Hidden Markov Model for Visual Guidance of Robot Motion in Dynamic Environment

Qiuming Zhu
University of Nebraska at Omaha, qzhu@unomaha.edu

Follow this and additional works at: http://digitalcommons.unomaha.edu/compscifacpub

Part of the Computer Sciences Commons

Recommended Citation
http://digitalcommons.unomaha.edu/compscifacpub/40

This Article is brought to you for free and open access by the Department of Computer Science at DigitalCommons@UNO. It has been accepted for inclusion in Computer Science Faculty Publications by an authorized administrator of DigitalCommons@UNO. For more information, please contact unodigitalcommons@unomaha.edu.
Abstract—Models and control strategies for dynamic obstacle avoidance in visual guidance of mobile robot are presented. Characteristics that distinguish the visual computation and motion-control requirements in dynamic environments from that in static environments are discussed. Objectives of the vision and motion planning are formulated as: 1) finding a collision-free trajectory that takes account of any possible motions of obstacles in the local environment; 2) such a trajectory should be consistent with a global goal or plan of the motion; and 3) the robot should move at as high a speed as possible, subject to its kinematic constraints. A stochastic motion-control algorithm based on a hidden Markov model (HMM) is developed. Obstacle motion prediction is illustrated in Section IV. Section VIII presents the trajectory-guided path-planning algorithm. Computer simulation and performance evaluation are presented in Section VI. Section VII contains concluding remarks.

Hidden Markov Model for Dynamic Obstacle Avoidance of Mobile Robot Navigation

Qiuming Zhu

A. Global and Local Path Planning

Motion planning for mobile robots and autonomous vehicles in dynamic environments is conducted in two processes: 1) global path planning, which is aimed at the accomplishment of the mission to be carried out by the robot, and 2) local path planning, which focuses on the finding of a collision-free path within the sight of the robot. Global path planning is performed in terms of the task requirement and a static view of the environment. It is called task-level planning. The result of this planning can be just an outline of the route or a set of subgoal points to be followed by the robot. It does not deal with the details of the motion trajectory and does not foresee any variations of the environment. Dynamic obstacle detection and
collision avoidance are dealt with in the local path-planning process. Such a planning process emphasizes the safety of the motion. A collision-free trajectory within the sight of the robot is identified in this process. This work addresses on the local path planning.

While local path planning focuses on collision avoidance, it should also consider the global goal of the motion. The objectives of local path planning can then be stated as:

1) finding a collision-free trajectory that takes account of any possible motions of the obstacles in the local environment;
2) such a trajectory should be consistent with the global goal of the motion; and
3) the robot should move at as fast a speed as possible, subject to its kinematic constraints.

B. Motion State Description

The local path planning of mobile robots in dynamic environments is modeled as a discrete-time process. A control cycle starts at the moment that a sample (an image) of a local scene is acquired by a visual sensing device. The image is processed immediately by a visual information processing unit. Any potential obstacles in the environment are detected. Predictions of the obstacle motion can then be made. A collision-free path is searched for according to the objectives.

The motion state of a robot at a time instance $t$ is represented by

$$\{p_\text{s}(t), v_\text{s}(t), a_\text{s}(t)\}$$

where $p_\text{s}(t)$ denotes the position of the robot, $v_\text{s}(t)$ is the velocity, and $a_\text{s}(t)$ the acceleration of the robot at time $t$. We use $s_\text{s}(t)$ to denote the path actually travelled by the robot in the time interval $[t, t + \Delta t]$, where $\Delta t$ is the control cycle. The motion of the robot can be described as

$$s_\text{s}(t) = p_\text{s}(t + \Delta t) - p_\text{s}(t) = \int_t^{t + \Delta t} v_\text{s}(\tau) \, d\tau$$

(2)

where

$$v_\text{s}(\tau) = v_\text{s}(t) + \int_t^\tau a_\text{s}(t) \, dt.$$  (3)

We use $T_\text{C}(t)$ to denote the vision and control processing time of the robot. $T_\text{C}(t)$ includes the time spent by the robot to process the scene image, detect the moving obstacles, and identify a collision-free path. This $T_\text{C}(t)$ mainly determines, or constrains, the motion speed of the robot. Basically, we should have

$$T_\text{C}(t) < \Delta t.$$  

The notation $\Delta t$ indicates that the control cycle is also a function of $t$. Let $T_\text{D}(t)$ denote the time delay incurred due to the kinematics of the robot; we then have

$$\Delta t = T_\text{C}(t) + T_\text{D}(t).$$  

C. Safety Motion Principle

The central problem in local path planning is to explore the values of $v_\text{s}(\tau)$ in (6). The values are affected by the presence of obstacles in the environment. We classify the moving obstacles in two categories: their motion parameters are known or unknown. One principle for a safe motion of the robot in the dynamic environments is that a robot should always keep a certain distance away from any obstacle whose motion parameters are unknown.

We define a safety range $s_a$ within which the robot has sufficient time to go through a number of sampling and control cycles. By these cycles, the motion states of the obstacles can be detected and proper actions can be taken by the robot to avoid any possible collision. Note that, for a static obstacle, this $s_a$ could be a very small number. For moving obstacles, it must be carefully computed. Let $v_\text{rel}(t)$ denote the velocity of obstacle $O_r$, and $s_r(\tau)$ the safety range of the robot with respect to obstacle $O_r$. $s_r(\tau)$ is a function of $v_\text{rel}(t)$ and $v_\text{s}(t)$, such that

$$s_r(\tau) = \alpha \int_t^{t + \Delta t} v_\text{rel}(\tau) \, d\tau - \beta \int_t^{t + \Delta t} v_\text{s}(\tau) \, d\tau + \gamma_0$$  

where $\alpha$ and $\beta$ are two coefficients; $\alpha, \beta > 0$. The $\gamma_0$ is a constant in which the kinematics of the mobile robot is embedded. Simply assuming $v_\text{rel}(t)$ and $v_\text{s}(t)$ to be constants during the time interval $\Delta t(\tau)$, the above equation becomes

$$s_r(\tau) = \alpha s_\text{rel}(\tau) \Delta t(\tau) - \beta s_\text{s}(\tau) \Delta t(\tau) + \gamma_0$$  

(7b)

$s_a(t)$ can be computed as the maximum of $s_r(\tau)$'s:

$$s_a(t) = \max\{s_r(\tau)\}.$$  

A trajectory-planning range $s_{\text{t}}(t)$ denotes a space within which local planning is made. From the safety motion principle, we see that

$$s_{\text{t}}(t) \geq s_r(t) + s_a(t).$$  

We call (9) the motion-planning constraint equation. Fig. 2 illustrates the relations of $s_t$, $s_r$, and $s_a$ in vectorial and regional diagrams.

The field of view $s_\text{f}(t)$ is the scope observed by the vision system. It represents how far the robot vision system looks ahead and how widely it looks around. We should have

$$s_\text{f}(t) > s_{\text{t}}(t).$$  

The inequality means that obstacles must be observed before reaching the range of local motion planning. Let's use $s_{\text{f}}(t)$ to denote the observing range in which the motion states of obstacles are de-
We call (11) the visual guidance constraint equation. Fig. 3 illustrates the relations of $s_1$, $s_r$, and $s_o$ in vectorial and regional diagrams.

Combining (9) and (11), the vision and control requirement of a local path planning can be specified as

$$s_r(t) = s_1(t) - s_o(t).$$

We also have

$$s_r(t) = s_1(t) - s_o(t).$$

Inserting (12) into (6), we get

$$\int_{t}^{t+T_c(t)+T_d(t)} u_r(\tau) d\tau \leq s_r(t) - s_o(t).$$

where $p_{c}$ stands for the processing capability of the vision and control system. An optimal control state of robot motion can be derived from the above discussion, which is

$$\int_{t}^{t+T_c(t)+T_d(t)} u_r(\tau) d\tau = s_r(t).$$

where $a(t) = 0$.

Obstacle avoidance is substantially more difficult than the static obstacle avoidance. Motion states of obstacles can generally be represented by $(p_o(t), v_o(t), a_o(t))$. For simplicity, the subscript $o_i$ is omitted in the following descriptions.

A. Constant Velocity Model

This is the simplest case of obstacle motion. We have in this model

$$a(t) = 0$$

Obstacle avoidance is governed by certain probability distribution functions. For example:

$$v(t) = e$$

where $w(t)$ is a random vector. In most cases, the probability distribution functions are unknown. Sometimes they can be assumed as a Gaussian or a uniform distribution. The function can be represented as

$$\text{Prob}(w(t)) = f_p(\mu(w(t)), \sigma^2(w(t)))$$

where $\mu(w(t))$ and $\sigma^2(w(t))$ are the mean and variance vectors, respectively, that regulate the distribution of $w(t)$. The $f_p$ is a probability distribution function. It follow that the obstacle velocity can be calculated by

$$v(t) = v(t - \Delta t) + \int_{t-\Delta t}^{t} w(\tau) d\tau.$$ 

where $\Delta t$ is the time interval and $\mu$ and $\sigma^2$ are the mean and variance vectors, respectively, that regulate the distribution of $w(t)$. The $f_p$ is a probability distribution function. It follow that the obstacle velocity can be calculated by

$$v(t) = v(t - \Delta t) + \int_{t-\Delta t}^{t} w(\tau) d\tau.$$ 

where $\Delta t$ is the time interval and $\mu$ and $\sigma^2$ are the mean and variance vectors, respectively, that regulate the distribution of $w(t)$. The $f_p$ is a probability distribution function. It follow that the obstacle velocity can be calculated by

$$v(t) = v(t - \Delta t) + \int_{t-\Delta t}^{t} w(\tau) d\tau.$$ 

B. Random Motion Model

This model considers that the changes of the motion states of an obstacle are governed by certain probability distribution functions. For example:

$$v(t) = e$$

where $e(t)$ is a constant vector. A random-motion model can be described alternatively as

$$\text{Prob}(w(t)) = f_p(\mu(w(t)), \sigma^2(w(t)))$$

where $\mu(w(t))$ and $\sigma^2(w(t))$ are the mean and variance vectors, respectively, that regulate the distribution of $w(t)$. The $f_p$ is a probability distribution function. It follow that the obstacle velocity can be calculated by

$$v(t) = v(t - \Delta t) + \int_{t-\Delta t}^{t} w(\tau) d\tau.$$ 

where $\Delta t$ is the time interval and $\mu$ and $\sigma^2$ are the mean and variance vectors, respectively, that regulate the distribution of $w(t)$. The $f_p$ is a probability distribution function. It follow that the obstacle velocity can be calculated by

$$v(t) = v(t - \Delta t) + \int_{t-\Delta t}^{t} w(\tau) d\tau.$$ 

C. Intentional Motion Model

In this model an obstacle moves in a scheduled route, such as a predetermined destination, or a programmed route. The obstacle may also try to avoid collision with others. In this case, we have

$$a(t) = \epsilon(t)$$

or

$$a(t) = \alpha_1 a(t - \Delta t) + \beta_1 e(t)$$

where $\epsilon(t)$ represents the variations of accelerations resulting from any interval or external forces of the obstacle. $\alpha_1$ and $\beta_1$ are two constants that specify the tendency of acceleration change. The function $\epsilon(t)$ depends very much on the particular environmental conditions.

III. Obstacle Motion Models

Many researchers have indicated that the problem of dynamic obstacle avoidance is substantially more difficult than the static problem [8], [10]. Unlike static obstacle avoidance, the process in dynamic environments must consider any possible moves of the obstacles in terms of a time axis. In this work, we simply restrict the motion of robots in a two-dimensional space $[X, Y]$. The dynamic obstacle avoidance is then carried out in a three-dimensional space $[X, Y, t]$. The difficulty of dynamic obstacle avoidance lies also on the uncertainty of the obstacle motions. To properly predict the obstacle motions, analysis of their motion models should be made. The following three models are studied in our work: 1) constant velocity model, 2) random motion model, and 3) intentional motion model.
The acquisition of $e(t)$ relies very much on the background knowledge of the obstacles and a thorough observation of the motion histories of the obstacles.

It should be pointed out that the actual motions of obstacles in the environment are often a combination of the above motion models. It is obvious that the motions of obstacles in most cases are best described by a stochastic process.

IV. HIDDEN MARKOV MODEL FOR OBSTACLE-MOTION PREDICTION

Prediction of obstacle motion is a fundamental step toward the identification of a collision-free trajectory of the robot motion. In this section we present a computational scheme for the prediction of obstacle motion states in time interval $[t, t + k]$, $k \geq 1$. The states of obstacle motion are denoted as $(p_o(t), v_o(t), a_o(t + k))$.

A. Hidden Markov Model

The motion states of obstacles in dynamic environments can be presented in a formal way through Markov models. The simplest form of a Markov model is a Markov chain [9]. An interesting extension of the Markov chain appears in what has become known as the hidden Markov model (HMM) [9]. The name comes from the consideration that the Markov states cannot be observed directly. They can only be inferred from observation of the time series of other relevant events. Thus, the underlying Markov model is "hidden." In HMM, a number of different stochastic processes are combined to produce observations. HMM’s are useful for a variety of problems where the time series of observations may go through distinct characteristic changes. Obstacle-motion prediction is one such problem.

In applying the HMM for obstacle-motion prediction, a sequence of probability functions and their parameters are defined. They are:

a) an initial motion state description of an obstacle $o_i$: $(p_o(t), v_o(t), a_o(t))$, which comes from the vision processing procedures;

b) an initial probability description of the motion dynamics of an obstacle $o_i$: $\text{Prob}(a_o(t))$;

c) a set of Markov transition parameters $\{\mu(a_o(t + j)), \sigma^2(a_o(t + j))\}$, which represent the mean and variance vectors of a probability transition function;

d) a presumed form of Markov transition function, denoted as $P(a_o(t + j))$;

e) a computation scheme for deriving $\text{Prob}(a_o(t + j))$ from applying the Markov transition functions; and

f) a computation scheme for deriving $\text{Prob}(v_o(t + j))$ and $\text{Prob}(p_o(t + j))$.

Starting with the mean and variance vectors

$$ [\mu(a_o(t + j)), \sigma^2(a_o(t + j))] \quad j = 1, 2, \ldots, k \quad (28) $$

we have the Markov transition function

$$ P(a_o(t + j)) = p(\mu(a_o(t + j)), \sigma^2(a_o(t + j))) \quad (29) $$

where $p$ is a probability density function. The initial mean and variance vector of the transition function are calculated by

$$ \mu(a_o(t)) = f_o(a_o(t - 1), a_o(t - 2), \ldots, a_o(t - N)) \quad (30) $$

and

$$ \sigma^2(a_o(t)) = f_o \left( (a_o(t - 1) - a_o(t - 2))^2, (a_o(t - 2) - a_o(t - 3))^2, \ldots, (a_o(t - N + 1) - a_o(t - N))^2 \right) \quad (31) $$

A diagram of such a probability assumption is shown in Fig. 4.

The Markov transition function $P(a_o(t))$ is extended to the entire time interval $[t, t + k]$, That is

$$ P(a_o(t + j)) = P(a_o(t + j - 1)) = \cdots = P(a_o(t)) \quad (33) $$

We then have the predicted probability of $a_o(t + k)$ at the time $t + k$

$$ \text{Prob}(a_o(t + k)) = \text{Prob}[a_o(t), a_o(t + 1), \ldots, a_o(t + k - 1)] $$

$$ = \left[ \prod_{j=0}^{k-1} P(a_o(t+j)) \right] \quad (34) $$

We also have the mean and variance vectors for the hidden Markov transition function $P(v_o(t))$ and $P(p_o(t))$

$$ [\mu(p_o(t)), \sigma^2(p_o(t))] \quad j = 0, 1, \ldots, k \quad (35) $$

$$ [\mu(p_o(t)), \sigma^2(p_o(t))] \quad j = 0, 1, \ldots, k \quad (36) $$

The initial states of above parameters can be computed by

$$ \mu(v_o(t)) = \mu(v_o(t - 1) + (a_o(t - 1)) $$

$$ = \mu(v_o(t - 1) + (a_o(t - 1)) \quad (37) $$

$$ \sigma^2(v_o(t)) = \sigma^2(v_o(t - 1) + (a_o(t - 1)) = \sigma^2(a_o(t - 1)) \quad (38) $$

and

$$ \mu(p_o(t)) = \mu(p_o(t - 1) + v_o(t - 1) + \frac{1}{2} a_o(t - 1)) $$

$$ = p_o(t - 1) + \mu(v_o(t - 1)) + \frac{1}{2} \mu(a_o(t - 1)) \quad (39) $$
\[
\sigma^2(p_n(t)) = \sigma^2(p_n(t-1) + v_n(t-1) + \frac{1}{2}a_n(t-1))
\]
\[
= \sigma^2(v_n(t-1)) + \frac{1}{2}\sigma^2(a_n(t-1)).
\] (40)

Notice that the time interval $\Delta t = 1$ in the above equations. We then have the hidden Markov transition function

\[
P(v_n(t)) = \frac{1}{\sqrt{2\pi}\sigma(v_n(t))} e^{-\frac{1}{2}\frac{(v_n(t)-\mu(v_n(t)))^2}{\sigma^2(v_n(t))}}
\] (41)

and

\[
P(p_n(t)) = \frac{1}{\sqrt{2\pi}\sigma(p_n(t))} e^{-\frac{1}{2}\frac{(p_n(t)-\mu(p_n(t)))^2}{\sigma^2(p_n(t))}}.\] (42)

The description of the velocity and the position of the obstacle will be expressed in probability density functions

\[
\text{Prob}(v_n(t + k)) = \text{Prob} \left[ v_n(t), v_n(t + 1), \ldots, v_n(t + k - 1) \right]
\]
(43)

and

\[
\text{Prob}(p_n(t + k)) = \text{Prob} \left[ p_n(t), p_n(t + 1), \ldots, p_n(t + k - 1) \right]
\] (44)

where the $P(a_n(t + j)), P(v_n(t + j))$, and $P(p_n(t + j))$ are the Markov transition functions. Equation (44) represents the probability value of position $p(x, y)$ being occupied by obstacle $O_c$ at time $t + k$. In practice, we need to find the probability that position $p(x, y)$ is occupied by any obstacle in the scene. This is calculated by

\[
\text{Prob}(p(t + k)) = \text{Max} \left[ \text{Prob}(p_n(t + k) | w_0) \right].
\] (45)

V. TRAJECTORY-GUIDED MOTION PLANNING

An image-space-based search approach is commonly used for robot motion planning in a static environment [3]-[5]. The approach can be extended to dynamic environments. To apply the approach, a collision-free trajectory is determined in a local map constructed from the scene images. The map depicts the positions and expected position changes of obstacles. These positions form "forbidden regions." A graph search is conducted to identify a collision-free path in a collision-free space of the map. The processing involves the search of a massive number of map elements, and therefore is computationally intensive. Moreover, the results need to be further converted from the geometric representation of the trajectory to the robot motion-control parameters, which are usually the driving forces corresponding to motion acceleration values. The map-based search approach therefore is less attractive than a direct control approach.

We have developed a trajectory-guided strategy [6], [12] for robot motion planning. The method inspects a finite set of candidate trajectories based on the probabilistic evaluations of the candidates using the HMM. The planning process attempts to proceed in several directions simultaneously, i.e., perform a parallel search.

\[\theta(t) = p_g(x, y) - p_i(t)\] (51)

gives a quantitative expression of the relative position of the robot and the goal. Dividing the tessellated motion space into eight sections, as denoted by the letters $A'$ to $H'$ in Fig. 5, $\theta(t)$ is always allocated in one of these sections. A priority list of $\theta_g(t)$ can then be formed according to its relative position with $\theta(t)$. This priority list is shown in Table I. Numbers on the first row denote the priority order. The smaller number represents a higher priority. A simplification of the above case can be made by transforming the point $p_g(x, y)$ to one fixed direction, say, north. A vertical line passing $p_g(x, y)$ will then allocate $\theta(t)$ to only three possible sectors:
depicted in Fig. 6. The number on the directional line indicates priority.

“B,” “C,” and “D.” Priorities of $\theta_n(t)$ in these cases are depicted in Fig. 6. The number on the directional line indicates priority.

C. Evaluation of Candidate Trajectories

The candidate trajectory is selected according to a probabilistic evaluation of collisions. Using $a_i(t) = \{\theta_n(t), A_n(t)\}$ to denote the $i$th trajectory in the candidate set, the corresponding velocities and positions of the robot in $\{t, t + k\}$ can be obtained by

$$v_r(t + k | a_i(t)) = v_r(t) + \int_{t}^{t+k} a_r(\tau) \, d\tau \quad (52)$$

and

$$p_r(t + k | a_i(t)) = p_r(t) + \int_{t}^{t+k} v_r(\tau | a_i(t)) \, d\tau \quad (53)$$

where $a_r(t) = [\theta_n(t), A_n(t)]$.

The probability that a position $p_r(t + j | a_i(t))$, $j = 1, 2, \ldots, k$, is occupied by an obstacle, $\text{Prob}(p_r(t + j | a_i(t)))$, is computed by

$$\text{Prob}(p_r(t + j | a_i(t))) = \max(\text{Prob}(p_r(t + j | \theta_n(t))) \, | v_r(t) \rangle) \quad (55)$$

A threshold value $P_r$ is used to control the termination of an evaluation process. When $\text{Prob}(p_r(t + j | a_i(t)) > P_r$, the evaluation of $a_i(t)$ is abandoned.

An overall evaluation of a candidate trajectory consists of two parts: 1) a deterministic value and 2) a probabilistic value. It is expressed as

$$\text{Val}(a_i(t + j)) = D(p_r(t + j | a_i(t))) \times (1 - \text{Prob}(p_r(t + j | a_i(t)))). \quad (56)$$

Here, $D(p_r(t + j | a_i(t))$ is the Euclidean distance or

$$D(p_r(t + j | a_i(t))) = \sqrt{(p_r(t + j + 1 | a_i(t)))^2 - p_r(t + j | a_i(t))^2} \quad (57)$$

A step-by-step description of the procedure is listed below.

D. Trajectory-Guided Path Planning (An HMM Process)

1) Form a candidate trajectory set, $C_i(t) = \{a_i(t), i = 1, 2, \ldots, m\}$, $a_i(t) = \{\theta_n(t), A_n(t)\}$

2) Order $\{a_i(t), i = 1, 2, \ldots, m\}$ according to the relative position of $\theta_n(t)$ and $\theta_n(t)$.

3) Evaluate $\{a_i(t), i = 1, 2, \ldots, m\}$ simultaneously while processors are available.

FOR each candidate trajectory $a_i(t)$,

3.1) Initialize an evaluation value $\text{Val}(a_i(t)) = 0$.

3.2) FOR each anticipated motion step $j$, $j = 1, 2, \ldots, k$, of $a_i(t)$, where

$$\int_{0}^{\tau_j} \left[ v_r(t) + \int_{0}^{\tau_j} (a_r(\tau)) \, d\tau \right] \, dt \leq S_j(t) \quad (58)$$

3.2.1) Calculate

$$\text{Prob}(p_r(t + j | a_i(t)) = \max(\text{Prob}(p_r(t + j | \theta_n(t))) \, | v_r(t) \rangle) \quad (55)$$

3.2.2) IF $\text{Prob}(p_r(t + j | a_i(t)) > P_r$, terminate the evaluation of $a_i(t)$.

Otherwise calculate

$$\text{Val}(a_i(t)) = \text{Val}(a_i(t)) + \left[ D(p_r(t + j | a_i(t))) \times (1 - \text{Prob}(p_r(t + j | a_i(t))))) \right] \quad (59)$$

3.3) When $j = k$, issue motion command in $a_i(t)$, process terminates.

4) If no motion command has been issued, select an $a_i(t)$ that has the largest $\text{Val}(a_i(t))$ and issue the motion command.

VI. COMPUTER SIMULATION

The computer simulation starts with a binary image where the number of obstacles and the motion models can be set a priori. Initial positions and the motion parameters of obstacles are generated randomly. The system is simulated with 1) a different number of obstacles, $d_0(t)$, 2) a different combination of obstacle motion models, and 3) different values of the field of view $s$. The global goal is specified by a center line of the motion space. Performances of the algorithms are evaluated according to the following criteria: 1) the collision-warning rate, 2) the average vision and control processing time $T_c$, and 3) the deviation of the local path from the global route. A collision-warning signal is generated when the distance between the robot and an obstacle is less than $s(t)$. The summation of warning signals is divided by the total number of motion steps to get the collision-warning rate. The average deviation indicates how well the robot has kept the designated global route when making maneuvers to avoid obstacles.

The performance of the HMM for obstacle-motion prediction was compared with a deterministic prediction approach (DPA) [6]. The results of the simulation expose many important properties. Fig. 7 shows a sequence of graphical display of the simulation, where obstacles are presented in different shapes and patterns. Simulation results are illustrated as follows. For the convenience of comparison, all measurements are scaled to a range of 1 to 10. Fig. 8(a) shows the collision-warning rate with respect to the changes of $s(t)$ for the HMM and the DPA. Fig. 8(b) shows the collision warning rate with respect to $d_0(t)$ of the two approaches. Fig. 9(a) shows the average $T_c$ with respect to the changes of $s(t)$ for the HMM and the DPA. Fig. 9(b) shows the average $T_c$ with respect to the obstacle density $d_0(t)$ of the two approaches. Fig. 10(a) shows the average deviation of the motion trajectory with respect to the changes of $s(t)$ for the HMM and the DPA. Fig. 10(b) shows the average deviation with respect to $d_0(t)$ of the two approaches.

It can be easily seen from Fig. 8 that the HMM approach has a
lower collision-warning rate than the DPA. The slope of the curve is also slower in the HMM approach than the DPA when the $d_{dop}(t)$ increases. On the other hand, as a tradeoff for safety, the results also show that the $T_c(t)$ and deviation increases more in HMM than the DPA. This is worthwhile as safety is the main consideration in most robot motion environments. From the simulation results, we also observe that, when the $s_{c}(t)$ is beyond a certain range, its increase has only a very limited effect on the improvement of the motion parameters. This is because collision avoidance in dynamic environments is basically a local operation. When the $s_{c}(t)$ is sufficiently large, compared to the $s_{c}(t)$, the further increase of $s_{c}(t)$ does not proportionally improve the motion performance. It is also worth indicating that the ratio of $s_{c}$ versus $s_{c}(t)$, expressed as
The problem of visual guidance and dynamic obstacle avoidance of a mobile robot or autonomous vehicle navigating in an unknown environment has been investigated. For collision avoidance in dynamic environments, where both the robot and the obstacles in the scene are moving, one critical problem is to accurately predict the motions of the obstacles. The prediction has to be made using only the limited amount of information from the observations. The uncertain nature of obstacle motion makes such a prediction difficult. Basic motion models of obstacles have been studied in this paper. The HMM for obstacle-motion prediction is described and the limited amount of information from the observations. The uncertain nature of obstacle motion makes such a prediction difficult. Basic motion models of obstacles have been studied in this paper. The HMM for obstacle-motion prediction is described and simulated. The visual guidance and motion-control algorithms discussed in this paper have the following properties: 1) the stochastic model provides an appropriate description of obstacle motions in dynamic environments; 2) the HMM for obstacle-motion prediction reflects the stochastic nature of obstacle motions and is consistent with the motion characteristics of the obstacles; and 3) the trajectory-guided local-path planning algorithm unifies the visual processing and motion control processes in a systematic representation of candidate trajectories. The algorithm is especially beneficial for being able to pursue several potential paths in parallel.

VII. CONCLUSIONS

The problem of visual guidance and dynamic obstacle avoidance of a mobile robot or autonomous vehicle navigating in an unknown environment has been investigated. For collision avoidance in dynamic environments, where both the robot and the obstacles in the scene are moving, one critical problem is to accurately predict the motions of the obstacles. The prediction has to be made using only the limited amount of information from the observations. The uncertain nature of the obstacle motion makes such a prediction difficult. Basic motion models of obstacles have been studied in this paper. The HMM for obstacle-motion prediction is described and simulated. The visual guidance and motion-control algorithms discussed in this paper have the following properties: 1) the stochastic model provides an appropriate description of obstacle motions in dynamic environments; 2) the HMM for obstacle-motion prediction reflects the stochastic nature of obstacle motions and is consistent with the motion characteristics of the obstacles; and 3) the trajectory-guided local-path planning algorithm unifies the visual processing and motion control processes in a systematic representation of candidate trajectories. The algorithm is especially beneficial for being able to pursue several potential paths in parallel.


Weld Pool Edge Detection for Automated Control of Welding

D. Brzakovic and D. T. Khani

Abstract—This work describes a vision system that determines the edges of the weld pool in sequences of gas-tungsten-arc welding images acquired by a coaxial viewing system. The vision system employs a transformation that maps the edge of a weld pool into a vertical line. The weld pool edge is detected in the transform domain by employing a directional filter, which retains only intensity changes of interest, and a one-dimensional edge detector. The edge of the weld pool, in the physical domain, is determined using the inverse transformation. The transformation employs parameters that are updated when processing a sequence of images and are initially determined by analyzing the first image frame in the physical domain.

Key Words—Directional filtering, edge detection, inverse transformation, temporal changes, transformation.

I. INTRODUCTION

For the past two decades there has been a great deal of interest in automating various welding processes. The objective of the automation is to ease the job of the machine operator and in some cases...