd > c\text{th} - \eta$ then $c\text{th} = cvd + \eta$; updating the CVD threshold.
   5. *If* $vvd > v\text{th}$
      5a. Find $z_k = \min_i d_m(X_j, z_i)$, $i = \{0, 1, \ldots, K\}$ the nearest center of a cluster to X_j,
      5b. $cvd = d_m(X_j, z_k)$,
      5c. *If* $cvd \leqslant c\text{th}$,
         5c1. $cluster(X_j) \leftarrow C_k$, $C_k = cluster(z_k)$,
         5c2. $n(C_k) = n(C_k) + 1$,
         5c3. Update centers, covariance matrices for C_k.
      5d. *Else if* $cvd > c\text{th}$
         5d1. $K = K + 1$; forms a new cluster,
         5d2. $z_K = X_j$; the center of the new cluster takes the value of the FV X_j,
         5d3. $n(K) = 1$; number of FV in the new cluster is initialized to 1,
         5d4. Set the distance thresholds to their starting values.
   6. Tag X_j as considered and assign $X_i = X_j$ so that X_j becomes the FV over which steps 3–5 are applied for the next pass of the algorithm.
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 a FV of another object.

The algorithm can be best understood through Figures 7(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 Figures 7(a) and (b) denote the starting feature vector considered (step 1 of the algorithm). The initial threshold, vth 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 gets added as a FV of the same object, the center,

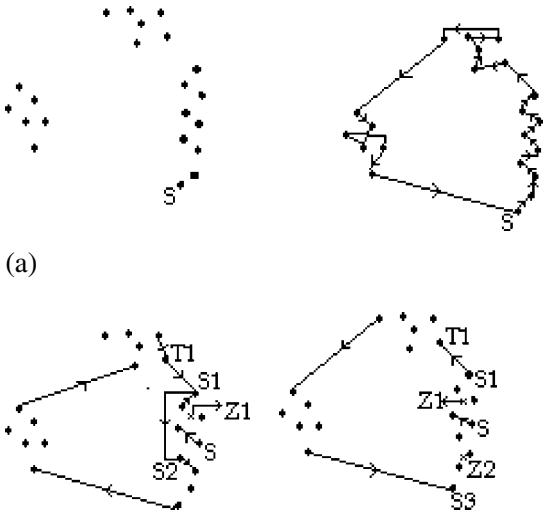


Figure 7. (a) The point cloud representation of an environemnt is shown on the left. The right figure represents the direction of the search starting S. (b) The search branches from FV S1 to S2 of the same cluster. When the search reaches S1 the center has shifted to Z1. Center Z1 is shown as a cross. (c) The search proceeds to T1 from S1 and reenters at S3. The centers of the split cluster Z1, Z2 are shown using a cross.

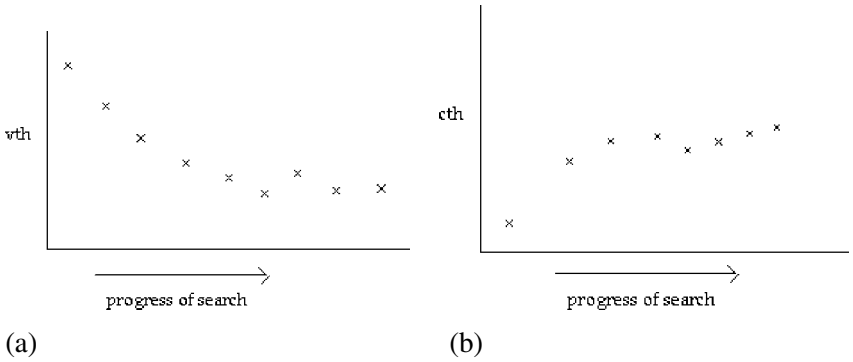


Figure 8. (a) Typical variation of threshold vth for a cluster; (b) typical variation of threshold cth for a cluster.

variances and the thresholds for that object get updated (steps 4(a)–(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 vth shrinks and settles at a value indicative of the average inter-vector partition within a cluster. Figure 8(a) shows the typical plot of vth for a cluster as the search proceeds. The forgetting factor $\beta = 1 - \alpha$

is responsible for adapting vth to the inter-vector 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 vth 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 5c a new cluster is formed with its center and thresholds initialized according to steps 5d1–5d4. 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 7(b). When the search reaches an end of the cluster with FV S1 (Figure 7(b)) the closest *unconsidered* FV to S1, which is S2 shall fail the criterion of step 4. However since it lies within the threshold cth with respect to center Z1 (Figure 7(b)) of the same cluster at that instant it (S2) gets assigned as a FV owned by the same object that owns S and S1. The search then proceeds from S2. The arrows in Figure 7(b) also indicate the branching of the search from S1 to S2. While vth is adapted to capture the average inter-vector partition distance within a cluster, cth 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 cth adapts itself to trace the boundary of the cluster when the Mahanalobis norm is employed. The typical variation of cth for a cluster with the progress of search is shown in Figure 8(b). The boundary parameter n of step 4f is instrumental in achieving this adaptation.

On the other hand, if T1 becomes the closest FV with respect to S1of 7b then T1 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 7(c) and the starting cluster is again reentered through the feature vector S3. If S3 passes the criterion of 5c then it gets assigned to the object that owns S and S1. If it fails, it splits the object into two clusters with centers Z1 and Z2 indicated by crosses in Figure 7(c). Such split-clusters are merged through a compatible cluster-merging scheme discussed very briefly below.

Compatible Cluster Merging (CCM). 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 needs to be merged. One category is the split cluster discussed above 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:

(i) The closest inter-pixel distance between any two feature vectors owned by the individual clusters is comparable with the steady state threshold vth obtained for each of the two clusters.

(ii) 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 [10]. Let the centers of the two clusters be v_i and v_j, the eigenvalues of the two clusters be $\{\lambda_{i1}, \ldots, \lambda_{in}\}$ and $\{\lambda_{j1}, \ldots, \lambda_{jn}\}$ and the normalized eigenvectors be $\{\phi_{i1}, \ldots, \phi_{in}\}$ and $\{\phi_{j1}, \ldots, \phi_{jn}\}$. The eigenvalues and eigenvectors are arranged in descending order of the eigenvalues. The criteria proposed by the authors' [10] are stated as follows

$$\text{(i)} \quad \frac{\|v_i - v_j\|}{\sqrt{\lambda_i} + \sqrt{\lambda_j}} \leqslant k_3, \quad k_3 \text{ lies between 2 and 4,} \tag{4}$$

$$\text{(ii)} \quad |\phi_{in}^{\mathrm{T}} \phi_{jn}| \geqslant k_1, \quad k_1 \text{close to 1,} \tag{5}$$

$$\text{(iii)} \quad \frac{\phi_{in}^{\mathrm{T}} + \phi_{jn}^{\mathrm{T}}}{2} \frac{v_i - v_j}{\|v_i + v_j\|} \leqslant k_2. \tag{6}$$

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 should 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 exists more sophisticated merging criteria [1] the above 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 [1].

2.2.2. *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.

2.2.3. *Parallel Edge Motion Detection*

The above classification scheme works well for dynamic objects whose visible edges are not completely parallel to the object's direction of motion. The objects whose visible edges are entirely 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 MBAPEMD.

3. Collision Avoidance by a Distributive Fuzzy Control

The collision avoidance behavior module comprises of a motion predictor (MP) and a fuzzy controller that distributes itself over the dynamic objects when more than one object is detected. The object attribute classification is achieved through the MBA discussed in the previous chapter. The MBA was chosen considering its rotational invariance, computational cheapness and ease of implementation. Future versions plan to incorporate CBA that is both robust to rotational displacement as well as noisy data. The motion predictor computes the future path of the object based on its motion parameters estimated through the features of the object extracted and tracked by the MBA. The locations of intersection of the object's center with the robot and the time (in samples) of collision from the current instant are computed for each object based on the motion parameters. The parameters are estimated using the recursive least squares technique considering their simplicity. The fuzzy rules compute the changes in the robot's velocity and direction based on the estimates of the location and time of collision. Though explicit probabilities of collision are not computed it is acknowledged that the point of intersection gives only a rough estimate of the region of collision. Fuzzy inference techniques are especially suited for handling such uncertainty in information and hence an appropriate choice for modeling not so precisely deterministic information.

The objective of the algorithm is to steer the robot to its destination past static and dynamic objects with no global information regarding the positions or trajectories of the objects or the nature of the object with regards to its static or dynamic attribute. The algorithm however makes the following assumptions while computing a collision free path.

- 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 has to stop every time it evaluates its neighborhood. The assumption is only to facilitate simplified computations.
- The dynamic objects chart piecewise linear paths though their positions and velocities are not known apriori. In many real-time environments objects do move for most duration along a linear path except while changing directions.

Thus the assumption of piecewise linear paths does not appear unrealistic for most situations. Piecewise linearity allows for changes in direction while expecting a predominantly linear trajectory.

3.1. FUZZY CODING OF COLLISION AVOIDANCE FOR A SOLITARY DYNAMIC OBJECT

The inputs to the fuzzy rules consists of:

(i) the angular separation between the direction of approach of the dynamic object and the robot's direction of motion ($\Delta\theta$),
(ii) the expected temporal difference between the object and the robot in reaching the collision point ($\Delta t = t_{\text{ocp}} - t_{\text{rcp}}$), t_{ocp}, t_{rcp} are time taken by the object and robot to reach the collision point and
(iii) the distance of the object's center to the robot at the instant of sampling (Δs).

The fuzzy rules output actions that actuate a change in the direction of robot's motion (Δy_d) and a change in its velocity (Δv_d). These rules vary according to the quadrant in which the moving object is positioned with reference to the robot's location. From the point of view of the robot these objects are classified as being on (i) *Front-Right* (FR, 1st quadrant), (ii) *Front-Left* (FL, 2nd quadrant), (iii) *Rear-Left* (RL, 3rd quadrant) and (iv) *Rear-Right* (RR, 4th quadrant). The nature of the rules that tackle an object that approaches the robot from its front-left (FL) are shown in the inference table for direction control (Table I). Table II represents the inference scheme for velocity control. The connotations of the fuzzy linguistic variables denoted in the table are to be read as the terms indicated in brackets below:

S (Small), L (Large), F (Far), N (Near), LP (Large Positive),
MP (Medium Positive), SP (Small Positive), ZP (Zero Positive),
LN (Large Negative), MN (Medium Negative), SN (Small Negative),
ZN (Zero Negative).

The first rule in Table I translates as:

If $\Delta\theta$ is *small* and Δt is *small positive* and Δs is *near*
then Δy_d is *large positive*.

Similar inference rules are fired when the robot encounters an object along the other three directions.

3.1.1. *Some Reflections on the Fuzzy Rules*

A brief discussion on the motivation behind the above rules is presented here. The rules have been formulated to capture the elements of commonsense reasoning

Table I. Fuzzy inference for orientation control of the mobile robot to tackle an object that approaches from the front-left

Rule index	$\Delta\theta$	Δt	Δs	Δy_d
1	S	SP	N	LP
2	S	SP	F	MP
3	S	LP	N	MP
4	S	LP	F	SP
5	L	SP	N	MP
6	L	SP	F	SP
7	L	LP	N	SP
8	L	LP	F	ZP
9	S	SN	N	LN
10	S	SN	F	MN
11	S	LN	N	MN
12	S	LN	F	SN
13	L	SN	N	MN
14	L	SN	F	SN
15	L	LN	N	SN
16	L	LN	F	ZN

Table II. Fuzzy inference for velocity control of the mobile robot

Rule index	$\Delta\theta$	Δt	Δv_d
1	S	SP	LP
2	S	LP	MP
3	S	SN	0
4	S	LN	SN
5	L	SP	MP
6	L	LP	SP
7	L	SN	MN
8	L	LN	SN

involved in collision avoidance of a dynamic object. The angular separation $\Delta\theta$, the time difference in reaching the point of collision Δt and the distance between the robot and the object Δs were identified as the crucial underpinnings involved in formulating an effective collision avoidance strategy. The following commonsense reasoning lies behind the fuzzy inference scheme presented for direction control.

(i) Small values of angular separation between the robot and the approaching object necessitate the robot to deviate from its current direction of motion. As

the separation becomes smaller the approach is more head-on and a purely velocity control would be of no avail. Hence smaller the angular separation larger should be the deviation in robot's motion direction and *vice versa*.

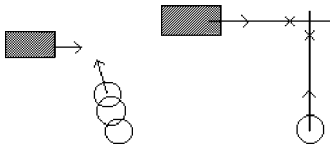
(ii) If the object gets first classified as dynamic at near distances to the robot a possibility of collision cannot once again be thwarted by velocity control alone irrespective of the angular separation. One such situation is shown in Figure 9(a). Here slowing down or increasing the speed of the robot without change in orientation may not guarantee a collision free path. Thus first classification of the dynamic attribute at small distances entails a moderate deviation that reduces as the distance increases.

(iii) If the temporal difference in reaching the point of intersection is small, it indicates that the robot and the object would come at very close distances in the future though they are not proximal at the current instant. Since the robot as well as the object is of finite size it could always be the case they graze each other at the closest point of approach despite changes in the robot's velocity. Hence as a measure of prudence a moderate change in direction is actuated for small temporal differences that decrease as the temporal difference increases. Moreover such measures are required considering the fact that the collision location can never be located accurately in practice. One such situation is shown in Figure 9(b) where the closest point of approach for the two bodies are marked with crosses.

The above musings get captured in the fuzzy rules tabulated in Table I. For example the first rule in the table denotes that if the angular separation is small and so are the temporal difference and the distance of the object at first identification the change in orientation is of a large magnitude. The magnitude reduces in a fuzzily manner if one or more of the above conditions are not vindicated.

As far as velocity control is concerned the following intuitive reasoning dictate the rules of Table II.

(i) If the robot would be marginally ahead of the object at the location of collision then the velocity of the robot ought to be increased in larger amounts with every sample. This would facilitate the robot to reach the point of collision with a larger time lead than the lead predicted at the instant of detection.



(a) (b)

Figure 9. (a) Initial identification of the object as dynamic at close distances from the robot entails the robot to deviate even if angular separation is large. (b) The bodies come close to each other in the future at positions indicated by crosses.

However if the robot is expected to reach the collision point well ahead of the object it would suffice to increase the velocity in minor quantities.

(ii) Similarly if the object would be ahead of the robot with a minor lead then the velocity of the robot is decreased in sizeable amounts to increase the lead of the object. If the early lead of the object was substantial the robot's velocity need to be decreased in a marginal fashion only.

The algorithm activates the collision avoidance module when the object first gets classified as dynamic. The algorithm deactivates the module once the point of intersection of the future trajectories appears on the rear of the robot or the object. Usually the deactivation does not happen over one instant, the algorithm waits for the intersection point to appear on the rear for three consecutive instances before deactivation.

For the purpose of convenience the robot is classified as performing one of the following behaviors: (i) the escape behavior, (ii) go-slow behavior and (iii) the overtake behavior. The escape behavior occurs when the object approaches the robot with narrow angular separation forcing the robot to deviate from its current direction of motion in a tangible manner. The robot is considered to perform overtake behavior as it increases its velocity to get past an approaching object and the go-slow behavior while it reduces its velocity to tackle the object. Such classification facilitates framing heuristics while tackling multiple dynamic objects that would be discussed further down.

3.2. EXTENDING TO MULTIPLE DYNAMIC OBJECTS

It is the case in general that two objects do not get classified as dynamic during the same instant. An easy way for extending the collision avoidance to multiple dynamic objects is through the PBA depicted below.

3.2.1. *Collision Avoidance through Priority Based Averaging (PBA)*

Each classified dynamic object that has yet to be completely avoided is assigned a priority:

$$p_i = \frac{w_{1i} + w_{2i}}{2}. \tag{7}$$

Here w_{1i} is the factor that considers the closeness of the dynamic object i from the robot and w_{2i} considers the rate of change in the distance from the object's center to the robot over the last time window. They are given by

$$w_{1i} = \begin{cases} 1, & 0 \leqslant \bar{d}_c < 0.35, \\ 2 - 2.85\bar{d}_c, & 0.35 \leqslant \bar{d}_c < 0.7, \\ 0, & \text{elsewhere}, \end{cases} \tag{8}$$

here, $\bar{d}_c$ is the value of Δs normalized in the range [0, 1].

$$w_{2i} = \begin{cases} 1, & 0 \leqslant d_{\mathrm{m}} < 2, \\ (5 - d_{\mathrm{m}})/3, & 2 \leqslant d_{\mathrm{m}} \leqslant 5, \\ 0, & \text{elsewhere}, \end{cases} \tag{9}$$

where d_{m} is the rate of change in Δs summed over the last four samples. The values of d_{m} that are negative are clipped to zero. It may be recalled that Δs is the distance from the robot to the object's center.

The net turn angle due to dynamic object avoidance behavior when *heuristics are not invoked* is given by:

$$\Delta y_d = \frac{\sum_i p_i \Delta y_{di}}{\sum_i p_i}. \tag{10}$$

The overall turn angle considering the behavior of static object avoidance, dynamic object avoidance and target reaching is given by

$$\begin{aligned} \Delta y &= w_d \Delta y_d + (1 - w_d)\Delta y_{\mathrm{st}}, \\ w_d &= \max(p_i) \quad i = 0, 1, \ldots, n, \end{aligned} \tag{11}$$

where n is the number of classified dynamic objects yet to be avoided and Δy_{st} is the turn angle due to static object avoidance and target orienting behaviors derived in reference 14 cited in this paper. Similar computations are carried out for the overall change in velocity.

3.2.2. *The Need for Additional Rules*

As mentioned before the PBA does not give a satisfactory performance on a fair number of occasions a particular case of which is discussed below. A configuration of the objects in the environment is represented as a string of 4 characters. The first 2 characters represent the positions of the 1st and the 2nd dynamic object with respect to the robot's heading direction at the time of classification. They take the denotations L, R implying Left and Right. The last two characters represent the behavior that would be executed by the robot to avoid the objects. They take the following notations (O,S,E) meaning *overtake*, *go-slow* and *escape*. A situation as shown in Figure 10 is denoted as LLOS meaning that the first and second objects are on the left of the robot and the algorithm prefers an *overtake* and a *go-slow* behavior for their avoidance. In Figures 11–14 the sketches are not the actual paths of the robot computed by the algorithm but a rough delineation to illustrate the idea. The actual simulation graphs are discussed in Section 4. In these rough sketches the robot and the dynamic objects are portrayed as points for convenience only.

Now going by the individual rules for the two objects in Figure 10 entails that the robot turn right and increase its speed for the first object while turn left and decrease the speed for the second. The conflict in the decisions involved for the

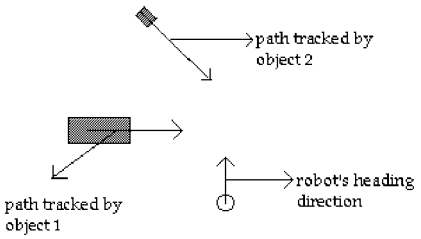


Figure 10. Configuration of objects in LLOS mode.

1st and the 2nd object is evident. A weighted average shall give a turn whose value is between the values obtained through the individual rules and a velocity that lies once again somewhere between the individual values. However it has been found that unless the robot moves with a turn angle that lies outside the convex combination of the two values and a velocity change that again lies outside the range it shall incur a collision. This forces us to introduce some more rules that can be called as heuristics for enhancing the performance of collision avoidance. These rules do not guarantee a self-sufficient scheme of avoidance but their incorporation nevertheless improves the performance. More rules have to be introduced as newer situations emerge and in this respect the limitations of the present method is accepted. An annotation of some of the heuristics that have been useful while dealing with objects that approach the robot from its front half is listed below. It should be noted that the heuristics are invoked only when the algorithm senses a possible collision if the robot moves along the direction given by the PBA as a conflict of options occur.

3.2.3. *Avoiding the Top Two Objects in the Priority List*

Heuristics listed out in this subsection are employed to avoid multiple dynamic objects in a pairwise fashion. In other words heuristics are employed for only the top two objects in the priority list. Another reason for employing heuristics in a pairwise fashion is that the number of such rules increases exponentially with an increase in number of objects considered. Moreover objects that are lower in the priority do not pose an imminent threat to the robot and are brought into the heuristics later. However a methodology for tackling 3 or more dynamic objects with nearly the same priorities is also described in Section 3.2.4 further down.

Heuristic 1 (H1) (for LLOS). If the count of the number of samples from the initiation of *overtake* mode in the first object exceeds four and the free space between the nearest endpoints of the two objects exceeds an angular threshold, turn the robot along the angular bisector between the two objects. From the subsequent scan navigate according to the decisions of the PBA. Figure 11(a) shows the sketch depicting the idea.

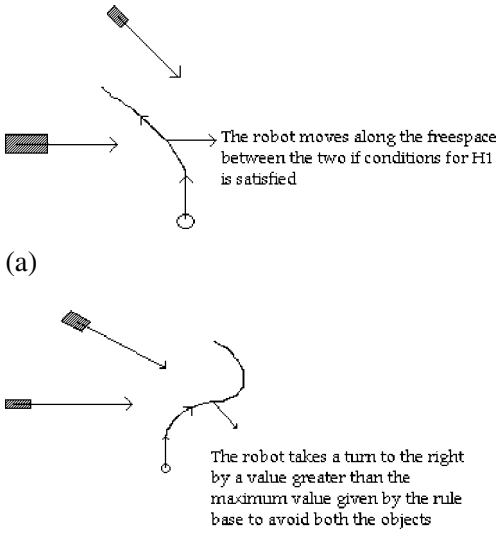


Figure 11. (a) A rough sketch of the path upon invoking H1 followed by the averager for a LLOS configuration of objects. (b) Rough sketch of the path tracked by the robot through H2 followed by the rule-base.

H2. If the conditions listed in H1 are not true then invoke the *overtake* mode for the second object with a turn to the right that is greater than the maximum of the two values and with an increase in velocity greater once again than the two individual values. Figure 11(b) shows the sketch of the path that would probably result.

Here it is to be noted that H2 overrides the decision made by the algorithm to launch the robot in the *go-slow* mode for the second object by invoking the *overtake* mode for the second object also. From the subsequent scan both the objects should appear on the robot's left and since the *overtake* mode has been invoked for both of them the averager can be invoked for steering the robot.

H3 (LLSO). The situation is illustrated in Figure 12(a). If the first object is yet to be avoided completely then invoke the *go-slow* mode for the second object to let both the objects pass by. From subsequent scans invoke the averager. The sketch of the probable path is given in Figure 12(b).

Here again path tracked by the robot without H3, purely based on the averager can graze the first object at very close distances on certain occasions.

H4 (LROO). If the objects are close to one another (shown in Figure 13(a)) turn in that direction where the angular separation with the object is less. For example if the separation with respect to the right object is less turn towards the right. From the subsequent scan both the objects shall be on the left and the robot can avoid the

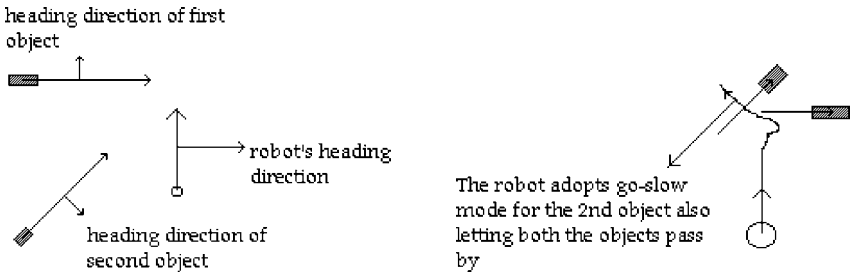


Figure 12. (a) Configuration of objects in LLSO mode. (b) Rough sketch of the path tracked by the robot through H3 followed by the rule base.

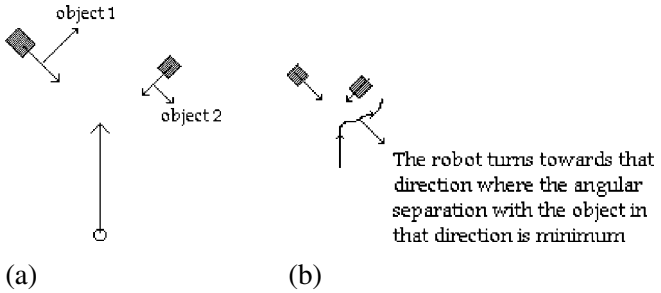


Figure 13. (a) Configuration of objects in LROO mode. (b) Rough sketch of the path tracked by the robot through H4 followed by the rulebase.

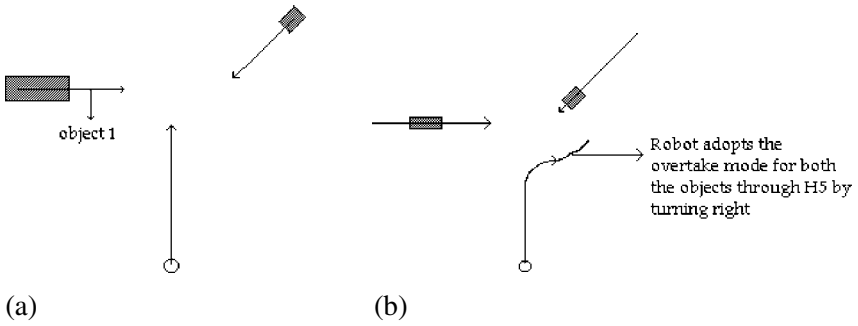


Figure 14. (a) Configuration of objects in LROS modes. (b) Sketch of the path tracked through H5 followed by rulebase for LLOO.

object at safe distances based on the outputs generated by the averager. Figure 13(b) shows the sketch of the probable path.

H5 (LROS). If the space between the objects is not enough (Figure 14(a)) turn to the right by an angle greater than the angular separation between the robot's heading direction for the second object and invoke the *overtake* mode for the second object also. Figure 14(b) shows the sketch of the probable path.

3.2.4. *Avoiding the Top Three or More Objects from the Priority List*

When more than two objects need to be avoided the change in orientation of the robot is computed as:

$$y_d = s(i)\Delta\phi_{\mathrm{m}} f(\Delta s_{\mathrm{m}}), \tag{12}$$

$$\Delta\phi_{\mathrm{m}} = \max\left|\phi_{i+1} - \phi_i\right|, \quad i = \{1, \ldots, n-1\}, \tag{13}$$

where n represents the number of objects in the priority list and ϕ_i is the angular separation of the ith object with the robot's heading direction. Here $s(i)$ takes a value 1 if the object i lies on the robot's right and -1 if it lies on its left. The $i+1$th object is closer in terms of its angular separation to the robot than the ith object if it were on the robot's right and farther away from the robot than the ith object if it were on the robot's left.

$\Delta s_{\mathrm{m}} = \min \Delta\bar{s}_i,\ i = 1, 2, \ldots, n$, where $\bar{s}_i$ is the normalized distance of the ith object's center from the robot and

$$f(\Delta s_{\mathrm{m}}) = \begin{cases} 1, & 0 \leqslant \Delta s_{\mathrm{m}} \leqslant 0.7, \\ \dfrac{(1 - s_{\mathrm{m}})}{0.3}, & 0.7 < \Delta s_{\mathrm{m}} \leqslant 1. \end{cases} \tag{14}$$

In essence the algorithm tries to find out the maximum freespace between all the pairs of adjacent objects considered from the priority list and turns along the bisector of this maximum freespace. Equation (14) acts as a smoothing function that moderates the turn from being excessively large over a single sample. The effect of (14) is to achieve the turn angle over a sequence of steps if the distance to the closest object is not near.

It is to be noted that the PBA is employed whenever the avoidance strategy adopted for the individual objects in a configuration of two or more dynamic objects do not give rise to a conflict.

4. Simulation Results

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 that are placed along the circumference with an angular separation of 15 degrees with respect to the robot's center. Simulations for motion detection and collision avoidance are reported in separate sections. In motion detection the trajectory of the robot till it identifies the dynamic object is shown while in collision avoidance the entire trajectory tracked till the target is reached is depicted. As mentioned before the maximum velocity of the robot is fixed at 6 pixels per sample (pps) and the minimum velocity at 1.5 pps. The maximum rate of change of velocity is fixed at 3 pixels over sample and the minimum rate of change of velocity is -3 pixels over a sample.

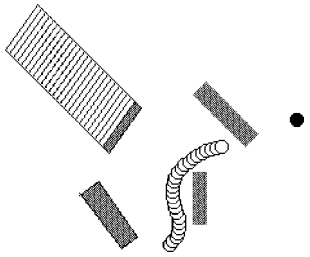
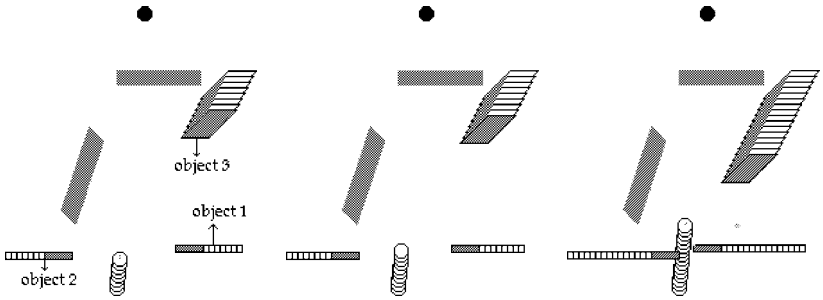


Figure 15. Instant of perceiving the dynamic object through MBA as the robot rotates to avoid the static objects.



(a) (b) (c)

Figure 16. (a) Instant of perceiving object 1 by MBA; (b) perception of object 2; (c) perception of object 3.

4.1. MOTION DETECTION

4.1.1. *Motion Detection through MBA*

Figure 15 shows the instant where the dynamic object on the rear is detected 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. The environment of Figure 16(a) consists of 3 dynamic and 2 stationary objects. The MBA could clearly identify the static and the dynamic objects. Figures 16(a)–(c) show the instances when the objects labeled 1, 2 and 3 got identified as dynamic. Figure 17 shows the navigation in a workspace consisting of long static objects. The MBA could once again track the various objects and classify their attributes accurately.

4.1.2. *Motion detection through CBA*

Figures 18–20 delineate the classification of dynamic objects through CBA-I. Figure 18(b) is the point cloud representation of the environment of Figure 18(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

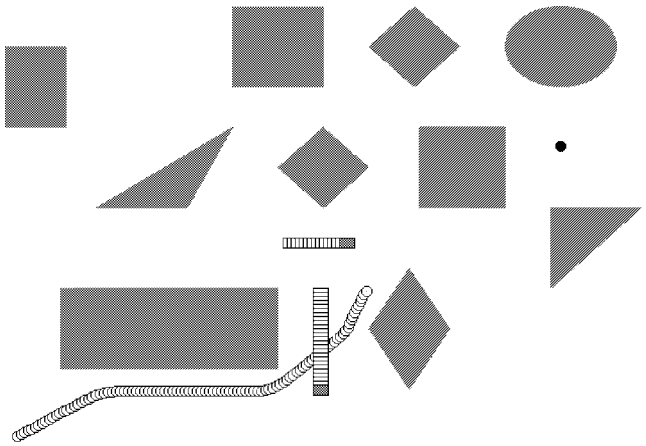
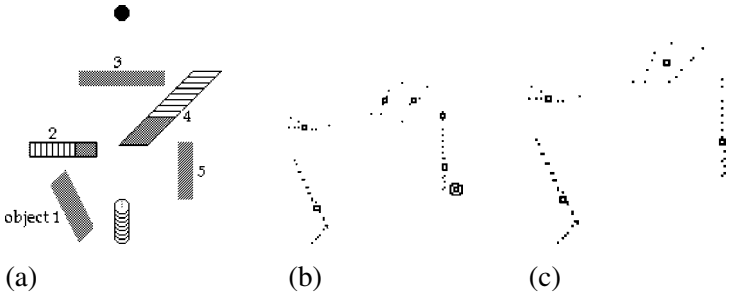


Figure 17. The MBA could classify the attributes of the objects in the above works pace accurately.



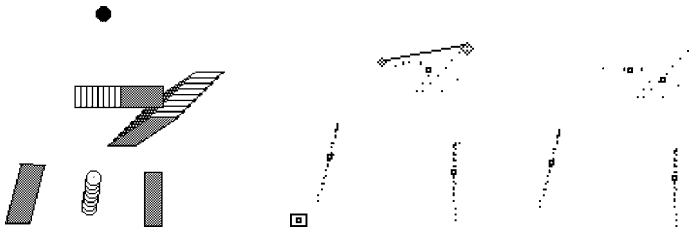
(a) (b) (c)

Figure 18. (a) An environment with 3 static and 2 dynamic objects. Objects are numbered. (b) The point cloud representation of 23(a) with the centers of clusters formed by CBA-I. A outlier cluster beside object 5 is shown cricled. (c) Cluster centers after CCM.

described in Section 2.2.1 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 a 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 18(b) the outlier besides 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 18(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-

Table III. Ratios of the standard deviations of the objects of Figure 18(a) along the principal axes of the clusters

Object index (i)	$\sqrt{\lambda_{i1}}$	$\sqrt{\lambda_{i2}}$	$\sqrt{\lambda_{i1}}/\sqrt{\lambda_{i2}}$
1	12.11	2.24	0.18
2	6.08	1.09	0.18
4	10	3.9	0.39
5	12.42	1.25	0.1



(a) (b) (c)

Figure 19. (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 2 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.

right (4) through the ratios of the eigenvalues and the one on the front left (2) through the scheme same as MBAPEMD. Table III shows the ratios of the square roots of the eigenvalues of the clusters of Figure 18(c). The labeling of the objects in Table III is consistent with the labeling indicated in Figure 18(a).

Figure 19(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 (Figure 19(b)) are merged and the CBA-I clusters both the objects together as a single entity. Such merged clusters get differentiated through a 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 19(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

Figure 20. (a) Navigation amidst 3 dynamic and 2 static objects labelled 1 to 5 in clockwise direction. (b) Initial clustering results. Dynamic objects 1 and 4 are clustered as 2 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.

distinct sets of cluster 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 set the eigenvectors are computed. The remaining FV get assigned to that cluster to which it is Mahanalobically closer. Thus the coupled cluster of Figure 19(b) gets decoupled in Figure 19(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 19(b) also gets discarded in Figure 19(c).

Figure 20(a) shows navigation amidst 2 static and 3 dynamic objects. The CBA clusters the 5 objects into 8 clusters. The clusters along with the centers is shown in Figure 20(b) with the dynamic objects labeled 1 and 4 getting decomposed into two clusters each. Figure 20(c) represents the centers after CCM. Once again the classification strategy could accurately identify the dynamic objects amidst static ones. Here all the three dynamic objects got classified by the eigenvalue method mentioned in Section 2.2.2.

4.2. COLLISION AVOIDANCE

4.2.1. *Collision Avoidance of a Solitary Dynamic Object*

Figures 21–23 consider collision avoidance of a single dynamic object. In all these figures the first part shows the instant of detection and the subsequent parts show the intermittent stages of avoidance till the target is reached. The MBA has been employed for motion detection in all the simulations as mentioned in the initial paragraph of Section 3. Figures 21(a)–(d) portray collision avoidance in the *escape* mode. The instant of detection is shown in Figure 21(a). It can be seen that the center of the robot lies within the plane swept out by the future trajectories of the two end points of the object. The deviation between the predicted trajectory and the actual path that would be followed is visible. The deviation occurs as the sensors do not detect the same location on the object during the various scans and this turns out to be the cardinal source of error that affects the performance of collision avoidance. Though the deviation is minimized by a least squares fit it cannot be

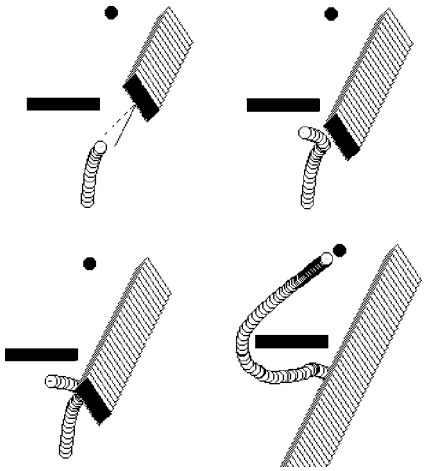


Figure 21. (a)–(d) Various stages in detection and avoidance of a dynamic object through escape mode. Figure 21(a) is the instant when the dynamic object is perceived.

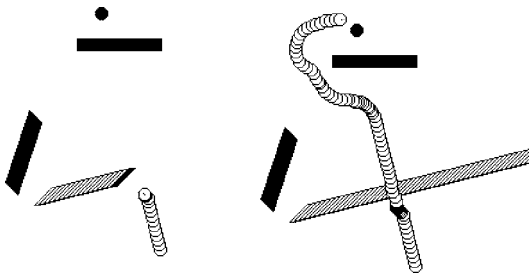


Figure 22. (a), (b) Collision avoidance in go-slow mode.

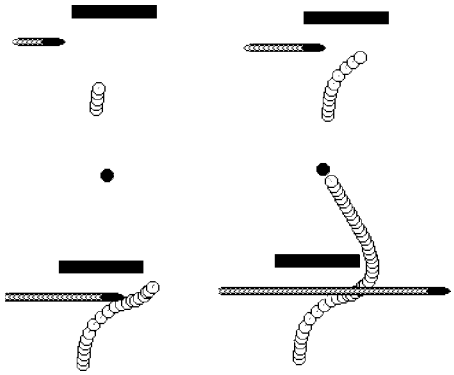


Figure 23. (a)–(d) Collision avoidance in overtake mode.

eliminated. The inherent approximate reasoning of fuzzy rules suppress this error to give an acceptable performance under most conditions. Figures 22(a), (b) illustrate avoidance in *go-slow* mode while Figures 23(a)–(d) illustrate the *overtake*

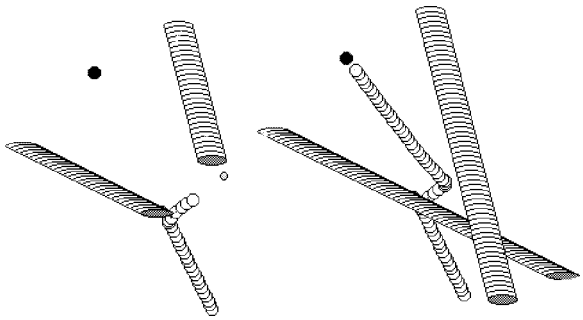


Figure 24. (a), (b) Collision avoidance of a LLOO configuration of objects.

mode. It can be seen that the robot decreases its velocity in the *go-slow* mode and increases its velocity in the *overtake* mode.

4.2.2. *Collision Avoidance of Multiple Dynamic Objects Considered Pairwise*

The simulation graphs of this section show collision avoidance of multiple dynamic objects tackled by considering them pairwise. Figures 24(a), (b) show collision avoidance for a *LLOO* sequence of objects. The propensity of the algorithm using the averager would be to turn further right since the second object was detected on the robot's left in Figure 24(a). However the target attractor module is given prominence by the heuristics considering the fact the line joining the robot's center to the target does not intersect the future paths of both the objects. A crisp turn to the left is essential for a smooth turn could make the robot vulnerable to the attack of the second object from its left. The path that ensues is shown in Figure 24(b). Figure 25 shows collision avoidance for an *LLOS* sequence of objects. The first part of Figure 25(a) shows the instant when the second object is detected. Though the second object would have been avoided in *go-slow* mode if detected solitarily the configuration of the objects in space forces invocation of H2 that launches the *overtake* mode for the second object also. Subsequently the robot negotiates the objects according to the fuzzy rules and the path that ensues is shown in the remaining parts of Figure 25. Had the robot been navigated without resorting to H2 a collision with the second object results as shown in Figure 26. Figure 27 shows collision avoidance of 3 objects when they are detected in quick succession while Figure 28 shows collision avoidance of four moving objects. Through the Figures 27, 28 the application of pairwise rule of heuristics is illustrated vividly. When the 2nd object is detected the heuristics are applied for 1st and 2nd as they are the only objects in the priority list. When the 3rd gets detected the priority of the 1st object becomes sufficiently low. Hence a possibility of conflict is checked for the actions generated by the rule-base for objects 2 and 3 that are on the top of the priority list. If a conflict arises heuristics are applied for objects 2 and 3 and from the subsequent instant the robot gets maneuvered through PBA. Similarly when the

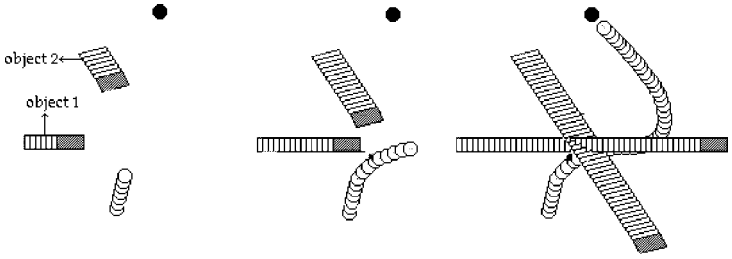


Figure 25. (a)–(c) Various stages in avoiding a LLOS configuration of objects through H2 followed by the fuzzy rules.

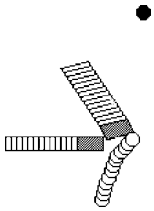


Figure 26. Without heuristics a collision is incurred for the same configuration of objects in Figure 25.

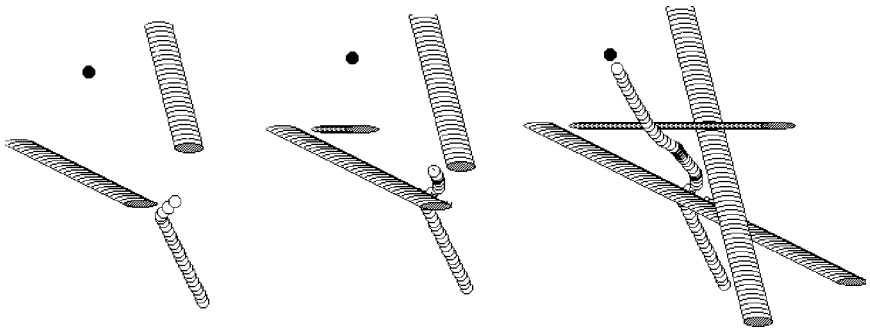


Figure 27. Various stages in avoiding three dynamic objects encountered in quick succession.

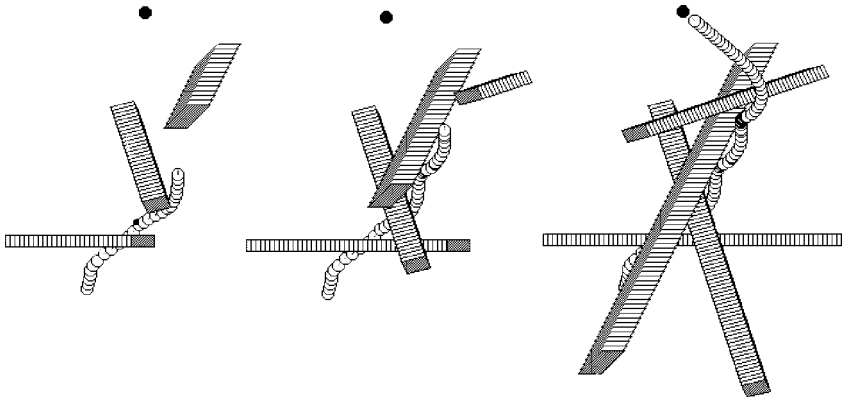


Figure 28. Collision avoidance of a sequence four objects.

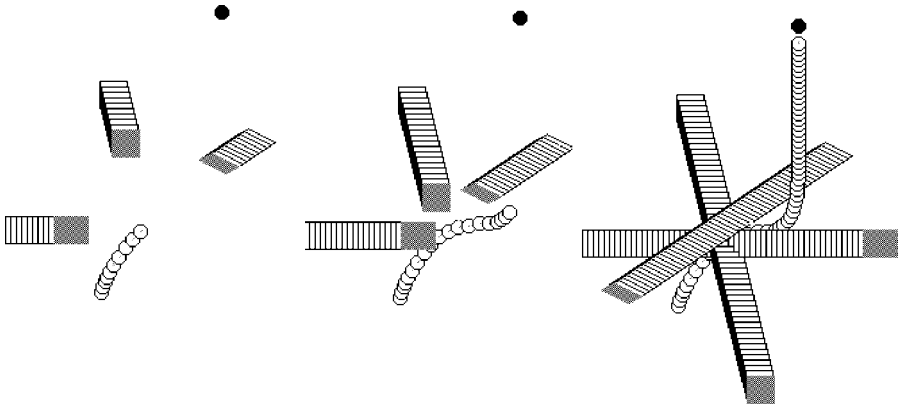


Figure 29. Collision avoidance of a sequence of 3 rapidly converging objects first identified at nearly same instances.

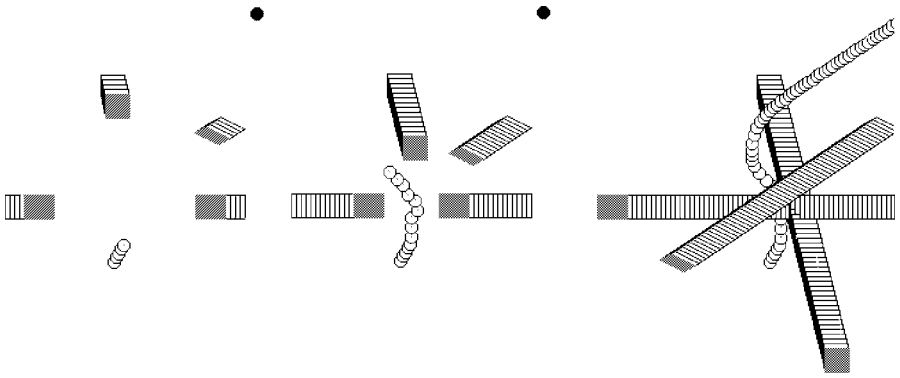


Figure 30. Collision avoidance of a sequence of four converging dynamic objects encountered in rapid succession.

fourth object is detected objects 3 and 4 are considered pairwise for heuristics in case of a potential conflict.

4.2.3. *Avoiding three or more Dynamic Objects from the Priority List*

There are instances however where pairwise consideration of objects for avoidance is not feasible since all the objects get detected in rapid succession and possess similar priorities. In such cases the objects are avoided through the procedure delineated in Section 3.2.4 till the application of pairwise heuristics or the PBA becomes possible. Figure 29 indicates a situation where three dynamic objects that converge radially onto the robot get detected in rapid succession and all of them need to be considered for avoidance. The subsequent parts of the figure show the various snapshots of the path charted by the robot. The various parts of Figure 30 portray the path traced by the robot as it avoids four converging dynamic objects with nearly equal priorities from the list.

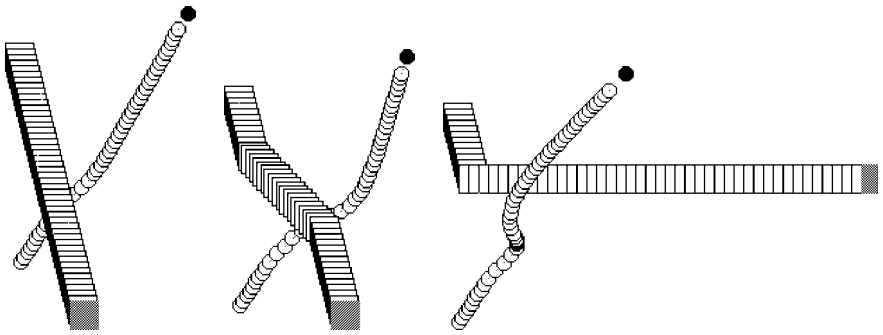


Figure 31. (a) Collision avoidance of an object that does not change its direction and velocity. (b) Object changes its direction by 45 degrees. (c) Object changes direction by 90 degrees and increases its velocity. Robot initially tends to overtake but later slows down.

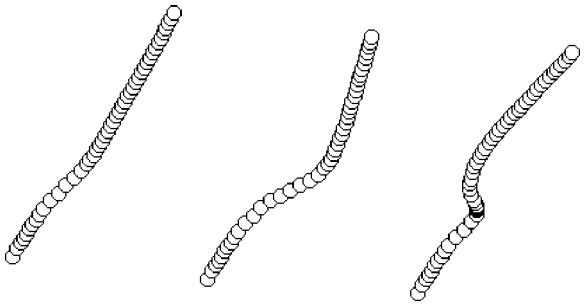


Figure 32. (a)–(c) The path tracked by the robot for the three cases of Figures 31(a)–(c) shown alone in the absence of object paths.

4.2.4. *Changing Direction*

An apprehension regarding the suitability of the proposed method for tackling objects that changes direction and velocity in midstream may arise. It is to be clarified that the proposed method is appropriate for such situations also as the following simulations indicate. Figure 31(a) shows the avoidance of a dynamic object that moves without any change in velocity or direction. Figure 31(b) shows avoidance of the same object as it changes its direction by 45 degrees in midstream. The navigation strategy seems capable for handling such changes in the object's motion. Figure 31(c) shows the graph where the object changes its direction by ninety degrees after the robot had already detected it and actuated the avoidance maneuvers. The object also increases its velocity along with a change in direction. As a result the robot, which had originally adopted an overtaking behavior, switches back to a slowing mode sensing the increase in velocity of the object as well as a change in its direction. Figures 32(a)–(c) highlight the path tracked by the robot by hiding the obstacle paths for the examples of Figures 31(a)–(c). The deviation in the robot's direction in Figure 32(a) when compared with Figure 32(b)

is prominent as it copes to avoid an object that has changed its direction by 45 degrees.

4.3. 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 (i) the point of view of estimation of the robot's egomotion or the localization problem and (ii) the reliability of sensor data. It is argued here that the above methods can be employed legitimately for experimental robots even from the point of view of these objections.

4.3.1. *The Problem of Localization*

Localization problem is the problem of inaccuracy in estimating the robot's own position while map building. Map building procedures project range readings that are with respect to a frame attached to the robot onto a global stationary frame. Since the robot's position varies between any two samples accurate representation on the map requires an accurate estimation of robot's own position. In this paper the accuracy of robot's own position is assumed while computing representations of the external world on a map. The question is could this algorithm be adopted per se on a robot operating in real world situations. The argument to the above question proceeds in two ways.

First, it is argued the problem of localization that arises in a real world environment is not the concern of this endeavor and the approaches presented here for attribute classification should be used in tandem with approaches that overcome localization errors.

Second, adopting the approaches directly onto a real robot that uses simple odometric sensors to detect its own position without a separate strategy for minimizing position errors would still not be a liability. Primarily the approaches presented in this chapter are 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 effort. The changes in robot's position over temporally proximal scans can be estimated to working accuracy based on odometric data itself. Recently Prassler et al. [19] have reported efficient real-time implementations on a similar problem through map building based on odometric data itself.

4.3.2. *Reliability of Sensory Data*

As far as range data is concerned laser range finders have proven to be more precise and reliable than sonar. An 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

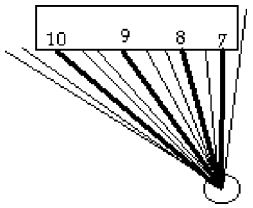


Figure 33. Sensors 7–10 detect the object with increasing ranges. That ray of the cone that is most likely to have detected the object is shown in a dark line. Each sensor is marked by 3 rays, a center ray and the two extremeties of the cone.

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 (Figure 33). These readings can vary in 3 ways. They can monotonically increase from 7 to 10 that indicates the object is closest to 7 or decrease monotonically that indicates it is closest to 10 or they can decrease and increase that indicates that they are closest to a sensor between 7 and 10. For the monotonically increasing case we consider the reading of sensors 10 to 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 33 where each sensor beam is marked by 3 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 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. Other ways of tackling the above problem exist in literature. The method of histogram mapping due to Borenstien [2] is a popular method to circumvent the above problem.

5. Conclusions and Possible Extensions

An algorithm that detects, tracks and avoids multiple moving objects during real-time navigation of a mobile robot has been presented and its efficacy established. The main contribution of the algorithm lies in its ability to classify a given sequence of range data to belong to a static or a dynamic object based on the two classification strategies provided. Most of the algorithms that employ dynamic object avoidance with range sensors in general do not incorporate such a classification scheme, which is probably the obvious first step in collision avoidance. Two new methods MBA and CBA have been presented as efficient scheme for classifying the attribute of an object. The CBA is relevant in an environment where sensor readings are interspersed with outlier-ridden data. It is also stable to rotational movements of the robot apart from its inherent capacity to filter outliers. In an environment where

noisy readings are not expected the MBA would be applicable as it is less intensive computationally than CBA and is unaffected by rotation. Besides this the paper also demonstrates the utility of a fuzzy control strategy for avoiding the various dynamic objects. It also argues regarding the indispensable need for incorporating certain heuristics for efficient collision avoidance for a mere extension of the rules for multiple dynamic objects through PBA exposes the robot to various situations of collision. These situations arise when the rules employed to avoid the first object and those that avoid the second object generate conflicting actions. Apart from this an extension of the collision avoidance strategy to tackle 3 or more dynamic objects encountered in rapid succession such that they cannot be considered pairwise through heuristics has also been formulated. Simulation results show the efficacy of the algorithm to steer past singular and multiple dynamic objects as well as its ability to handle objects that change their velocity and direction.

The future scope of the work is probably multifarious that include considering objects that trace trajectories that are not piecewise linear such as a parabolic trajectory, to cope with unexpected situations such as a dynamic object emerging from behind a static object and an extension to cooperative collision avoidance that involves multiple robots. A further investigation on tracking occluded objects and incorporating such information for enhancing collision avoidance while turning around a bend or a corner by expecting the emergence of the occluded object from behind the corner is also being considered.

References

1. Babuska, R.: *Fuzzy Modeling for Control*, Kluwer Academic, Dordrecht, p. 101.
2. Borenstien, J. and Korem, Y.: The vector field histogram: Fast obstacle avoidance for mobile robots, *IEEE Trans. Robot. Automat.* **7**(3) (1991).
3. Cai, Q., Mitchie, A. and Aggarwal, J. K.: Tracking human motion in an indoor environment, in: *Proc. of Internat. Conf. on Image Processing*.
4. Chang, C. C. and Song, K. T.: Environment prediction for a mobile robot in a dynamic environment, *IEEE Trans. Robot. Automat.* **13**(6) (1997).
5. Fujimura, K. and Samet, H.: A hierarchical strategy for path planning among moving obstacles, *IEEE Trans. Robot. Automat.* **5**, 61–69.
6. Fujimori, A., Teramoto, M., Nikiforuk, P. N. and Gupta, M. M.: Cooperative collision avoidance between multiple mobile robots, *J. Robotic Systems* **17**(7) (2000), 347–363.
7. Griswold, N. C. and Eem, J.: Control for mobile robot in presence of moving objects, *IEEE Trans. Robot. Automat.* **6** (April 1990), 263–268.
8. Hiraga, I. et al.: An acquisition of operator's rules for collision avoidance using fuzzy neural networks, *IEEE Trans. Fuzzy Systems* **3**(3) (1995), 280–287.
9. Horn, B. K. and Schunk, B. G.: Determining optical flow, *Artificial Intelligence* **17**, 185–203.
10. Krishnapuram, R. and Frieg, C. P.: Fitting an unknown number of lines and planes to image data through compatible cluster merging, *Pattern Recognition* **25**(4) (1992), 385–400.
11. Lamadrid, J. F. and Gini, M. L.: Path tracking through uncharted moving obstacles, *IEEE Trans. SMC* **20**(6) (1990), 1408–1422.
12. Lee, P. S. and Wang, L. L.: Collision avoidance by fuzzy logic for AGV navigation, *J. Robotic Systems* **11**(8) (1994), 743–760.

13. Mae et al.: Object tracking in cluttered background based on optical flows and edges, in: *Proc. of the Internat. Conf. on Pattern Recognition*, 1996, pp. 196–200.
14. Madhava Krishna, K., and Kalra, P. K.: Solving the local minima problem for a mobile robot by classification of spatio-temporal sensory sequences, *J. Robotic Systems* **17**(10) (2000), 549–564.
15. Murray, D. and Basu, A.: Motion tracking with an active camera, *IEEE Trans. Pattern Aanal. Mach. Intell.* **16**(5) (1994), 449–459.
16. Nam, Y. S., Lee, B. H., and Kim, M. S.: View time based moving obstacle avoidance using stochastic prediction of obstacle motion, in: *Proc. of IEEE Internat. Conf. on Robot. Automat.*, Minneapolis, MN, 1996, pp. 1081–1086.
17. Papnikolopoulos, N. P., Khosla, P. P., and Kanade, T.: Visual tracking of a moving target by a camera mounted on a robot, *IEEE Trans. Robot. Automat.* **9**(1) (1993), 14-35.
18. Pin, F. G. and Bender, S. R.: Adding memory processing behavior to the fuzzy behaviorist approach: Resolving limit cycle problems in mobile robot navigation, *Intelligent Automat. Soft Comput.* **5**(1) (1999), 31–41.
19. Prassler, E., Scholz, J., and Elfes, A.: Tracking multiple moving objects for real-time robot navigation, *Autonom. Robots* **8** (2000), 105–116.
20. Shih, C. L., Lee, T. and Gruver, W. A.: A unified approach for robot motion planning with moving polyhedral obstacles, *IEEE Trans. Systems Man Cybernet.* **20**(4) (1990), 903–915.
21. Song, K. T. and Chang, C. C.: Reactive navigation in dynamic environment using a multisensor predictor, *IEEE Trans. Systems Man Cybernet.* **29**(6) (1999).
22. Srivastava, P., Satish, S. and Mitra, P.: A distributed fuzzy logic based n-body collision avoidance system, in: *Proc. of the 4th Internat. Symposium on Intelligent Robotic Systems*, January 1998, pp. 166–172.
23. Zhu, Q.: Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation, *IEEE Trans. Robot. Automat.* **7** (June 1991), 390–397.