

VII. CONCLUSIONS

The possibilities for implementation of a sliding mode algorithm for training multilayer NN proposed in [12] as an online mechanism for adaptation in closed-loop feedback neurocontrol systems have been investigated. It is confirmed that the algorithm can be used to train the network structures as they interact with the external environment. The learning rule can be generalized for any number of layers and for recurrent networks as well. The network structures trained with the sliding mode learning algorithm are robust and learn fast, both of which are features inherited from SMC. By using the proposed approach we have obtained results that show faster convergence ability and better performance on reducing mapping error in case of online learning neural network structures which can lead to an improvement of the tracking performance of neuro-adaptive systems.

ACKNOWLEDGMENT

The authors would like to acknowledge the fellowship provided to the first author by the Scientific and Technological Research Council of Turkey (TUBITAK) within the framework of the NATO-PC Advanced Fellowship Program.

REFERENCES


Evolving Programming-Based Univerctor Field Navigation Method for Fast Mobile Robots

Yong Jae Kim, Jong Hwan Kim, and Dong Soo Kwon

Abstract—Most of navigation techniques with obstacle avoidance do not consider the robot orientation at the target position. These techniques deal with the robot position only and are independent of its orientation and velocity. To solve these problems this paper proposes a novel univerctor field method for fast mobile robot navigation which introduces a normalized two-dimensional vector field. The method provides fast moving robots with the desired posture at the target position and obstacle avoidance. To obtain the sub-optimal vector field, a function approximator is used and trained by evolutionary programming. Two kinds of vector fields are trained, one for the final posture acquisition and the other for obstacle avoidance. Computer simulations and real experiments are carried out for a fast moving mobile robot to demonstrate the effectiveness of the proposed scheme.

Index Terms—Evolutionary programming, navigation, soccer robots, univerctor field navigation method, wheeled mobile robots.

I. INTRODUCTION

Navigation with obstacle avoidance is one of the key issues to be looked into for successful applications of autonomous mobile robots. Navigation involves three tasks: mapping and modeling the environ-

Manuscript received April 17, 1999; revised February 3, 2001. This paper was recommended by Associate Editor R. A. Hess.

The authors are with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea (e-mail: yjkim@vivaldi.kaist.ac.kr; johkim@vivaldi.kaist.ac.kr).

Publisher Item Identifier $1083–4419/01$05219-0.
ment, path planning and selection, and path following [1]. The path following task is composed of trajectory planning, which makes the generated path a time parameterized line, and generating tracking control. The traditional navigation method [2] separates path planning and path following into two isolated tasks. In the path planning step, a path generation algorithm is developed which does not cross the obstacles and connect the destination with the starting point. In the path following step, tracking problem is the key issue to be studied and there are various approaches such as sliding mode control and feedback linearization control [3]–[5]. On the other hand, in unified navigation such as potential field method [2], [6], [7], these two steps are unified in one task.

Figs. 1(a) and (b) show the conventional and the unified navigations, respectively. Using the conventional navigation, the processor interprets the environment, computes the path and then controls the robot to follow the path. When the robot goes off from the given path, it should return immediately as indicated by $\alpha$. While doing so, the obstacle avoidance behavior and the effectiveness of such a path are not guaranteed. This is due to the path following process which does not consider the environment information. As an extreme case, a robot at the point $g$ can collide with the obstacle when it follows path $c$ to return to the computed path $I$. Also conventional navigation methods do not consider the robot orientation at the target position. For instance, when a robot dribbles a ball in a robot soccer game [8]–[10] or pushes a load in an industrial field, it is very important that the robot acquires the final robot orientation. Using conventional methods, achieving such tasks are difficult. Moreover, in the path planning step, the generated path ignores the mechanical properties of the robot. On the other hand, in unified navigation, the processor computes the desired direction and control input at every moment and the associated path $b$ need not to be fully determined a priori.

To control fast mobile robots, a simple controller is required, which satisfies the mechanical properties such as limitations of wheel speed or translational speed of the robot center. Efficiency of trajectories and short navigation time are also to be ensured. These are important issues of soccer robots also. Added to this the robot soccer system has a dynamic environment with moving obstacles and moving targets which needs special attention [10]. Considering dribbling and kicking action in robot soccer, robot posture (position and orientation) is of utmost importance. Here the paper aims to address the specific problem of robot posture at the target position with emphasis on optimizing the navigational path of the robot.

In the early stages of robot soccer, traditional navigation methods were popular where the option was to use the simple shortest paths or Dubin’s path [11] or the composition of rotation, circular motion and straight motion for the path planning step [9], [12]. Recently research interest is being focused on the application of fuzzy logic, evolutionary computation, reinforcement learning, unified navigation method and so on [13]–[16].

The proposed navigation method is an improved unified navigation method which is designed for fast mobile robots considering kinematic restrictions. Using the navigation method the robot can navigate rapidly to the desired posture (position and orientation) without oscillations or unwanted inefficient motions. A novel vector field, namely the univector field which is a normalized two-dimensional (2-D) vector field where the vector is designed as a unit vector is proposed. To obtain the suboptimal univector field for navigation, a function approximator and its learning algorithm are proposed based on evolutionary programming (EP) which takes into consideration the kinematic properties of the robot. Two kinds of univector fields are trained. One is concerned about the final posture of the robot and hence deals with the field for the desired orientation. This takes the robot to a desired posture. The other is the field for obstacle avoidance. Combining these two well-trained fields, the complete field for an environment with obstacles can be generated. In this field, the robot can move to the desired position and orientation without collision. By adopting the univector field, both the performance and the obstacle avoidance capability of the unified navigation approach are improved. The proposed navigation method is implemented on a differential drive mobile robot designed for MiroSot (http://www.fira.net).

In Section II, the kinematic properties of differential-drive mobile robots are discussed. In Section III, the definition of univector field and the concept of univector field navigation are described. The learning algorithm and univector field tracking controller are also presented. Sections IV and V report simulations and experimental results, respectively. Concluding remarks follow in Section VI.

II. MODEL OF A MOBILE ROBOT

In this paper, a nonslipping and pure rolling differential-drive mobile robot is considered [17]. It is assumed that the posture (position and orientation) of the robot is known at each instant. The experimental setup is described in Section V. The mechanical structure of the mobile robot is shown in Fig. 2(a), where $L$ is the base width of the robot and $R$ is the radius of the wheel.

The kinematics of the robot can be described using Eq. 2(b). Posture $p_x$ and position $q$ of the robot are defined as

$$p_x = \begin{bmatrix} x_c \\ y_c \\ \theta_c \end{bmatrix}, \quad q = \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

where $(x_c, y_c)$ is the position of the center of robot, and $\theta_c$ is the heading angle of the robot with respect to absolute coordinates $(x, y)$. Velocity vector $s$ is defined as follows:

$$s = \begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{\begin{bmatrix} V_L + V_R \\ V_R - V_L \end{bmatrix}}{L} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix}$$

where $v$ is the translational velocity of the center of robot and $\omega$ is the angular velocity with respect to the center of robot. Equation (2) gives the relation between the velocity vector and the velocities of two wheels, $V_L$ and $V_R$, where $V_L$ is the left wheel velocity and $V_R$ is the right wheel velocity.

![Fig. 1. Conventional and unified navigations.](image-url)
In this paper, we propose a univector field in which the magnitude attractive force from the desired position and a repulsive force from the robot. (b) Robot modeling.

The robot kinematics associated with the Jacobian matrix and velocity vector is defined as

\[ \dot{p} = \begin{bmatrix} \cos \theta_e & 0 \\ \sin \theta_e & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = J(\theta_e) \dot{\theta}_e \]  

(3)

To get the robot position and orientation, (3) should satisfy the following nonholonomic constraint:

\[ x_e \sin \theta_e - y_e \cos \theta_e = 0 \]  

(4)

which is equivalent to \( \frac{dy_e}{dx_e} = \tan \theta_e \), meaning that the moving direction at every instant is the same as the heading angle of robots. It implies pure rolling and nonslipping as assumed.

The restrictions for the mobile robots are given as follows:

\[ |v| \leq v_{m_e} \]

\[ |V_L| \leq V_{m_w} \]

\[ |V_R| \leq V_{m_w} \]  

(5)

where \( v_{m_e} \) is the maximum speed of the robot center and \( V_{m_w} \) is the maximum speed of the wheels. The robot kinematics and the associated constraints given in (5) are taken into consideration in the simulations and experiments.

### III. UNIVECTOR FIELD NAVIGATION METHOD

Potential field method is a unified navigation method which is generally used in robot control. As it is very simple, it is possible to control robots in real time. However, it may not be possible to maintain the required velocity. Also when the obstacle is big, the robot is liable to break into oscillation. Moreover its heading direction at an arriving position cannot be guaranteed [18]. In potential field navigation the robot moves in a direction proportional to a resultant force comprised of an attractive force from the desired position and a repulsive force from the obstacle to be avoided. This may result in inefficient trajectories.

Better control of the robot is possible with a modified vector field. In this paper, we propose a univector field in which the magnitude of vectors is unity at all the positions. The modified vector field not only yields better navigation performance with obstacle avoidance capability but also provides the robot with the desired posture at the target position.

#### A. Univector Field Generation

A univector field \( N \) for the robot navigation is defined as

\[ N: F \rightarrow \mathbb{R} \]  

(6)

where \( F \) is the workspace of the robot in \( \mathbb{R}^2 \) and \( I \) is a set of unit vectors with arbitrary angles. It is assumed that the magnitude of vectors in the field is unity at all points. As normalized vectors are used, the univector field \( N \) can be represented in terms of its angles, as follows:

\[ \phi: F \rightarrow [-\pi, \pi] \]  

(7)

While controlling the robot, these unit vectors correspond to the desired robot heading directions. The control law is described in detail in Section III-C. Fig. 3 illustrates the univector field for a desired posture at a point \( g \) with the final orientation to the right. The tiny straight lines with a point on one end, depicts the univector fields; the direction being away from the point. The five trajectories, shown by closely packed small squares represent the trajectories of the robot for five arbitrary initial postures. The univector field at a position \( p \) is defined as \( N(p) \). The angle \( \phi(p) \) of the vector at a robot position \( p \) is generated by

\[ \phi(p) = \angle \hat{p} \hat{g} - n\alpha \]  

(8)

with

\[ \alpha = \angle \hat{p} \hat{r} - \angle \hat{g} \hat{r} \]

where

- \( n \) positive constant;
- \( g \) target point;
- \( r \) determines the desired final orientation.

The symbol \( \angle \) denotes the angle of the vector. The field vector and the turning motion of the robots vary in accordance with the parameter \( n \) and the distance between the two points \( g \) and \( r \). Using (8), we can obtain a univector field for the desired posture at a point \( g \). As the robot approaches the target point \( g \) along the calculated vector field, the angle \( \phi \) becomes zero. Thenceforth the robot moves straight toward the target point \( g \). This univector field method was implemented to the robot soccer system [10] for kicking motion, where the point \( g \) was the ball position and the heading position \( r \) was adjusted to the desired kicking direction. The distance between the two points \( g \) and \( r \) was adjusted heuristically to generate the univector field.

Fig. 3. Heuristic univector field for final posture at a point \( g \).
As shown in Fig. 3, in this heuristic univector field method, some inefficient paths are generated. For example, the robot at a point \( d \) on the right of the goal point moves directly to the goal from the starting point \( d \), and then near the goal position it makes a circular motion to acquire the final orientation. The path is unnecessarily long, and moreover, the trajectory of the circular motion is not optimized.

To exploit the univector field \( N \) for robot control with better performance, the field has to be modified. For this purpose a grid net and a function approximator are developed. To start with, in general, a grid of size \( n \times m \) is located within the workspace as shown in Fig. 4(a). The shape and density of the grid net can be varied in accordance with the application and the desired accuracy. A node represents the point of intersection of the grid lines. More dense grid implies larger number of nodes. \( p_{i,j} \) is the position of node \((i, j)\) and \( N_{i,j} \) represents the field vector at \( p_{i,j} \).

The set of angles of univectors \( N_{i,j} \) forms an \( n \times m \) matrix, which is defined as univector field matrix \( \Phi \) as follows:

\[
\Phi = \{ \phi_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m \}
\]  

where \( \phi_{i,j} \) is the angle of vector \( N_{i,j} \). To determine the field vector at an arbitrary position \( p \), interpolating operation is adopted. At first the operator finds the four neighboring nodes at positions \( p_{i,j} \), \( p_{i,j+1} \), \( p_{i+1,j} \), and \( p_{i+1,j+1} \) surrounding the point \( p \). Then as shown in Fig. 4(b) the distances \( d_a, d_b, d_c, \) and \( d_d \) between these four nodes and \( p \) are computed. Interpolated field vector \( N(p) \) and its angle \( \phi(p) \) at \( p \) are calculated as follows:

\[
N(p) = \frac{\|N\|}{\|N\|}
\]

\[
\phi(p) = \angle N(p)
\]

with the equations shown at the bottom of the page. \( N_{i,j} \) at any node \((i, j)\) is known beforehand. Thus \( N(p) \) can be evaluated. \( N(p) \) represents an intermediate vector for \( N_{i,j}, N_{i,j+1}, N_{i+1,j}, \) and \( N_{i+1,j+1} \) vectors. As \( p \) approaches \( p_{i,j} \), \( N(p) \) converges to \( N_{i,j} \). Thus, by setting the elements of the matrix \( \{N_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m \} \) to each of the node values, all the vectors in the field \( N \) can be fully determined.

For example, consider the following \( 3 \times 3 \) univector field matrix:

\[
\Phi = \frac{\pi}{2} \begin{bmatrix}
\frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{4} \\
\frac{3\pi}{2} & \frac{\pi}{2} & 0 \\
\frac{\pi}{2} & \frac{\pi}{6} & \frac{\pi}{4}
\end{bmatrix}
\]  

Fig. 5(a) shows the field vectors represented by the matrix \( \Phi \) and Fig. 5(b) is the univector field calculated using (10). In the next section, the training of the vector field \( N(p) \) is discussed.

B. Evolutionary Programming for the Grid Net

To control a fast mobile robot, many conditions which are difficult to represent are to be satisfied. Researchers focus on some of them, based on their interest and the field of application. To optimize such a complex system, evolutionary programming (EP) is an efficient tool.

Evolutionary algorithm is a probabilistic algorithm which maintains a population of individuals. In any evolutionary algorithm, each individual represents a potential solution to the problem at hand and is implemented as some data structure (chromosome). Each solution is evaluated to give some measure of fitness. Then, a new population is formed by selecting the more fit individuals. Some members of the new population undergo transformations such as mutation and crossover to form new solutions. After repeating these procedures the best individual converges to a sub-optimal solution. Evolutionary programming is a kind of evolutionary algorithm. In evolutionary programming the chromosome need not be represented by bit-strings and the transformation process includes additional operators appropriate for the given

---

Fig. 4. Grid net of the function approximator.

---

Fig. 5. Simple example of grid net.
structure and the given problem. To train the univector field, the univector field matrix $\Phi$ is used as a chromosome. For a detailed review, the reader is referred to [19].

In this paper, the evaluation function is decided based on the elapsed time, the orientation error, the positioning error, the distance from the obstacle and the maximal angular acceleration.

These criteria are merged to form an evaluation function $f(t, \theta, p, \omega)$ for each followed path as follows:

$$f(t, \theta, p, \omega) = k_1 t + k_2 |\theta(t) - \theta_d| + f_1(p) + f_2(p) + f_3(\omega)$$

(12)

where $t$ is the elapsed time and $\theta_d$ is the desired final orientation. This evaluation function is used for training the univector field matrix $\{N_i \mid 1 \leq i \leq n, 1 \leq j \leq m\}$.

The first term in the evaluation function is to ensure quick reachability of the target point and the second term forces the robot to converge to the desired final orientation $\theta_d$. The third term $f_1(p)$ makes the robot move to the target position

$$f_1(p) = \begin{cases} 0, & \text{if arrived at } p_d \text{ within allowable error bound} \\ T_p + \max_{t \in [0, t]} |p(t) - p_d|, & \text{otherwise} \end{cases}$$

(13)

where

- $p(t)$ position of the robot center at time $t$;
- $p_d$ target position;
- $T_p$ penalty value that is added when the robot does not arrive at $p_d$.

If the robot does not reach the target position, as indicated in (13), the distance from the robot center to the target point, the corresponding value $\min_{t \in [0, t]} |p(t) - p_d|$ and $T_p$ are used to obtain $f_1(p)$. The fourth term $f_2(p)$ prevents the robot from colliding with an obstacle and assumes the values

$$f_2(p) = \begin{cases} 0, & \text{no obstacle collision} \\ B_p + \max_{t \in [0, t]} |p(t) - p_d|, & \text{otherwise} \end{cases}$$

(14)

where

- $B_p$ penalty value;
- $\Omega \subset [0, t]$ time interval during which the robot is within an obstacle boundary;
- $p_6$ closest point on the obstacle boundary from the robot center.

When the robot collides with an obstacle, the function $f_2(p)$ is calculated by projecting the robot trajectory nearest to the obstacle center. The shortest distance of such a point from the periphery of the obstacle is used for getting the value of the function $f_2(p)$. The last term $f_3(\omega)$ makes the robot angular acceleration, $\omega$, not to exceed its limit $\alpha_{\text{max}}$:

$$f_3(\omega) = \begin{cases} 0, & \text{when } \omega \text{ is within the limit } \alpha_{\text{max}} \\ A_p + \max_{t \in [0, t]} |\omega(t) - \alpha_{\text{max}}|, & \text{otherwise} \end{cases}$$

(15)

In computer simulation, the scaling factor $k_1$ and $k_2$ are taken as one and five, respectively. The penalty values $T_p$, $B_p$, and $A_p$ are taken as 500 cm, 100 cm, and 50 rad/s, respectively. The value of $T_p$ is set to be greater than the sum of the other two terms $B_p$ and $A_p$ in evaluation function. The terms $f_1(p)$, $f_2(p)$ and $f_3(\omega)$ are made to satisfy the constraints, which ensure that the robot reach the target position without collisions and have a motion without ripples. The remaining terms $k_1 t$ and $k_2 |\theta(t) - \theta_d|$ of (12) are used as fine tuning for short navigation time and desired final orientation. Once the univectors are trained properly, the values of $f_1(p)$, $f_2(p)$ and $f_3(\omega)$ are all zero.

### TABLE I

<table>
<thead>
<tr>
<th>EP ALGORITHM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Initialization</td>
</tr>
<tr>
<td>(a) $i \leftarrow 0$</td>
</tr>
<tr>
<td>(b) Initialize population</td>
</tr>
<tr>
<td>2. While (not termination condition) do</td>
</tr>
<tr>
<td>(a) $i \leftarrow i + 1$</td>
</tr>
<tr>
<td>(b) $j \leftarrow 0$</td>
</tr>
<tr>
<td>(c) While (not $j = n_p$) do</td>
</tr>
<tr>
<td>i. $j \leftarrow j + 1$</td>
</tr>
<tr>
<td>ii. Mutate $j$th univector field matrix</td>
</tr>
<tr>
<td>iii. $F(j) \leftarrow 0$</td>
</tr>
<tr>
<td>iv. $k \leftarrow 0$</td>
</tr>
<tr>
<td>v. While (not $k = n_p$) do</td>
</tr>
<tr>
<td>A. $k \leftarrow k + 1$</td>
</tr>
<tr>
<td>B. Simulate the robot navigation</td>
</tr>
<tr>
<td>C. Calculate evaluation function $f$</td>
</tr>
<tr>
<td>D. $F(j) \leftarrow F(j) + f$</td>
</tr>
<tr>
<td>(d) Select the best candidates using $F(i)$</td>
</tr>
<tr>
<td>3. End</td>
</tr>
</tbody>
</table>

The penalty values $T_p$, $B_p$, and $A_p$ determine the properties of optimization progress. At the first stage of training, the individuals that fail to drive the robot to the target position are weeded out because $T_p$ is the strongest penalty. But if all the individuals violate this constraint, the term on the right-hand side of $T_p$ in (13) becomes meaningful. By this term the individuals are inclined to approach the target position. The terms $B_p$ and $A_p$ in (14) and (15) are used in a similar manner. The penalty values of $T_p$, $B_p$, and $A_p$ can be varied to suit the designer’s intentions.

The EP algorithm using the evaluation function in (12) is summarized in Table I. In Table I, $n_p$ is the number of individuals (population) and $n_s$ is the number of simulations per individual. The evaluation values for each individual are stored in $F(i)$, which is used to select the best individual. Different termination conditions can be used. In this paper the optimization is terminated if the total number of generations exceeds a predefined one. By this algorithm, the suboptimal univector field matrix can be obtained.

For mutation, the following self-adaptive Gaussian mutation [20] which is widely applied in optimization problems has been used

$$\sigma_{ij} = \sigma_{ij} \exp(\tau' G(0, 1) + \tau G_{ij}(0, 1))$$

$$m_{ij} = \sigma_{ij} G_{ij}(0, 1)$$

$$\tau' = \frac{1}{2 \sqrt{K_v}}$$

(16)

Here $K_v$ is the number of variables in each chromosome. $G(0, 1)$ is a random variable of normal probability distribution whose mean and variance are 0 and 1, respectively. The global factor $\exp(\tau' G(0, 1))$ allows an overall change in the mutability and guarantees the preservation of all degrees of freedom, whereas $\exp(\tau G_{ij}(0, 1))$ allows individual changes with a mean step size of $\sigma_{ij}$. The univector field matrix is updated using the following equation:

$$P_{ij} = \phi(P_{i+1}) + k m_{ij, j} + \frac{1 - k}{4} \cdot (m_{i, j}, j + m_{i, j-1} + m_{i+1, j} + m_{i, j+1})$$

(17)
where $0 \leq k < 1$. The smoothing coefficient $k$ suppresses the ripples in the fields. For details on constrained optimization by evolutionary programming, the reader is referred to [21].

C. Univector Field Tracking Controller

To apply the univector field method for navigation, a field tracking controller is required. The control inputs to the wheels reduce the error in angle between the robot heading direction and the field vector. The error in angle $\theta_e$, between the robot heading angle $\theta_e$ and the vector orientation $\phi$ is given as follows:

$$\theta_e = \hat{\theta}_e - \phi.$$  

(18)

The derivative of $\theta_e$ is

$$\dot{\theta}_e = \omega - \dot{\phi}.$$  

(19)

By (3), $\dot{\phi}$ is expressed as

$$\dot{\phi} = \frac{\partial \phi}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial \phi}{\partial y} \frac{\partial y}{\partial t}$$

$$= \frac{\partial \phi}{\partial x_c} \cos \theta_e x + \frac{\partial \phi}{\partial y_c} \sin \theta_e y$$

$$= G(x_c, y_c, \theta_e)v$$

(20)

where

$$G(x_c, y_c, \theta_e) = \frac{\partial \phi}{\partial x_c} \cos \theta_e + \frac{\partial \phi}{\partial y_c} \sin \theta_e.$$  

The following control input $\omega$ and $v$ are being used

$$\omega = G(x_c, y_c, \theta_e)v - K_w \text{sgn}(\theta_e)\sqrt{|\theta_e|}$$

$$v = \begin{cases} v_{m,c}, & \text{if } |p_x - p_s| > D_s \\ \frac{|p_x - p_s|}{D_s} v_{m,c}, & \text{otherwise} \end{cases}$$

(21)

where

$K_w$ and $D_s$ positive constants;

$p_s$ target position;

$\text{sgn}$ sign function.

From (18) and (21),

$$\theta_e = -K_w \text{sgn}(\theta_e)\sqrt{|\theta_e|}.$$  

(22)

Then, $\theta_e$ will become zero within a time $T \geq (2\sqrt{|\theta_e(0)|}/K_w)$ [22].

Equation (21) represents a kind of sliding mode controller.

IV. COMPUTER SIMULATIONS

To illustrate the efficiency of the proposed univector field for posture acquisition and obstacle avoidance, computer simulations were carried out on a Pentium IBM PC. For each individual, the simulation was carried out 25 times with uniformly distributed random starting postures. Throughout the simulations, the elitist $(\mu + \lambda) - E P$ selection method [19] was used with grid nets of size $6 \times 10$ on upper half (the lower half is symmetric with the upper half). The maximum wheel velocity of the robot $v_{m,c}$, the maximum velocity of robot center $v_{m,c}$ and the maximum angular acceleration of robot $\alpha_{max}$ were 100 cm/s, 50 cm/s and 10 rad/s$^2$, respectively. The parameter $K_w$ of (21) was ten and the number of individuals was 20 with an equal number of offsprings.

A. Univector Field for Final Posture Acquisition

To obtain the desired posture at the target position, the robot must converge to the desired point with its heading angle to the desired orientation. It was assumed that the final position is the center of the field (0,0) and the final orientation is to the right (0 rad) as shown in Figs. 6(a) and (b). Since the presence of any obstacles are not considered, the fourth term $f_e(p)$ in the evaluation function (12) was not used. Each simulation was stopped when the robot reached an allowable error bound of the target point or the time exceeded a maximum limit. As the vector at the target position cannot be defined, the controller set $v$ to zero to prevent any oscillation at the goal position. Figs. 6(a) and (b) show the gradual generation of best univector field.

Fig. 6(a) shows the situation when all the constraints are not satisfied. It is observed that the robots move to the target position but the desired orientation at the target position is not achieved. A comparison with Fig. 6(b) shows that the trajectories have ripples. On the other hand, after 500 generations, when all the constraints are satisfied, the target is reached with the desired orientation at the target point. This is shown in Fig. 6(b).

B. Univector Field for Obstacle Avoidance

Figs. 7(a) and (b) show the results for circular obstacle avoidance. The direction of motions of the robot is to the right border of each frame. Since the second term $K_w |\theta_e(t_s) - \theta_d|$ in the evaluation function (12) is concerned about heading angle at the target position, this term was not used. Each simulation was stopped when the robot crossed over the right border or the time exceeded a maximum limit. As the generation goes on, the navigation trajectory approaches most desired path.
C. Navigation Using the Trained univector Fields

Combining these two well-trained fields in Section IV-A and IV-B, the complete field for an environment with obstacles can be generated. At first, the univector field for final posture is set on the desired target position. Then univector field for obstacle avoidance is applied at each obstacle. Following simulations and experiments are executed using this combined univector field.

Figs. 8(a) and (b) show the difference in performance between the univector field navigation method and the conventional potential field navigation method. The target position is $y$ and the desired orientation is toward the left. The arrows at the target position show the final orientation of robots. Inherently, the potential field method cannot be applied to nonholonomic mobile robots without any modification. To make qualitative comparisons the field tracking controller, presented in Section III-C, was introduced on the potential field method. Figs. 8(a) and (b) highlight the advantages of the proposed univector field navigation method over the potential field navigation method. One of the
The other advantage is the robot motion near the obstacle. In the potential field method, the robot receives the repulsive force in the vicinity of the obstacle and then changes the course abruptly to avoid it. This results in a trajectory that maintains a more than required distance from the obstacle. Also the desired final posture is not attained. If the potential field parameters are changed to obtain better performance, the result is that the robot avoids the obstacle at an unnecessarily far distance from the obstacle and yields inefficient trajectories. On the other hand, the univector field navigation method generates a more effective path and results in correct final posture.

To apply the proposed navigation method to soccer robots, an improvement for dealing with moving obstacles and targets is required. Navigation in dynamic environment can be interpreted as an intercept problem. To solve it researchers study the estimation of moving environments. For an example of tackling the intercept problem in a robot soccer system, refer to [10]. For the kicking motion, the robot must have a preassigned fixed velocity at the target position. This can be achieved by having the control input $v$ in (21) to be the fixed value $v_{m}$. But it weakens the accuracy because stabilization at the target position does not become effective. An empirical modification of the field may overcome this drawback. This aspect need to be explored further before any concrete claims can be made.

V. EXPERIMENTS

To demonstrate the effectiveness of the proposed scheme, it was implemented in the real robot system. To apply the simulated results to the real robot system, the two trained univector fields for final posture acquisition and obstacle avoidance were combined. The overall system in Fig. 10 is composed of a robot, a host computer, a vision system, and a communication system. The overhead vision system detects the position and orientation of the robot and obstacle. Using this vision information, the host computer applies the proposed navigation method to suitably combine the two univectors, which is used to calculate the velocities of the robot wheels. The calculated wheel velocities are transmitted to the robot through the communication system.

The vision system is composed of a TMC-7 CCD camera with a resolution of $320 \times 240$ pixel and an image grabber with a processing rate of 30 frames/s. Due to noise in the vision system, there exist measurement errors in position and orientation. The vision system in the

Fig. 12. Experimental results with and without an obstacle.
experimental setup has measurement errors of about 2.4 cm for position and 4.83° for angle calculations. The host computer is a Pentium processor with 133 MHz clock. Software for implementing the control algorithm is developed in Visual C++ programming language. The communication system has a RF (Radio Frequency) module running at a speed of 9600 bps.

Fig. 11 shows the image of the mobile robot named MIRO which is developed for the purpose of playing the MiroSot robot-soccer game [10]. The robot size is 7.5 cm × 7.5 cm × 7.5 cm, with a base width of 6.5 cm. To keep the balance in motion, the robot has two caster wheels at the bottom—one in the front and the other in the rear. The robot has an AT89C52 microcontroller, two DC motors, and two LM629 motion controllers.

In the experiment, a sampling time of 33 ms was used. Other conditions were the same as in computer simulations. Fig. 12(a) shows the case of final posture acquisition without an obstacle. Figs. 12(b)–(d) show the results obtained with an obstacle, where the center positions of obstacles have a radius of 6 cm are (−7 cm, 20 cm), (25 cm, −36 cm), and (0 cm, 20 cm), respectively. The arrows in Fig. 12 show the final orientation of the robot which agrees with the desired one. The controller constrained the maximum velocity of the robot in each experiment to 50 cm/s. The average velocities of the robot motion in Figs. 12(a)–(d) were 46.0 cm/s, 43.5 cm/s, 45.2 cm/s, and 43.2 cm/s, respectively. These velocities were computed by considering the time taken and the distance traversed by the robot to enter a circular region of diameter 2.4 cm centered at the target position. When the robot enters this circular region, it stops navigation. The figures show that the robots move to the target position with the desired heading angle by the shorter and smoother path with obstacle avoidance capability. Fig. 12 shows good performance for all cases.

The proposed method has been applied only to static obstacles. Hence, in its present form it cannot be applied to robot soccer. Therefore the next step is to explore the possibilities of modifying the univector field to suit a dynamic environment comprising of many obstacles and moving target. Subsequent research will be investigate this possibility.

VI. CONCLUSIONS

A novel navigation method for final posture acquisition and obstacle avoidance was developed for fast moving mobile robots. The method was the result of introducing a modified univector field into the unified navigation method. To obtain the suboptimal vector field, a grid net, a function approximator and its learning algorithm were proposed. The developed navigation method was implemented on a differential-drive mobile robot. Both simulations and experimental results show that the proposed method successfully navigates fast moving mobile robots through efficient trajectories and results in desired final posture of the robots.

REFERENCES