

# Discrete Event Control for Mobile Robots

**David J. Austin**

david@faceng.anu.edu.au  
Department of Engineering  
The Australian National University  
Canberra, Australia  
Fax: int + 61 2 6249 0506

**Brenan J. McCarragher**

brenan@faceng.anu.edu.au  
Department of Engineering  
The Australian National University  
Canberra, Australia  
Fax: int + 61 2 6249 0506

## Abstract

*A new technique for the control of mobile robots using a discrete event model is presented. Physical obstacles are represented by constraints on the motion of the robot. In the discrete event model, the discrete state of the system is defined by the currently active constraint. This paper uses dynamic constraints which depend upon the velocity as well as the position of the robot. Dynamic constraints are more accurate than simple, position-based models. The discrete event model is used for the control of a nonholonomic mobile robot in indoor navigation tasks. Experimental results demonstrating the effectiveness of the method for navigation tasks are presented.*

## 1 Introduction

Mobile robotics has long been an area of interest with the goal of expanding the robotics industry from the limited manipulators that are presently used in factories. Unfortunately, the real world presents many new challenges due to the dynamic and uncontrolled environment. Two central problems are of interest in mobile robotics: the development of a model of the environment and the navigation between goals using the environmental model. Both of these problems can be effectively addressed using discrete event methods. This paper uses a discrete event model for the mobile robot system and the surrounding environment. This discrete event model is then used for navigation between goals.

A great deal of research has been undertaken to study methods for map making and navigation. For example, [Kunz *et al.*, 1997] presents a method for the automatic generation of graph-based maps of an office environment and [Chong and Kleeman, 1997] uses sonar to construct maps. Also, [Thrun and Bucken, 1996] considered integration of grid-based and topological maps while [Janet *et al.*, 1997] uses geometric, rather than grid-based, certainty maps. Finally, [Sgouros *et al.*, 1996] considered

the indoor navigation problem. However, the dynamics of the mobile robot change in a discrete fashion as the robot is constrained by obstacles. Traditional methods for mobile robot control neglect the discrete aspect of the system.

Discrete event methods have been considered for mobile robots previously, but only by a few researchers. In particular, Kosecka [Kosecka and Bajcsy, 1994; Kosecka *et al.*, 1995; Kosecka, 1996] considered discrete event modelling of navigation of mobile robots with various tasks defined as discrete states (e.g. `moving`, `steer_away`, `path_following`). However, these models are rather arbitrary without a mathematical basis for the state definitions. Instead, we propose a constraint-based method for defining the discrete states. As the mobile robot moves, it becomes constrained by the obstacles within the environment. We will develop constraints that depend upon the dynamics of the robot. For example, the constraint due to a wall depends upon the approach velocity as well as the distance to the wall. This paper presents a rigorous method for the discrete event modelling of a mobile robot in an office environment.

For this paper, we will consider an indoor environment because accurate mapping is possible. We will also impose the restriction that there are no moving obstacles present in the environment. Basically, we will develop a map of the environment and rely on this map for navigation. In future research, we will consider the use of sensors to detect and avoid unexpected or moving obstacles.

## 2 Discrete Event Modelling

Discrete event modelling is an ideal tool for mobile robotics as the gain and loss of constraints due to obstacles are fundamentally discrete events. Furthermore, discrete event models can be readily extended to deal with dynamic obstacles, such as people or other robots. For this paper, we only consider the static obstacles within the office. More complex obstacles and moving obstruc-

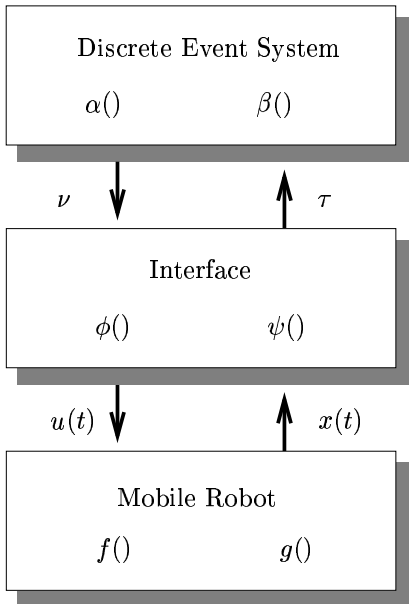


Figure 1: *Block Diagram Representation of System*

tions can also be modelled and will be considered in future works.

The structure of the adopted discrete event model is as shown in Figure 1. The system consists of three parts, which are the mobile robot, the discrete event controller, and the interface. Each component is detailed below, with the description tailored to the indoor navigation systems that we consider. Mathematically, the mobile robot will be modelled by a set of differential equations describing the motion of the robot. The discrete event controller is modelled as an automaton describing task-level decision making, whereas the interface serves as the communication between the mobile robot and the decision maker.

## 2.1 The Mobile Robot System

Consider the general motion control of a mobile robot. The equation of motion of the robot in free space is given generally as:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \quad (1)$$

where  $\mathbf{x}(t)$  is the continuous time state vector, and  $\mathbf{u}(t)$  is the input vector. For the mobile robot, the state vector is defined as:

$$\mathbf{x}(t) = [x \ y \ \theta \ \dot{x} \ \dot{y} \ \dot{\theta}]^T \quad (2)$$

where  $x$ ,  $y$  and  $\theta$  are the position and orientation of the robot in some inertial reference frame.

The mobile robot that we will consider is nonholonomic and so we cannot control all of the state variables independently. The control of the robot allows the specification of a tangential velocity  $v_t$  and a turning speed

or angular velocity  $\omega$ . The ideal equations of motion for the robot are:

$$\dot{x} = v_t \cos \theta \quad (3)$$

$$\dot{y} = v_t \sin \theta \quad (4)$$

$$\dot{\theta} = \omega \quad (5)$$

Of course, the real world is not ideal so that many factors perturb the motion. Examples of disturbances that are difficult to detect are uneven terrain and wheel slip.

As the robot interacts with the environment, the free-space dynamics become constrained by the environment. For each obstacle in the environment, the constraint may be written as:

$$g_i(\mathbf{x}(t)) = 0 \quad (6)$$

where  $g_i$  is the constraint equation for the  $i^{\text{th}}$  obstacle. When the robot is constrained by multiple obstacles, the corresponding constraint equations are active. Note that the constraint depends upon the complete continuous state vector of the robot, including the velocity components. This allows the representation of all physical constraints upon the robot.

For example, a wall constraint involves the velocity and the maximum deceleration of the robot as well as the position of the robot relative to the wall. This constraint may be written as:

$$v = \sqrt{2a_{max}\Delta x} \quad (7)$$

where  $v$  is the velocity of the robot towards the wall,  $a_{max}$  is the maximum deceleration of the robot and  $\Delta x$  is the distance between the robot and the wall. Figure 2 shows an example of a wall constraint, in which the constraint function is plotted. For velocities below the curve, the motion of the robot is not restricted by the wall whilst, for points above the curve, the robot will collide with the wall.

We will consider the constraints imposed by the office walls and also the supporting pillars within the office. In addition, we will consider intersections between walls (or corners in the office) as constraints. Corners can be modelled as a special case of the pillar constraint as corners are effectively a zero-radius pillar. Thus, there is a discrete state for each wall, pillar, and corner.

## 2.2 The Discrete Event System

The mobile robot is controlled by a task-level, discrete event controller modelled as an automaton. The automaton is a quintuple,  $(S, E, C, \alpha, \beta)$ , where  $S$  is the finite set of states,  $E$  is the set of plant events,  $C$  is the set of controller events,  $\alpha : S \times E \rightarrow S$  is the state transition function, and  $\beta : S \rightarrow C$  is the output function. The *discrete state* is a function of the constraint of the robot by obstacles in the environment. Each discrete state in

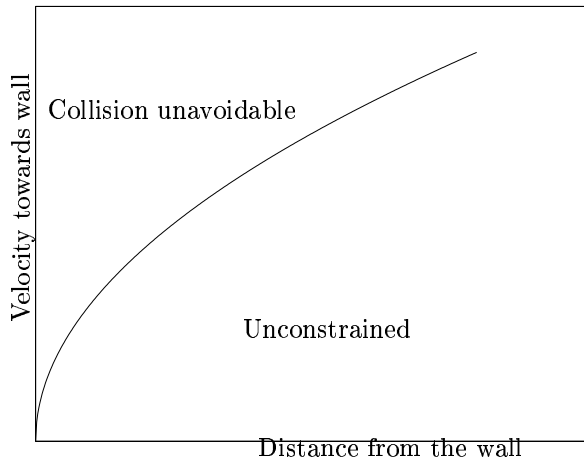


Figure 2: *Constraint as a function of distance and approach velocity*

$S$ , denoted  $\gamma_k$ , represents the constraint of the robot by a different set of obstacles. Each *controller event* in  $C$ , denoted  $v_k$ , is generated by the discrete event controller, whereas each *plant event* in  $E$ , denoted  $\tau_k$ , is generated by the activation of a constraint upon the robot. Plant events are generated when the mobile robot encounters an obstacle.

The dynamics of the discrete event controller are given generally by

$$\gamma_{k+1} = \alpha(\gamma_k, \tau_k) \quad (8)$$

$$v_k = \beta(\gamma_k) \quad (9)$$

where  $\gamma_k \in S$ ,  $\tau_k \in E$  and  $v_k \in C$ . The input and output signals of the discrete event controller are asynchronous sequences of events, rather than continuous time signals. The function  $\alpha$  is functionally dependent on  $f$  and  $g$  as defined by the mobile robot system. Note that the state of the mobile robot system is  $\mathbf{x}$ , whereas  $\gamma$  is the state variable of the discrete event system and is dependent on  $\mathbf{x}$ .

### 2.3 The Interface

The interface is responsible for the communication between the plant and the controller, since these components cannot communicate directly due to the different types of signals being used. The interface consists of two maps  $\phi$  and  $\psi$ .

The first map  $\phi$  converts each controller event into a plant input. The input to the plant consists of a trajectory for the mobile robot to follow.

$$\mathbf{u}(t) = \phi(v_k, \mathbf{x}) \quad t_k \leq t < t_{k+1} \quad (10)$$

Note that the trajectory supplied to the robot is dependent upon the continuous state of the robot as well as the discrete controller event.

The second map  $\psi$  converts the state space of the plant into the set of plant events.

$$\tau_k = \psi(\mathbf{x}(t)) \quad (11)$$

Note that equation (11) does not imply that  $\tau_k$  changes continuously as  $\mathbf{x}(t)$  changes. The state space of the plant is partitioned into contiguous regions. The function  $\psi$  generates a new plant event only when the state first enters one of these regions. The map  $\psi$  is called a *process monitor*.

## 3 Trajectory Generation

There are two types of trajectory used by the robot to navigate to the goal. The first is a discrete state trajectory and the second type is the continuous trajectories used within each discrete state. We will now discuss the generation of both the discrete state and continuous trajectories.

The discrete state trajectory establishes the next desired discrete state. This is used by the discrete event controller to determine the controller event according to equation (9). Generation of the discrete state trajectory is fairly simple because there are a limited number of discrete states. In addition, the small number of discrete states permits automatic generation of the discrete state trajectory using a search technique. However, for this paper the discrete state trajectory is developed by the programmer.

In response to each controller event, the interface supplies a continuous trajectory to the robot according to equation (10). When the robot becomes constrained by an obstacle, there are two possible behaviours. The first is to move around the obstacle maintaining the constraint and the second is to back away avoiding the constraint. Maintaining the constraint is most useful as it provides wall-following trajectories. The method used for trajectory generation depends upon the constraint on the robot. There are four different types of constraint situation: unconstrained, wall constraint, pillar constraint and corner constraint. We will now consider trajectory generation for each of these.

### 3.1 Unconstrained Motion

When the robot is in free space, the movement is unconstrained and we are completely free to choose the trajectory. There are many different methods for trajectory generation but, for simplicity we will consider circular or straight-line trajectories. To find the trajectory which results in the desired discrete state, we use a forward simulation technique. Basically, we divide the control space (translational velocity and angular velocity) into discrete elements and simulate the motion of the robot for each element of the control space to see

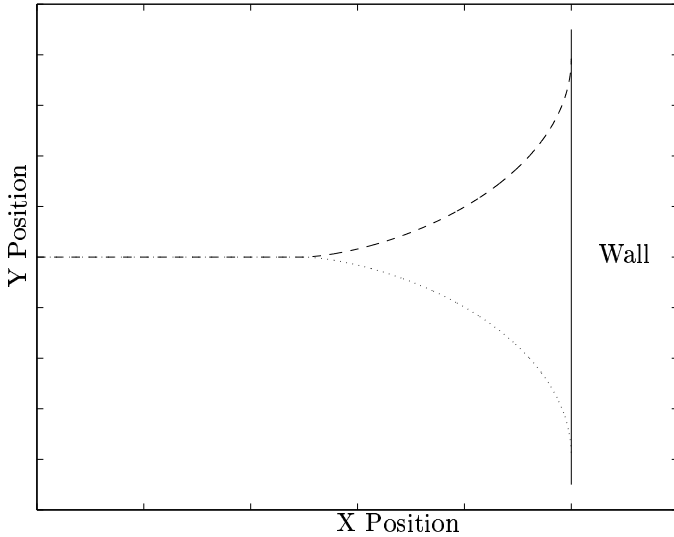


Figure 3: *Paths used to avoid collision with a wall constraint*

which discrete state occurs. The control command resulting in the desired discrete state is then selected. The computation required for this method is not excessive as the calculation is only made once, when we enter an unconstrained discrete state.

### 3.2 Wall constraint

When the robot becomes constrained by a wall, the constraint function  $g_i$  is given by equation (7) and shown in Figure 2. Depending upon the desired discrete state, the robot must follow the wall and maintain the constraint or move away from the wall into free space.

If we wish to maintain the constraint to move towards the desired discrete state then a path of the form shown in Figure 3 must be selected. We choose to maintain the speed of the robot and steer to the left or right to avoid collision. Thus, the desired orientation of the robot is given by:

$$\cos\theta = \pm \sqrt{\frac{2a_{max}\Delta x}{v}} \quad (12)$$

where  $a_{max}$  is the maximum acceleration of the robot,  $\Delta x$  is the distance from the wall, and  $v$  is the velocity of the robot towards the wall. Note that equation (12) has two possible solutions for  $\theta$ , these correspond to steering left or right as shown in Figure 3. We select the steering direction depending upon the desired discrete state.

If we wish to avoid the wall, rather than follow it, we use the simulation technique outlined above to determine a control command.

### 3.3 Pillar constraint

The second type of obstacle within the environment is the supporting pillar. When a pillar is approached we

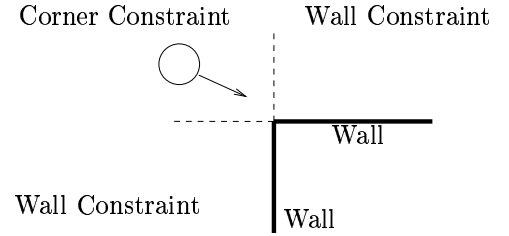


Figure 4: *Corner constraint*

wish to follow a circular trajectory around it. Hence, the angular velocity of the robot is given by:

$$\omega = \frac{v}{r} \quad (13)$$

where  $v$  is the velocity of the robot,  $r$  is the radius of the circle that the robot follows and  $\omega$  is the angular velocity in radians per second. Finally, if we wish to avoid the pillar, rather than move around it, we use the simulation technique to determine a control command.

### 3.4 Corner constraint

The corner constraint applies at the intersection between two wall constraints as shown in Figure 4. The corner constraint represents the restriction of the robot motion caused by the intersection of the walls. As with the pillar constraint, we wish to maintain a constant distance and follow a circular trajectory, using equation (13). Again, if we wish to avoid the corner constraint, rather than move within the constraint, we use the simulation technique outlined above to determine a control command.

## 4 Environment Map

The environmental map  $\psi$  (see equation (11)) is used to determine the discrete state of the robot as it interacts with the environment. Map making is a very complex issue and has been the subject of much research for a considerable period [Chong and Kleeman, 1997; Kunz *et al.*, 1997; Thrun and Bucken, 1996; Janet *et al.*, 1997]. However, the focus of this paper is discrete event control, so we have adopted a rather simplistic mapping strategy.

For this paper, the discrete state  $\gamma$  is determined solely from the continuous state information of the robot according to equation (11). We do not use any of the sensors to detect obstacles. Rather, the sonar sensors are used to correct for drift in the dead-reckoning position obtained from the wheel encoders. This is a simplistic method but is sufficient to demonstrate the discrete event control technique.

## 5 Experiments

The discrete event modelling technique was applied to experimental navigation tasks within an office environ-



Figure 5: *Nomad 200 robot used for experiments.*

ment. The experiments were conducted using the Nomad 200 robot shown in Figure 5. The Nomad 200 robot is subject to a nonholonomic constraint as discussed above. A map of the office environment was supplied to the robot and the position of the robot was obtained from the wheel encoders with sonar measurements used to correct for encoder errors.

Figure 6 shows the path taken through the office to achieve the goal. It can be seen that navigation within the office environment is highly constrained by obstacles and that the constraints change frequently. The discrete event framework provides a complete model for the discrete changes in constraint.

Figure 7 shows the discrete state of the robot as it performed the task, the breaks in the trajectory indicate where the discrete state of the robot changes. The discrete state of the robot changes whenever the robot encounters a new constraint. Figure 7 shows the effect of the dynamic constraints. For example, in the first state transition, from state 0 to state 1 it can be seen that the velocity of the robot towards the wall causes a transition into state 1 earlier than might otherwise be expected. Figure 8 shows the approach velocity of the robot for the transition from state 0 to state 1. Here, the robot

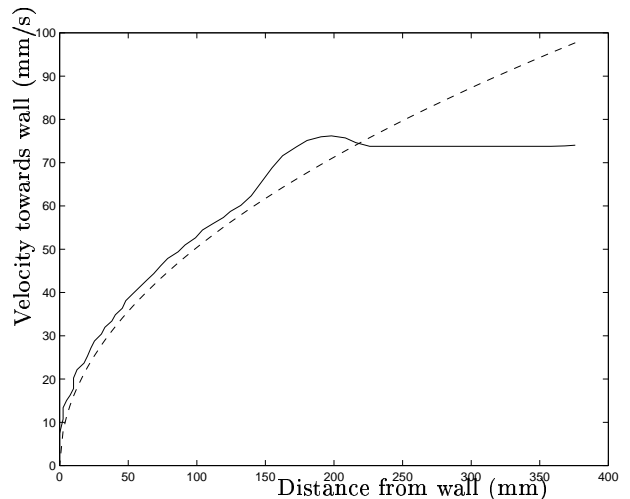


Figure 8: *Comparison of theoretical approach velocity (dashed) and actual approach velocity (solid).*

approaches the wall at constant velocity and recognises the constraint of the wall at a distance of approximately 220 mm. The robot then changes state from the free space state 0 to the constrained state 1 and follows the trajectory given by equation (12). Due to processing delays the robot initially exceeds the constraint as shown in Figure 8.

All of the discrete state transitions in the path are fairly clear with the possible exception of the transition from state 3 to state 0, shown in Figure 7 by the arrow. Here the robot has been instructed to cross the corridor by following the discrete state trajectory  $\{...2, 3, 0, 4...\}$ . To cross the corridor, the robot must move from constrained state 3 to the unconstrained state 0. To move into the unconstrained state the robot does not maintain the constraint in state 3 but finds a control command to reach free space, as was discussed in Section 3.

## 6 Conclusions

A rigorous discrete event modelling technique for non-holonomic mobile robots is proposed. This technique recognises that the dynamics of the mobile robot change in a discrete fashion as the robot becomes constrained by obstacles. Unlike previous discrete event models for mobile robots [Kosecka and Bajcsy, 1994; Kosecka *et al.*, 1995; Kosecka, 1996], the model proposed here is rigorous, using the presence of obstacles to define the discrete state. The discrete state also depends upon the dynamics of the robot so that an accurate model of the obstacle is provided for control purposes. A dynamic model permits greater speeds and so, results in faster completion of the task. Experimental results for an indoor office navigation task were presented, demonstrating the effectiveness of the discrete event model. These

