Abstract—Dynamic environments, containing both static and moving objects, present a great challenge for collision avoidance. Inspired by early processing in the biological systems, the input to the learning system introduced here includes a range map and a motion map. Avoiding computationally expensive real-time models of the dynamic environment, the presented developmental learning method enables the system to learn to quickly identify relevant feature subspaces at every time instance from the real-time input sensory stream. An attention selection mechanism and the Incremental Hierarchical Discriminant Regression (IHDR) were used to identify and switch to different feature subspaces at every time instance. The developmental learning method represents an architecture enabled trade-off among the limited computational power, the real-time requirements and the complex dynamic environments. Real-time simulations are implemented to provide promising results on the avoidance of moving obstacles.

I. INTRODUCTION

A dynamic environment contains static and moving objects. Complex dynamic environments that humans and animals face daily correspond to a great challenge for biological systems and artificial systems alike. The high-dimensional inputs (where each pixel corresponds to a dimension), the change (motion) of some subparts in the inputs, and the requirement for the system to respond in real time are three major operational factors that the system must deal with.

It is known that the biological nervous system does not maintain a real-time monolithic range map of the world. If we consider every neuron as a feature and its response is a value about the degree of the match between an input and the feature, the representation in the biological nervous system consists of hierarchical feature-based response patterns updated in real time [7], [14]. The internal representation of the biological nervous systems is highly distributed and is based on the self-organization of neurons (feature detectors) in the cortex, instead of the geometric 3-D location of the external world. In other words, there is no monolithic 3-D map in the nervous representation. Further, these features are developed from experiences guided by cell-centered genetic developmental principles, instead of totally hard-wired by the genes.

The dynamic features in the input are critical for predicting a potential collision and for retrieving a desired collision-avoiding behavior. However, the dynamic information in the biological system is not maintained through real-time update of a monolithic 3-D map. Instead, the neurons that are sensitive to temporal changes detect motions present in some subparts of the input [2], [10] and these motion sensitive neurons are incorporated with those space feature detection neurons through the cortical self-organization during the development. The model of our system was motivated by these biological characteristics.

Many previous methods were motivated by engineering considerations. The potential fields approach [8] uses resultant force which is the sum of attractive force from goal and repulsive force from obstacles to determine the moving direction and avoid the obstacles. The curvature-velocity [13] and dynamic window methods (DW) [5] formulate obstacle avoidance as a constrained optimization problem in a 2-D velocity space. Obstacles and the robot’s dynamics are considered by restricting the search space to a set of admissible velocities. In order to deal with local minima, limited look-ahead search strategies are integrated. For example, the enhanced VFH algorithm (VFH* [15]) uses A* [12] and the global DW approach [3] uses NF1 [9]. These methods do not have a task-nonspecific developmental lower macro-layer and, thus, are restrictive in the particular environmental settings. [18] presented a developmental model for dealing with collision avoidance in static environments. They showed that a distance-based simple attention mechanism greatly reduced the number of training samples needed to reach a desired performance level.

However, there has been no published work, as far as we know, that deals with collision avoidance in dynamic environments using a developmental learning framework. The work presented in this paper takes up this challenge. We model the early processing of our system to contain two types of feature detector cells, one map containing static distance information and the other map containing motion information. An attention mechanism is incorporated using programmed-in (internal) attention selection behaviors. This attention mechanism avoids, to a large degree, the dimensions of the input that are irrelevant to the desired behaviors. The following IHDR mapping engine simulates cortical mapping
from the response of the attention selection module to the desired motor behaviors. The IHDR engine develops cells (feature detectors) in a hierarchical (tree) structure incrementally, where each layer in the tree structure corresponds to a cortical representation using a set of discriminant feature detectors. By discriminant features, we mean that features are developed automatically based on the supervision of the desired motor output, thus, drastically reduces the dimension of the feature space and greatly improves the generalization power [6], [17]. Although not meant to be perfect for all possible scenarios (e.g., the system does not take barely passable entrances because the system is not taught to), the proposed developmental learning method allows the human to train the system in complex dynamic settings without the need to explicitly design perception and decision making rules which are known to be brittle as demonstrated by many prior studies.

In what follows, we first provide the system architecture for the avoidance of moving obstacles. The detailed approaches for the developmental paradigm are presented in Sec. II. The experimental results and conclusions are reported in Secs. III and IV, respectively.

II. APPROACH

Fig. 1 shows the system architecture of the presented approach. The sensory inputs, \( d(t) \in D \), are obtained from a range scanner by the mobile vehicle, where \( D \) denotes the space of all possible range images in a specific environment. \( d(t) \) is a vector of distance, whose \( i \)th component denotes the distance to the nearest obstacle at a specific angle. The dynamic feature extraction (DFE) module is introduced to extract dynamic information for moving obstacles. In addition, we apply an attentional module to reduce the dimension of the feature space.

The obstacle avoidance behavior is formulated as a direct mapping from the current inputs into actions:  
\[
y(t + 1) = (a(t + 1), v(t + 1)) = f(r(t), m(t)).
\]

where \( r(t) \) is the output vector from attentional module and \( m(t) \) is the dynamic information extracted from the collision time to moving obstacles. The action vector \( y(t) \in \mathcal{Y} \) denotes the action outputs represented by agent’s steering angle \( a(t) \) and speed \( v(t) \). The vehicle learns the mapping from \( r(t) \) and \( m(t) \) to the correct control of speed and steering angle incrementally under the supervision of a human operator. In what follows, we will explain each component of the architecture in details.

A. Dynamic feature extraction (DFE)

Biological motivated by the characteristics that neurons are sensitive to temporal changes to detect motions, we introduce some new features, called Time to Collision, to contain the information of temporal changes of moving obstacles.

\[
m_i(t) = \begin{cases} T \times \frac{r_i(t)}{r_i(t)-r_i(t-1)}, & r_i(t) - r_i(t-1) > \varepsilon > 0 \\ M, & r_i(t) - r_i(t-1) \leq \varepsilon \end{cases}
\]

where \( T \) is the sampling time. \( r(t) = (r_1(t), ... , r_n(t)) \) denotes the input vector from laser scanner after the attention selection. \( \varepsilon \) is a positive parameter to control the range of \( m_i(t) \) and \( M \) is the maximum of time to collision. \( m(t) = (m_1(t), ... , m_i(t), ... , m_n(t)) \) represents the output of dynamic feature extraction.

![Fig. 2. Generalized coordinate system of the car-like vehicles.](image)

Fig. 2 illustrates the global coordinate system of the vehicle agent, where the kinematic model is defined as

\[
\begin{align*}
x_v(t) &= v(t) \cos \theta_v(t) \\
y_v(t) &= v(t) \sin \theta_v(t) \\
\dot{\theta}_v(t) &= \frac{1}{l}v(t) \tan(\alpha(t))
\end{align*}
\]

\((x_v(t), y_v(t))\) is a Cartesian position to the center point of the vehicle rear axis, where \( v(t) \) is the velocity orthogonal to the rear axis and \( \theta_v(t) \) is the orientation of the car body with respect to \( x \) axis. \( l \) denotes the distance between the front and rear axes. \( \alpha(t) \) represents the steering angle of the vehicle.

Assuming a moving obstacle point in global coordinate system is \((x_o(t), y_o(t), v_o(t), \theta_o(t))\), it generates the state
of vehicle’s local coordinate system as following:

\[
\begin{bmatrix}
  x(t) \\
  y(t)
\end{bmatrix} =
\begin{bmatrix}
  \cos \theta_v(t) & \sin \theta_v(t) \\
  -\sin \theta_v(t) & \cos \theta_v(t)
\end{bmatrix}
\begin{bmatrix}
  x_o(t) - x_v(t) \\
  y_o(t) - y_v(t)
\end{bmatrix}
\]

(4)

and thus

\[
\begin{cases}
  \dot{x}(t) = v(t) - v_o(t) \cos(\theta_v(t) - \theta_v) \\
  \dot{y}(t) = v_o(t) \sin(\theta_v(t) - \theta_v)
\end{cases}
\]

(5)

When the obstacle is static, \( v_o(t) = 0, \theta_v(t) = 0, x_o(t) = X_o, y_o(t) = Y_o \) is only determined by the movement of vehicle itself.

Considering that \( r_i(t) \) is the distance to a moving obstacle, we have

\[
r_i(t) = \sqrt{x^2(t) + y^2(t)}
\]

(6)

By applying the relations in Eq. (5),

\[
\dot{r}_i(t) = (x^2(t) + y^2(t))(x_i(t)v(t) - x_i(t)v_o(t))
\times \cos(\theta_v(t) - \theta_v) + y_i(x)v_o(t) \sin(\theta_v(t) - \theta_v)
\]

(7)

Eq. (7) shows that in the same direction \( i \), the difference of \( r_i(t) \) between two frames provides the moving obstacle velocity \( v_o(t) \) and its direction \( \theta_v(t) \). From the kinematics analysis above, therefore, it is concluded that dynamic information, either for moving obstacle or moving vehicle itself, is included in the feature of time to collision \( m(t) \).

B. Attention selection

The appearance-based approach uses monolithic views, where the entire range data (or visual image) frame is treated as a single entity. However, high dimensionality of the input will lead to very slow development. Attention selection aims at extraction of the critical areas around the vehicle and the reduction of the input space. Properly implemented, it will improve both developmental speed and accuracy. The driving experience indicates that human beings pay more attention to the front-view areas bounded by road edges or lane markings. The side view is often neglected. In environments with dynamic obstacles, drivers must pay more attention to the most hazardous obstacles (i.e., potential collisions). Based on these ideas, two kinds of attention selection schemes are introduced in this paper. One is attention area selection, the other is critical distance selection.

An ellipse function is used to define the attention area.

\[
\frac{x^2}{a^2} + \frac{(y - c)^2}{b^2} = 1 \quad (0 < y < R_{\text{max}}, \quad \frac{a}{b} \sqrt{b^2 - c^2} > \frac{W}{2})
\]

(8)

where \( W \) is the width of the vehicle. \( R_{\text{max}} \) is the maximum distance provided by the sensor.

Given a point \( P(x, y) \) on the ellipse, \( e_i \) is the distance from the point to the origin, where the angle with \( x \) axis is \( \gamma_i \)

\[
e_i = \frac{a^2 \cos \gamma_i \pm \sqrt{(b^2 - c^2) \cos^2 \gamma_i + a^2 \sin^2 \gamma_i}}{b^2 \cos^2 \gamma_i + a^2 \sin^2 \gamma_i} \quad e_i > 0
\]

The operation of the attentional effector for input \( d(t) \) and output \( r(t) \) is defined by:

\[
r_i = \min(d_i, e_i)
\]

(9)

where \( r(t) = (r_1(t), ..., r_i(t), ..., r_n(t)) \) is the output vector of the attentional action.

To determine the parameters in Eq. (8), three different methods can be used. (i) Learning from a human operator’s attention decision behavior. (ii) Calculated from a set of linear functions related to the vehicle speed \( v(t) \) (i.e., \( a = k_a v(t) + b_a, b = k_b v(t) + b_b, \) and \( c = k_c v(t) + b_c \). Note that \( a, b, c \) must be first preselected for two different speeds. (iii) Preset them according to some vehicle speed thresholds. For example, a single threshold allows a more narrow area \( (a_f, b_f, c_f) \) to be used at fast speeds and a wider area \( (a_s, b_s, c_s) \) at slow speeds. This was the approach we used in our experiment, presented in Section III.

A caution attention area will take effect based on thresholds on the nearest obstacle distance and time to collision values. This is designed to take effect in critical situations and has precedence over all other attention areas. It is two half circles with radius of \( R_c \) (critical distance) and \( T_c \) (critical time), as formulated in Eq. (10).

\[
\begin{cases}
  r_i(t) = \min(r_i, R_c), \quad r_j(t) < R_c \quad (i, j = 1, ..., n) \\
  m_i(t) = \min(m_i, T_c), \quad m_j(t) < T_c \quad (i, j = 1, ..., n)
\end{cases}
\]

(10)

![Fig. 3. Attention selection mechanism.](image)

Fig. 3 shows these attention selection schemes with different parameters.

It is worth noting that the attention selection should be disabled when the vehicle is approaching an intersection. After the vehicle exits the intersection, the attention selection is re-engaged. Knowledge of being in an intersection comes from an higher level recognition module or planner.

C. Incremental Hierarchical Discriminant Regression

In the context of the appearance-based approach, the mapping (e.g., \( f \) in Eq. (1)) from high dimensional sensory input space \( \mathcal{X} \) into action space \( \mathcal{Y} \) remains a nontrivial problem
in machine learning, particularly in incremental and real-time formulations. By surveying the literature of the function approximation with high dimensional input data, one can identify two classes of approaches: (i) approaches fit global models, typically by approximating a predefined parametrical model using a pre-collected training data set [1], [4], and (ii) approaches fit local models, usually by using temporally-spatially localized simple (therefore computationally efficient) models and growing the complexity automatically (e.g., the number of local models and the hierarchical structure of local models) to account for the nonlinearity and the complexity of the problem [6], [16], [17].

Comparing the two kinds of regression methods, local model learning approaches are more for incremental and real-time learning in high dimension spaces due to their high efficiency. IHDR organizes its local models in a hierarchical way, as shown in Fig. 4, the tree structure recursively excludes many far-away local models from consideration, thus, the time to retrieve and update the tree for each newly arrived data point \( X \) is \( O(\log(n)) \), where \( n \) is the size of the tree or the number of local models. This extremely low time complexity is essential for real-time online learning with a very large memory. Another advantage of IHDR is that it derives automatically discriminating feature subspaces in a coarse-to-fine manner from input space \( X \) in order to generate a decision-tree architecture for realizing self-organization.

In testing, only the obstacle vehicle was controlled. This generated the teaching signals to train the agent vehicle. Obstacle vehicle were directly controlled by the keyboard. Both a developmental vehicle and a non-developmental late the real-time online learning of moving obstacle avoidance. Both a developmental vehicle and a non-developmental obstacle vehicle were directly controlled by the keyboard. This generated the teaching signals to train the agent vehicle. In testing, only the obstacle vehicle was controlled.

Online incremental training processing does not explicitly have separate training and testing phases. The learning process is repeated continuously when the actions are imposed.

**III. EXPERIMENTAL RESULTS**

A virtual driving environment was implemented to simulate the real-time online learning of moving obstacle avoidance. Both a developmental vehicle and a non-developmental obstacle vehicle were directly controlled by the keyboard. This generated the teaching signals to train the agent vehicle. In testing, only the obstacle vehicle was controlled.

Fig.5 shows the interface used in the experiment, where four windows were designed to provide different views during the learning process. The upper left window displays

---

**Fig. 4.** The autonomously developed IHDR tree. Each node corresponds to a local model and covers a certain region of the input space. The higher level node covers a larger region, and may partition into several smaller regions. Each node has its own Most Discriminating Feature (MDF) subspace \( \mathcal{D} \) which decides its child models’ activation level.

We do not intend to formulate IHDR in this paper. Please refer to [17] for a complete presentation. There are two kinds of nodes in IHDR: internal nodes (e.g., the root node) and leaf nodes. Each internal node has \( q \) children, and each child is associated with a discriminating function:

\[
l_i(x) = \frac{1}{2}(x - c_i)^T W_i^{-1} (x - c_i) + \frac{1}{2} \ln(|W_i|),
\]

where \( W_i \) and \( c_i \) denote the distance metric matrix and the x-center of the child node \( i \), respectively, for \( i = 1, 2, ..., q \). Meanwhile, a leaf node, say \( c \), only keep a set of prototypes \( \mathcal{P}_c = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \). The decision boundaries for internal nodes are fixed, and a leaf node may develop into an internal node by spawning once enough prototypes are received.

The need of learning the matrices \( W_i, i = 1, 2, ..., q \) in Eq. (11) and inverting them makes it impossible to define \( W_i \) (for \( i = 1, 2, ..., q \)) directly on the high dimensional space \( X \). Given the empirical observation that the true intrinsic dimensionality of high dimensional data is often very low [11], it is possible to develop most discriminating feature (MDF) subspace \( \mathcal{D} \) to avoid degeneracy and other numerical problems caused by redundancy and irrelevant dimensions of the input data (see Fig.4).

**D. Algorithm**

The learning procedure is outlined as follows:

1) At time frame \( t \), grab a new laser map \( d(t) \).
2) Use Eq. (9) to compute \( e(t) \) by applying attention \( e(t) \) to a given \( d(t) \).
3) Use Eq. (2) to calculate the time to collision \( m(t) \).
4) If the teaching signal is given as an imposed action \( y(t) \), then go to step 5. Otherwise go to step 7.
5) Use input-output pair \( (x(t), y(t)) \) to train the IHDR tree as one incremental step.
6) Send the action \( y(t) \) to the controller which gives the steering angle and speed to vehicle effectors. Increment \( t \) by 1 and go to step 1.
7) Query the IHDR tree to retrieve an action \( y(t+1) \).

Send \( y(t+1) \) to the controller which gives the steering angle and speed to vehicle effectors. Increment \( t \) by 1 and go to step 1.

---
a global map of the driving environment, including road boundaries, moving obstacles and the agent vehicle. The upper right window shows the current range map and displays the output signals of steering angle and velocity. Three parallel green lines are used to denote left track boundary, right track boundary and vehicle’s central line, respectively. The tracks indicate the potential movement of the vehicle by using the current steering angle, where the curvature of the tracks is related to the front wheel angle and the length is dependent on the velocity of the vehicle. The lower left window plots the features of obstacle distances (red) and time to collision (blue). Both features are displayed by the resolution of 0.5°, where the total dimension of the input space is $2 \times 180/0.5 = 720$. The lower right window illustrates the tracks of both the vehicle and a moving obstacle. The same color in both tracks shows the positions of the vehicle and the obstacle at the same time. If the colors are coincident at the crossing point of two tracks, a collision has happened.

The IHDR learning engine uses range maps and time to collision as the input features. The incremental and online property of IHDR achieves the refinement quickly. It switches the mode from testing to learning when the output from the learning machine is not correct or not accurate enough. After the teaching signal is learned by the IHDR machine, the performance will be improved. To show the capability of the regression mechanism, two IHDR trees were trained simultaneously: One is generated for steering angle outputs and the other is used for speed outputs. We interactively trained the simulated robot in 16 scenarios, which acquired 1451 total samples.

In order to test the generalization capability of the learning system, we performed 10-fold cross validation for both IHDR trees. For each test round, let $y_i$ and $\hat{y}_i$ denote, respectively, the true and estimated outputs, and $e_i = y_i - \hat{y}_i$ denotes the error for the $i$th testing sample. We define the mean square error $\sigma_e$ as $\sigma_e = \sqrt{\frac{1}{n} \sum_{i=1}^{n} e_i^2}$ where $n$ is the number of testing samples. The results of cross-validation test for two IHDR trees are shown in Fig. 6.

![Fig. 6. The results of cross validation for IHDR. Red squares denote the mean square error for the estimation of the robot’s orientation. The unit of y-axis is in degree. Blue circles represent the mean square error for the robot’s velocity. The unit of y-axis is the robot’s velocity.](image)

Different scenarios were tested for the learned capabilities for road following, static obstacle avoidance, and moving obstacle avoidance, where the different initial states are set
to make the learning space more general and balanced. After 20 sections of training, we tested the vehicle in the driving environment by the vehicle start point at any initial velocity and direction. The test performance shows that the agent was able to follow the road precisely, to avoid the static obstacle and to slow down when there is a moving obstacle near by. The rate of success was 90% during 30 times of testing, where the success is defined as being able to navigate to the end section in the presence of static and moving objects, without hitting any obstacles. Fig.7 provides three representative scenarios where different moving obstacles are successfully avoided.

IV. DISCUSSION AND CONCLUSION

This paper describes a range-based learning system to avoid static and dynamic obstacles through interactive experiences. The attention mechanism selects regions of critical importance for analysis so that fewer training samples are needed for reach a performance level. The extraction of dynamic features is proved to be effective for the learning of avoiding moving obstacles. The power of the learning-based method, without using explicit world representation, is capable of learning a very complex function between the input range map and the desired behavior. The excessive amount of computational and sensory requirements are not necessary. The success of the learning for high dimensional input \((r, m)\) is mainly due to the power of the general purpose engine IHDR, and the real-time speed is attributed to the logarithmic time complexity of IHDR. The online incremental learning is useful, so the trainer can dynamically select scenarios according to the robots weakness (i.e., problem areas) in performance. It is true that training needs extra effort, but it enables the behaviors to change according to a wide variety of unforeseen changes in the range map.

REFERENCES


