International Journal of Control
2008, 1–14, iFirst

Autonomous robot navigation using optimal control of probabilistic regular languages

Goutham Mallapragada, Ishanu Chattopadhyay and Asok Ray*

Mechanical Engineering Department, The Pennsylvania State University, PA 16802, USA

(*Received 2 October 2007; final version received 25 January 2008*)

This paper addresses autonomous intelligent navigation of mobile robotic platforms based on the recently reported algorithms of language-measure-theoretic optimal control. Real-time sensor data and model-based information on the robot's motion dynamics are fused to construct a probabilistic finite state automaton model that dynamically computes a time-dependent discrete-event supervisory control policy. The paper also addresses detection and avoidance of livelocks that might occur during execution of the robot navigation algorithm. Performance and robustness of autonomous intelligent navigation under the proposed algorithm have been experimentally validated on Segway RMP robotic platforms in a laboratory environment.

Keywords: robot navigation and obstacle avoidance; discrete event supervisory control; language measure

1. Introduction

Navigation and obstacle avoidance is a widely addressed topic in robotics research. Apparently, a sufficiently general solution of this problem needs to be developed to address the following issues:

- navigation in unknown terrain and noisy environments;
- real-time processing of sensor data and code execution;
- robustness to intermittent failures in sensors and communication links;
- inherent vehicular dynamics of the robotic platforms.

A majority of the reported navigation techniques involve continuous-parameter implementation, where the computational cost becomes a limiting factor. Diverse techniques, such as configuration space approach, Voronoi diagram, retraction method, potential field function, visibility graph, accessibility graph, and tangent graph, have been used with reasonable success (Lozano-Perez 1983; Fujimura 1991; Hwang and Ahua 1992) to address the problem of intelligent navigation. Among these analytical tools, the potential field methods have emerged as a particularly attractive choice due to their elegance and simplicity (Koren and Borenstein 1991). However, they suffer from several drawbacks as delineated below.

1. Likelihood of being trapped in local minima leading to cyclic behaviour.
2. No passage between closely spaced obstacles.
3. Oscillations in the presence of obstacles and/or narrow passages.

In most of the above navigation techniques, either a completely known or partially known map of the environment is assumed. Advances in sensing technology and on-board instrumentation have made it possible to perceive the environment in real time with remarkable accuracy, which has facilitated a purely reactive approach to obstacle avoidance. However, processing large volumes of sensor data in real time raises open issues concerning computational efficiency and robustness, especially under noisy environments and intermittent failures of sensor and communication links. A typical robot needs information from multiple sensors (e.g., sonars, laser, infrared, and tactile) and control inputs from external sources to arrive at optimal commands for the lower level continuous controller. Fusion of such diverse information (i.e., sensor data and external commands) often leads to conflicting output commands from the navigation algorithm resulting in a livelock situation for the robot (Mallapragada, Chattopadhyay and Ray 2006).

Another issue is the robot's motion dynamics that become increasingly important as the mass and working velocities of autonomous platforms are increased. So far very few techniques have been

*Corresponding author. Email: axr2@psu.edu

ISSN 0020–7179 print/ISSN 1366–5820 online
© 2008 Taylor & Francis
DOI: 10.1080/00207170801947096
http://www.informaworld.com

reported in open literature, which takes into account the effects of robot dynamics in obstacle avoidance (Brock and Khatib 1999). For example, Fox, Burgard and Thrun (1997) have applied the Dynamic Window concept to collision avoidance for optimising a continuous function of (v, w) that reflects dynamics, goal heading and the obstacle map. An immediate consequence is high computational cost and one has to resort to simplifying assumptions to find meaningful solutions of (v, w). A discrete-event approach has been proposed in Chattopadhyay, Chakraborty and Ray (2005) for navigation but no dynamics have been included in the model.

This paper addresses autonomous intelligent navigation of mobile robotic platforms based on the concept of measure-theoretic optimal control of probabilistic regular languages (Chattopadhyay and Ray 2007). With the objective of dynamically computing a time-dependent discrete-event supervisory control policy, the proposed algorithm fuses real-time navigation sensor data and model-based information on motion dynamics to construct a probabilistic finite state automaton model for representation of the (regular) language of the robot's dynamical behaviour. This model facilitates both local and global path planning by assigning appropriate characteristic weights to the automaton states. Furthermore, a non-regular observer is constructed to estimate the behaviour of the navigation algorithm to detect livelock situations, if any. Upon detection, a livelock is mitigated by sending external commands to the navigation algorithm.

The paper is organised in six sections. Section 2 briefly reviews the fundamentals of language measure (Ray 2005). Section 3 presents an extension of the language measure (Ray 2005) to time-dependent control specifications. Section 4 introduces the notion of navigation automaton and represents the continuous dynamics of the plant (e.g., mobile robot) as a probabilistic finite state machine. This section also provides a method to detect and avoid livelocks. Section 5 presents simulation and experimental results on robotic platforms. The paper is summarised and concluded in Section 6 along with recommendations for future research.

2. Preliminary concepts

This section briefly reviews the concept of signed real measure of regular languages (Ray 2005; Ray, Phoha and Phoha 2005) followed by a review of the notion of renormalised measure. This section also reviews the optimal supervision problem and its solution.

2.1 *Brief review of language measure*

Let a deterministic finite state automaton (DFSA) be represented as $G_i \triangleq (Q, \Sigma, \delta, q_i, Q_m)$, where Q is the finite set of states with $|Q| = n$, and $q_i \in Q$ is the initial state; Σ is the (finite) alphabet of events with $|\Sigma| = m$; the Kleene closure of Σ is denoted as $\Sigma^\star$, that is, the set of all finite-length strings of events including the empty string ε; the (possibly partial) function $\delta: Q \times \Sigma \to Q$ represents state transitions and $\delta^\star: Q \times \Sigma^\star \to Q$ is an extension of δ; and $Q_m \subseteq Q$ is the set of marked (i.e., accepting) states. The system dynamics are modelled as a probabilistic finite state automaton (PFSA), which is a DFSA augmented by the characteristic function and event generation probabilities.

Definition 1: The characteristic function χ: $Q \to [-1, 1]$ assigns a signed real weight to each state q_i, $i = 1, 2, \ldots, n$ such that relatively more positive weights are assigned to relatively more desirable states.

Definition 2: The event generation probabilities are specified as $\tilde{\pi}$: $\Sigma^\star \times Q \to [0, 1]$ such that $\forall q_j \in Q$, $\forall \sigma_k \in \Sigma$, $\forall s \in \Sigma^\star$,

1. $\tilde{\pi}[\sigma_k, q_j] \triangleq \tilde{\pi}_{jk} \in [0, 1)$; $\sum_k \tilde{\pi}_{jk} = 1$;
2. $\tilde{\pi}[\sigma, q_j] = 0$ if $\delta(q_j, \sigma)$ is undefined; $\tilde{\pi}[\varepsilon, q_j] = 1$;
3. $\tilde{\pi}[\sigma_k s, q_j] = \tilde{\pi}[\sigma_k, q_j]\ \tilde{\pi}[s, \delta(q_j, \sigma_k)]$.

The $\tilde{\mathbf{\Pi}}$-matrix is defined as $\tilde{\mathbf{\Pi}}_{ij} = \tilde{\pi}(q_i, \sigma_j)$, $q_i \in Q$, $\sigma_j \in \Sigma$.

Remark 1: The $\tilde{\mathbf{\Pi}}$-matrix is analogous to the morph matrix of a Markov chain in the sense that an element $\tilde{\pi}_{ij}$ represents the probability of the jth event occurring at the ith state.

Definition 3: The probabilistic state transition map of the probabilistic finite state automaton (PFSA) is defined as a function $\pi: Q \times Q \to [0, 1)$ such that

$$\pi(q_j, q_k) = \begin{cases} 0 & \text{if } \{\sigma \in \Sigma\colon \delta(q_j, \sigma) = q_k\} = \emptyset \\ \displaystyle\sum_{\sigma \in \Sigma: \delta(q_j, \sigma) = q_k} \tilde{\pi}(\sigma, q_j) \triangleq \pi_{jk} & \text{otherwise.} \end{cases} \quad (1)$$

The $\mathbf{\Pi}$-matrix, defined as $\mathbf{\Pi}_{ij} = \pi(q_i, q_j)$, $q_i, q_j \in Q$.

Remark 2: The $\mathbf{\Pi}$-matrix is analogous to the state transition probability matrix of a Markov chain in the sense that an element π_{jk} is analogous to the transition probability from state q_j to state q_k. The language generated by a PFSA is defined to be a probabilistic regular language.

2.2 *Measure of probabilistic regular languages*

The regular language generated by the PFSA under consideration is a sublanguage of the Kleene closure Σ^* of the alphabet Σ.

Definition 4: The formal measure of the generated language of a PFSA with respect to a defined characteristic weight vector (Chattopadhyay and Ray 2006) is defined as

$$\mathbf{v}(\theta) = \theta\, \boldsymbol{\mu}(\theta) = \theta[\mathbf{I} - \boldsymbol{\Pi}(1-\theta)]^{-1}\boldsymbol{\chi} \quad \text{with} \quad \theta \in (0,1). \tag{2}$$

Proposition 1: *The limiting measure vector* $\nu(0) \triangleq \lim_{\theta \to 0^+} \mathbf{v}(\theta)$ *exists and* $\|\mathbf{v}(0)\|_\infty \leqslant 1$.

Proof: Given in Chattopadhyay and Ray (2006). □

Proposition 2: *Let* $\mathbf{P}$ *be the transition matrix of a finite Markov chain (or equivalently a probabilistic regular language). Then, as the parameter* $\theta \to 0^+$, *the limiting measure vector is obtained as:* $\mathbf{v}(0) = \mathscr{P}\boldsymbol{\chi}$ *where the matrix operator*

$$\mathscr{P} \triangleq \lim_{k\to\infty} \frac{1}{k}\sum_{j=0}^{k-1} \mathbf{P}^j.$$

Proof: Given in Chattopadhyay and Ray (2006). □

Corollary 1 (to Proposition 2)**:** *The expression* $\mathscr{P}\mathbf{v}(\theta)$ *is independent of* θ. *Specifically, the following identity holds for all* $\theta \in (0,1)$.

$$\mathscr{P}\mathbf{v}(\theta) = \mathscr{P}\boldsymbol{\chi}. \tag{3}$$

Proof: Given in Chattopadhyay and Ray (2006). □

2.3 *Supervision of probabilistic finite state automaton (PFSA)*

Plant models considered in this paper are deterministic finite state automata with well-defined event occurrence probabilities. In other words, the occurrence of events is probabilistic, but the state at which the plant ends up, given a particular event has occurred, is deterministic. Furthermore, no emphasis is laid on the initial state of the plant and it is assumed that the plant may start from any state. Furthermore, having defined the characteristic state weight vector $\boldsymbol{\chi}$, it is not necessary to specify the set of marked states, because if $\chi_{\mathrm{i}} = 0$, then q_i is not marked and if $\chi_i \neq 0$, then q_i is marked.

Definition 5 (control philosophy)**:** Disabling any transition σ at a given state q results in reconfiguration of the automaton structure as: Set the self-loop $\delta(q,\sigma) = q$ with the occurrence probability of σ from the state q remaining unchanged in the supervised and unsupervised plants.

Remark 3: The control philosophy in Definition 5 is natural in the following sense. If $q_i \underset{\sigma}{\to} q_k$, and the controllable event σ is disabled at state q_i, then the sole effect of the supervisory action is to prevent the plant from making a transition to the state q_k. That is, the plant is forced to stay at the original state q_i and this is represented by the additional self-loop at state q_i instead of the original arc from q_i to q_k.

Definition 6 (controllable transitions)**:** For a given plant, transitions that can be disabled in the sense of Definition 5 are defined to be controllable transitions in the sequel. The set of controllable transitions in a plant is denoted $\mathscr{C}$. Note controllability is state-based.

It follows from Definition 2 and Definition 6 that plant models are completely specified by a sextuple as

$$G = (Q, \Sigma, \delta, \widetilde{\Pi}, \chi, \mathscr{C}). \tag{4}$$

2.4 *The optimal supervision problem: formulation and solution*

A supervisor disables a subset of the set $\mathscr{C}$ of controllable transitions and hence there is a bijection between the set of all possible supervision policies and the power set $2^{\mathscr{C}}$. That is, there exists $2^{|\mathscr{C}|}$ possible supervisors and each supervisor is uniquely identifiable with a subset of $\mathscr{C}$ and the language measure ν allows a quantitative comparison of different supervision policies. The following notations are needed for elementwise comparison of finite-dimensional vectors and matrices for the analysis developed in the sequel.

Notation 1: Let V^a and V^b be r-dimensional real vectors. The following elementwise equality and inequalities imply that

$$\left(V^a =_{\boldsymbol{E}} V^b\right) \Leftrightarrow \left(V_i^a = V_i^b\right) \forall i \in \{1,\ldots,r\} \tag{5a}$$

$$\left(V^a \geqq_{\boldsymbol{E}} V^b\right) \Leftrightarrow \left(V_i^a \geqq V_i^b\right) \forall i \in \{1,\cdots,r\} \tag{5b}$$

$$\left(V^a >_{\boldsymbol{E}} V^b\right) \Leftrightarrow \left(V_i^a > V_i^b\right) \forall i \in \{1,\ldots,r\} \tag{5c}$$

Definition 7: For an unsupervised plant $G = (Q, \Sigma, \delta, \widetilde{\Pi}, \chi, \mathscr{C})$, let $G^\dagger$ and $G^\ddagger$ be the supervised plants with sets of disabled transitions, $\mathscr{D}^\dagger \subseteq \mathscr{C}$ and $\mathscr{D}^\ddagger \subseteq \mathscr{C}$, respectively, whose measures are $\mathbf{v}^\dagger$ and $\mathbf{v}^\ddagger$. Then, the supervisor that disables $\mathscr{D}^\dagger$ is defined to be superior to the supervisor that disables $\mathscr{D}^\ddagger$ if $\mathbf{v}^\dagger \geqq_{\boldsymbol{E}} \mathbf{v}^\ddagger$ and strictly superior if $\mathbf{v}^\dagger >_{\boldsymbol{E}} \mathbf{v}^\ddagger$.

Definition 8 (optimal supervision problem)**:** Given a (non-terminating) plant $G = (Q, \Sigma, \delta, \widetilde{\Pi}, \chi, \mathscr{C})$, the problem is to compute a supervisor that disables a subset $\mathscr{D}^\star \subseteq \mathscr{C}$, such that $\mathbf{v}^\star \geqq_{\boldsymbol{E}} \mathbf{v}^\dagger\ \forall \mathscr{D}^\dagger \subseteq \mathscr{C}$ where $\mathbf{v}^\star$ and

$\mathbf{v}^\dagger$ are the measure vectors of the supervised plants $G^\star$ and $G^\dagger$ under $\mathscr{D}^\star$ and $\mathscr{D}^\dagger$, respectively.

The solution to the supervision problem is reported in Chattopadhyay and Ray (2007) as an optimal control policy for a terminating plant (Garg 1992a,b) with a substochastic transition probability matrix $\widetilde{\Pi}(1-\theta)$ with $\theta \in (0,1)$. To ensure that the computed optimal policy coincides with the one for $\theta = 0$, the suggested algorithm chooses a *small* value for θ in each iteration step of the design algorithm. However, choosing θ to be too small may cause numerical problems in convergence. Two pertinent algorithms, reported in Chattopadhyay and Ray (2007), are presented below.

Algorithm 1 computes how small a θ is actually required, i.e., computes the critical lower bound $\theta_\star$. Thus Algorithm 2, in conjunction with Algorithm 1 solves the optimal supervision problem for a generic PFSA as reported in Chattopadhyay and Ray (2007).

Next we present Propositions 3–6 that are critical for development of the autonomous navigation algorithm in the sequel are presented here without proof for the sake of brevity. The complete proofs are provided in Chattopadhyay and Ray (2007).

Proposition 3: *Let $\mathbf{v}^{[k]}$ be the language measure vector computed in the kth iteration of Algorithm 2. The measure vectors computed by the algorithm form an elementwise non-decreasing sequence, i.e., $\mathbf{v}^{[k+1]} \geqq_E \mathbf{v}^{[k]}\ \forall k$.*

Proposition 4 (Effectiveness)**:** *Algorithm 2 is an effective procedure* (Hopcroft, Motwani and Ullman 2001), *i.e., it is guaranteed to terminate.*

Proposition 5 (Optimality)**:** *The supervision policy computed by Algorithm 2 is optimal in the sense of Definition 8.*

Proposition 6 (Uniqueness)**:** *Given an unsupervised plant G, the optimal supervisor $G^\star$, computed by Algorithm 2, is unique in the sense that it is maximally permissive among all possible supervision policies with optimal performance. That is, if $\mathscr{D}^\star$ and $\mathscr{D}^\dagger$ are the disabled transition sets, and $\mathbf{v}^\star$ and $\mathbf{v}^\dagger$ are the language measure vectors for $G^\star$ and an arbitrarily supervised plant $G^\dagger$, respectively, then $\mathbf{v}^\star =_E \mathbf{v}^\dagger \rightarrow \mathscr{D}^\star \subset \mathscr{D}^\dagger \subseteq \mathscr{C}$.*

3. Supervisory control with time-dependent specifications

This section modifies and extends the supervisor design procedure, presented in Algorithm 2, to time-dependent control specifications. The key generalisations are presented below.

Algorithm 1: Computation of the critical lower bound $\theta_\star$

```
input : $\mathbf{P}$, $\chi$
output: $\theta_\star$
begin
  Set $\theta_\star = 1$;                     /* set to the supremum */
  Set $\theta_{curr} = 0$;                    /* set to the infimum */
  Compute $\mathscr{P}$;      /* Stable Prob. Dist. See [9] */
  Compute $M_0 = [\mathbb{I} - \mathbf{P} + \mathscr{P}]^{-1}$;
  Compute $M_1 = \left[\mathbb{I} - [\mathbb{I} - \mathbf{P} + \mathscr{P}]^{-1}\right]$;
  Compute $M_2 = \inf_{\alpha \neq 0} \left\| [\mathbb{I} - \mathbf{P} + \alpha\mathscr{P}]^{-1} \right\|_\infty$;
  for j = 1 to n do
    for i = 1 to n do
      if $(\mathscr{P}\chi)_i - (\mathscr{P}\chi)_j \neq 0$ then
        $\theta_{curr} = \frac{1}{8M_2}\left|(\mathscr{P}\chi)_i - (\mathscr{P}\chi)_j\right|$
      else
        for r = 0 to n do
          if $(M_0\chi)_i \neq (M_0\chi)_j$ then
            Break;
          else
            if $\left(M_0 M_1^r \chi\right)_i \neq \left(M_0 M_1^r \chi\right)_j$ then
              Break;
            endif
          endif
        endfor
        if r == 0 then
          $\theta_{curr} = \frac{\left|\{(M_0 - \mathscr{P})\chi\}_i - \{(M_0 - \mathscr{P})\chi\}_j\right|}{8M_2}$;
        else
          if r > 0 AND r ⩽ n then
            $\theta_{curr} = \frac{|(M_0 M_1 \chi)_i - (M_0 M_1 \chi)_j|}{2^{r+3} M_2}$;
          else
            $\theta_{curr} = 1$;
          endif
        endif
      endif
    endfor
  endfor
end
```

Definition 9: The parametric characteristic function $\chi(\Lambda(t))$ assigns a signed real weight to each state q_i as a time-dependent Riemann-integrable function of a finite set of parameters $\Lambda(t) = \{\lambda_1(t), \ldots, \lambda_k(t)\} \in \mathbb{R}^k$, where the parameter t denotes the continuous time.

Definition 10: Given a plant $G \equiv (Q, \Sigma, \delta, \widetilde{\Pi}, \chi(\Lambda(t)), \mathscr{C})$, the generalised language measure is defined, for a given initial time t_0, as:

$$\boldsymbol{\eta} = \lim_{t_f \to \infty} \left(\frac{1}{t_f - t_0}\right) \int_{t_0}^{t_f} \mathbf{v}(\chi)\,dt \in \mathbb{R}^n,$$

where n is the number of states.

Algorithm 2: Computation of optimal supervisor

```
input: $\mathbf{P}$, $\chi$, $\mathscr{C}$
output: Optimal set of disabled transitions $\mathscr{D}^\star$
begin
```

```
Set $\mathscr{D}^{[0]} = \emptyset$;                 /* Initial disabling set */
Set $\widetilde{\Pi}^{[0]} = \widetilde{\Pi}$;   /* Initial event prob. matrix */
Set $\theta_\star^{[0]} = 0.99$;              /* set as an upper bound */
Set k = 1;
Set Terminate = false;
While (Terminate == false) do
    Compute $\theta_\star^{[k]}$;                        /* Algorithm */
    Set $\widetilde{\Pi}^{[k]} = \frac{1-\theta_\star^{[k]}}{1-\theta_\star^{[k-1]}}\widetilde{\Pi}^{[k-1]}$;
    Compute $\mathbf{v}^{[k]}$;
    for j = 1 to n do
        for i = 1 to n do
            Disable all controllable transitions $q_i \xrightarrow{\sigma} q_j$
            such that $\mathbf{v}_j^{[k]} < \mathbf{v}_i^{[k]}$;
            Enable all controllable transitions $q_i \xrightarrow{\sigma} q_j$
            such that $\mathbf{v}_j^{[k]} \geqq \mathbf{v}_i^{[k]}$;
        endfor
    endfor
    Collect all disabled transitions in $\mathscr{D}^{[k]}$;
    if $\mathscr{D}^{[k]} == \mathscr{D}^{[k-1]}$ then
        Terminate = true;
    else
        k = k + 1;
    endif
endw
$\mathscr{D}^\star = \mathscr{D}^{[k]}$;                 /* Optimal disabling set */
end
```

Assuming that the state set Q and the event alphabet Σ are invariants, the optimisation procedure can be formalised as follows:

Lemma 1: *Let $G(t) \triangleq (Q, \Sigma, \delta, \widetilde{\Pi}, \chi(\Lambda(t)), \mathscr{C})$ be a plant model and let $G^\star(t)$ be a time-dependent supervisor that disables the transition set $\mathscr{D}^\star(t)$ at the time instant t. Then, $G^\star(t)$ is optimal for every $t \in [0, \infty]$ in the sense of Definition 8 if and only if $G^\star(t)$ is the optimal supervisor at every fixed t, computed by Algorithm 2 for the plant $(Q, \Sigma, \delta, \widetilde{\Pi}, \chi(\Lambda(t)), \mathscr{C})$.*

Proof: The result follows by noting that, for two arbitrary (but integrable) measure vectors $\mathbf{v}^\star(\chi(\Lambda(t))$, $\mathbf{v}(\chi(\Lambda(t)) \in \mathbb{R}^n$, we have

$$\mathbf{v}^\star(\chi(\Lambda(t)) \geqq_{\boldsymbol{E}} \mathbf{v}\big(\chi(\Lambda(t)\big) \ \forall t \in [0, \infty]$$

$$\Longrightarrow \lim_{t_f \to \infty} \frac{1}{(t_f - t_0)} \int_{t_0}^{t_f} \mathbf{v}^\star\big(\chi\big) \mathrm{d}t \geqq_{\boldsymbol{E}} \lim_{t_f \to \infty} \frac{1}{(t_f - t_0)} \int_{t_0}^{t_f} \mathbf{v}\big(\chi\big) \mathrm{d}t.$$

□

If the control specification varies on a slow time scale such that it can be sampled at a preset sampling rate ΔT, Lemma 1 yields the optimal control algorithm shown in Algorithm 3.

Algorithm 3: Optimal control with time-dependent specifications

input : Plant model, parametric time-dependent characteristic function
output: Optimal supervisor with possible time-dependent enablings/ disablings

```
begin
    t = 0;
    while t < $\infty$ do
        Set control specification: $\chi = \chi(\Lambda(t\Delta T))$;
        Compute optimal supervisor for current plant
        $(Q, \Sigma, \delta, \tilde{\Pi}, \chi(\Lambda(t\Delta T)), \mathscr{C})$ by Algorithm 2;
        while $t\Delta T < (t+1)\Delta T$ do
            Enforce computed supervisor;
            Increment t;
        endw
    endw
end
```

The navigation algorithm, developed in the sequel, determines the specification weights from sensory observation of the obstacle distribution in an environment that may change slowly relative to the dynamics of robot motion. In this regard, the algorithm is formulated based on the concept of singular perturbation (Kokotovic, Khalil and O'Reilly 1999) to address the two-time-scale behaviour in the presence of both fast and slow dynamics. The parameters of slow dynamics due to environmental changes are approximated to be constants in the fast time scale of robot motion. On the other hand, the governing equations of fast dynamics are reduced to algebraic expressions by assuming the robot motion to be instantaneous relative to the slow time scale of the environment (i.e., obstacle dynamics). Uniform asymptotic stability and convergence of the two-time-scale composite system is established by an application of Tikhonov's theorem (cited and explained in Kokotovic et al. (1999)) under the following two assumptions:

1. Internal stability of the fast scale dynamics, i.e., the real part of each eigenvalue of the Jacobian in fast-scale dynamics is less than or equal to an *a priori* fixed strictly negative real number.
2. Uniform asymptotic stability of quasistatic equilibria in the fast-scale dynamics under small perturbations in the slow-scale parameters, i.e., all such equilibria must be contained within a domain of attraction that is determined by the pre-specified upper bounds of the exogenous perturbations.

The key requirement here is that the above assumptions must be valid. The first assumption is assured to be valid by the product design of the robot manufacturer (i.e., Segway in the present experiments). The second assumption is satisfied by prohibiting any major perturbations of the environment, such as large movements of the obstacles, during the execution of each iteration of Algorithm 3. In other words, the obstacles are restricted to be either stationary or slowly moving objects. This assertion is not very restrictive because Algorithm 2 is executed significantly fast relative to the data refresh rate.

Operation of the navigation sensors (e.g., laser range finders and sonars) at higher frequencies effectively increases the data refresh rates. Since the obstacle dynamics are sufficiently slow (i.e., in the sense of Tikhonov) compared to the robot motion, the effects of variations in the sampling rate and other issues such as inter-sample transient behaviour and the nature of samplers are insignificant from the perspectives of robot control.

4. Application to intelligent navigation

This section introduces the notion of navigation automaton as a probabilistic finite state machine (PFSA) (see Equation 4). Let us consider a generic plant, equipped with a continuous controller at the lower level of the hierarchy. A typical example of such a plant is a mobile robotic platform (e.g., Segway RMP, or PIONEER DX) running the open-source `PLAYER` server (Gerkey, Vaughan and Howard 2003). It is further assumed that the low-level controller can accept commands of velocity set-points. Such a situation is easily realisable by running the `PLAYER` server (Gerkey et al. 2003) on a Segway RMP and using a client program to write velocity commands to the robot via the `PLAYER` position drivers. In this model, only a finite set of velocity set-points serve as commands from the higher level.

The navigation automaton is constructed with the following components.

1. A finite alphabet $\Sigma = \mathscr{C} \bigcup \Sigma^{\mathcal{D}}$ of events, where $\mathscr{C}$ is the set of controllable commands, and $\Sigma^{\mathcal{D}}$ is the set of uncontrollable dynamic events.
2. A finite set $Q = Q^{\mathcal{D}} \bigcup Q^{\mathscr{I}}$ of states, where $Q^{\mathcal{D}}$ is set of decision states from which a robot is given a command $c_j \in \mathscr{C}$, and $Q^{\mathscr{I}}$ is the set of intermediate states through which the robot traverses while a command from a decision state is in execution. The robot settles down to another decision state via an uncontrollable transition $\sigma_i \in \Sigma^{\mathcal{D}}$ from an intermediate state $q_i^{\mathscr{I}} \in Q^{\mathscr{I}}$.
3. The transition function δ: $Q \times \Sigma \to Q$ is restricted such that only controllable transitions may occur from a decision state $q^{\mathcal{D}} \in Q^{\mathcal{D}}$ and only uncontrollable transitions may occur from an intermediate state $q_i^{\mathscr{I}} \in Q^{\mathscr{I}}$. It follows that the plant must visit an intermediate state in between two decision states and vice versa.
4. The event cost matrix $\tilde{\Pi}$ (see Definition 2).
5. The time-dependent state weight vector $\chi(\Lambda(t))$ (see Definition 9).

The above concepts are formally defined as follows.

Definition 11: The command set $\mathscr{C}$ is the set of (controllable) symbolic commands that a higher level supervisor may issue to the lower level of continuous dynamics. Each controllable symbol (i.e., command) corresponds to a specific velocity set-point that is written to the low-level continuous controller of the robot. The individual commands are denoted by $c_j \in \mathscr{C}$ and the cardinality of $\mathscr{C}$ (i.e., total number of commands) is denoted by $|\mathscr{C}|$.

Definition 12: The set of decision states $Q^{\mathcal{D}}$ is a finite partition of the achievable continuous velocity state space of the plant. For a planar robot, each $q_i^{\mathcal{D}} \in Q^{\mathcal{D}}$ forms an equivalence class of velocity pairs (i.e., linear velocity and turn rate) that the robot can achieve. Let $\Sigma^{\mathcal{D}}$ be another finite set having the same cardinality as $Q^{\mathcal{D}}$ such that $Q^{\mathcal{D}} \cap \Sigma^{\mathcal{D}} = \emptyset$.

Remark 4: The set $Q^{\mathcal{D}}$ can also be seen as a partition of the entire dynamic state space. The partitions are functions of only velocity components and do not depend on the acceleration components. The set $\Sigma^{\mathcal{D}}$ consists of uncontrollable events corresponding to the uncontrollable transitions of the physical system upon execution of a command when the robot is in a state belonging to $Q^{\mathcal{D}}$.

Next, we introduce the concept of navigation automaton.

Definition 13: A navigation automaton is a PFSA $G \triangleq (Q, \Sigma, \delta, \tilde{\Pi}, \chi(\Lambda(t)), \mathscr{C})$ with the following structure:

$$Q = Q^{\mathcal{D}} \bigcup Q^{\mathscr{I}} \tag{6a}$$

$$Q^{\mathscr{I}} = \left(Q^{\mathcal{D}} \times \mathscr{C}\right) \tag{6b}$$

$$\Sigma = \mathscr{C} \bigcup \Sigma^{\mathcal{D}} \tag{6c}$$

$$\delta\colon Q^{\mathcal{D}} \times \mathscr{C} \to Q^{\mathscr{I}} \vee Q^{\mathscr{I}} \times \Sigma^{\mathcal{D}} \to Q^{\mathcal{D}} \tag{6d}$$

$$\forall q_i^{\mathcal{D}} \in Q^{\mathcal{D}}, \forall c_j \in \mathscr{C},\ \tilde{\pi}(q_i^{\mathcal{D}}, c_j) = \frac{1}{|\mathscr{C}|}, \tag{6e}$$

where all events in $\mathscr{C}$ are (controllable) commands and all (dynamic) events in $\Sigma^{\mathcal{D}}$ are uncontrollable.

Remark 5: Equation (6e) specifies that commands are generated at each decision state with approximately uniform probability.

Figure 1 provides an example of a navigation automaton with three commands $\{c_1, c_2, c_3\}$ and two decision states $\{q_1^{\mathcal{D}}, q_2^{\mathcal{D}}\}$. The intermediate states are denoted as q_j^i where the indices i and j denote that

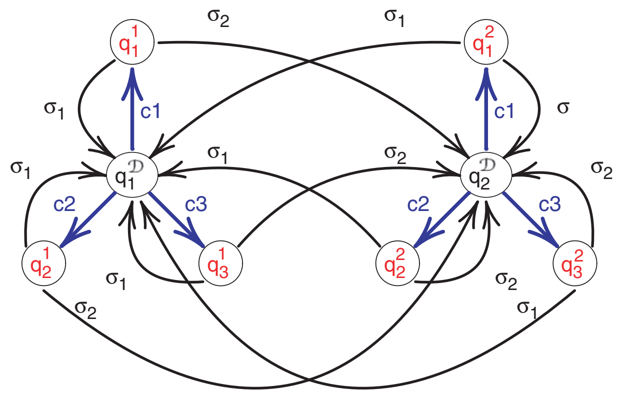

Figure 1. An example of navigation automaton with 3 command inputs and 2 discretised velocity states.

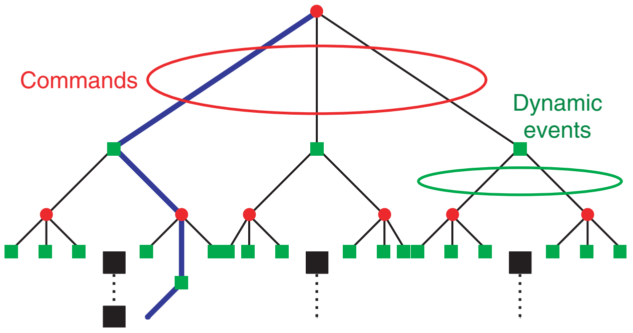

Figure 2. Operation of a navigation automaton.

the state is reachable from the decision state $q_i^{\mathcal{D}}$ with the command σ_j.

Next, we describe how the navigation of a supervised autonomous plant, similar to what is described at the beginning of the section, can be modelled by the sequential operation of a navigation automaton. The process is best explained by interpreting the transition sequences on a tree as shown in Figure 2, where the nodes and arcs represent the states and transitions, respectively. In Figure 2, the decision states are denoted by squares and the intermediate states by circles.

Let the robot be initiated at the root of the tree at a decision state, and a command is generated to pass a velocity set-point to the lower level of continuous dynamics. Due to inherent inertia, the robot does not, in general, attain the velocity set-point instantaneously. Let the dynamic events be reported by the robot at a preset frequency, say ζ. The velocity vector (i.e., linear velocity v and angular velocity ω in case of a planar robot) at the end of the time period belongs to one of the equivalence classes of the partitioned velocity space (see Definition 12).

Let the new decision state be $q_i^{\mathcal{D}}$, which is interpreted as a two-step transition sequence for the navigation automaton. First the automaton moves to an intermediate state by following the generated command and then makes the transition to the decision state $q_i^{\mathcal{D}}$ by a dynamic event. This is represented by first two segments, initiating from the root and denoted by the thick dashed path on the tree of Figure 2. The next command (i.e., velocity set-point) is generated from the current decision state and, in this way, the process is repeated. Thus, the operation of the plant, interpreted as the sequential operation of the navigation automaton, generates an unique path, denoted by a thick path in Figure 2, passing alternately through decision and intermediate states.

In the absence of a supervisor, a random sequence of commands is generated and the plant would execute a random walk with the navigation automaton following a random path down the tree. A supervisor can modify the path of the automaton by selectively disabling generated commands and thereby modify the physical trajectory of the robot. Generation of dynamic events from the intermediate states is a function of the robot dynamics and their effects are incorporated in the transition probabilities of the dynamic events generated from each intermediate state. For example, following Definition 2 in Figure 1, $\tilde{\pi}(q_2^1, \sigma_1)$ is the probability that the robot attains a velocity vector (i.e., a decision state) $q_1^{\mathcal{D}}$ being given a velocity set-point c_2 from the decision state $q_1^{\mathcal{D}}$. In the next section, we describe how to compute the transition probabilities for a specific system.

4.1 *Computation of transition probabilities*

The navigation automaton model specifies the transition probabilities of the commands from the decision states to be uniform, i.e.,

$$\tilde{\pi}(q_i^{\mathcal{D}}, c_j) = \frac{1}{|\mathscr{C}|} \quad \forall q_i^{\mathcal{D}} \in Q^{\mathcal{D}} \quad \forall c_j \in \mathscr{C}. \tag{7}$$

However, computation of the transition probabilities $\tilde{\pi}(q_i^{\mathscr{I}}, \sigma_j)$ of dynamic events $\sigma_j \in \Sigma^{\mathcal{D}}$ from intermediate states $q_i^{\mathscr{I}} \in Q^{\mathscr{I}}$ is more involved as it is dependent on the physical dynamics of the robot. Let the continuous dynamical model of the robot be denoted by $\mathscr{S}$.

Definition 14: Let x be the discrete-time state space vector of the mobile robot and $u = [v_c \quad \omega_c]$ be an input control vector of linear and angular velocities. Also let $y = [v \quad \omega]$ be the observed output velocites of the plant. Then the plant dynamics $\mathscr{S}$ at the time instant n is defined as the ordered pair $\mathscr{S} = (f, g)$ of continuous functions where

$$\begin{aligned} x_{n+1} &= f(x_n, u_n) \\ y_n &= g(x_n, u_n). \end{aligned}$$

The model $\mathscr{S}$ needs to be identified from experimental data through standard system identification techniques (Ljung 1999). The identified plant dynamics model $\mathscr{S}$ is utilised to obtain the event cost of the dynamic events as explained below.

Let the automaton be in a decision state $q_i^{\mathcal{D}}$ and a command c_j be generated, which results in an instantaneous transition to an intermediate state $q_k^{\mathscr{I}}$. Note that the velocity state of the robot must still belong to the equivalence class of $q_i^{\mathcal{D}}$ to preclude an infinite system acceleration. The command c_j has already passed a velocity set-point to the continuous level by mapping c_j to u_n through a discrete-event to continuous mapping. The current acceleration is unknown, which implies incomplete state information. Hence, there may be more than one state space trajectories that the robot may take and thus visit different decision states on its way, as seen in Figure 3.

The average settling time t_s of the system is estimated from the continuous plant model $\mathscr{S}$. Then, a Monte Carlo simulation experiment is performed by randomly choosing a velocity vector from the equivalence class of $q_i^{\mathcal{D}}$ and a random system acceleration vector. The robot system reports the decision states that it visits at a preset frequency. For each run, a symbolic string $\omega = q_{i_1}^{\mathcal{D}} q_{i_2}^{\mathcal{D}} \cdots q_{i_m}^{\mathcal{D}}$ with $q_{i_r}^{\mathcal{D}} \in Q^{\mathcal{D}} \forall r$ is obtained. The frequency probabilities of the various decision states are computed for each string. For a sufficiently large number of simulation experiments based on the identified model $\mathscr{S}$, the average of all runs yields the required transition probabilities $\tilde{\pi}(q_k^{\mathscr{I}}, \sigma_j)$ for $\sigma_j \in \Sigma^{\mathscr{D}}$. Algorithm 4 summarises the procedure.

Algorithm 4: Transition probabilities for dynamic events

```
input  : $Q^{\mathscr{I}}, \mathscr{S}$
output: State transition probability vector
        $\tilde{\pi}(q_i^{\mathscr{I}}, \cdot) := \tilde{\pi}(q_i^{\mathscr{I}}, \sigma_1) \cdots \tilde{\pi}(q_i^{\mathscr{I}}, \sigma_M)$ where
        $M = |\Sigma^{\mathscr{D}}|$
begin
  $k = 1$;
  Find $q_k^{\mathscr{D}}$ such that $\exists c_j \in \mathscr{C}$ with $q_k^{\mathscr{D}} \xrightarrow[c_j]{} q_i^{\mathscr{I}}$;
  while $p^k$ Not Converged do
    I. Choose random velocity vector $\nu \in [q_k^{\mathscr{D}}]$ such
       that $q_k^{\mathscr{D}} \xrightarrow[c_j]{} q_i^{\mathscr{I}}$;
    II. Choose random acceleration vector;
    III. Simulate plant up to average settling time $t_s$;
    IV. Obtain sequence of visited decision states
        $\omega = q_{i_1}^{\mathscr{D}} q_{i_2}^{\mathscr{D}} \cdots q_{i_M}^{\mathscr{D}}$;
    V. Compute current frequency probability vector q;
    VI. Compute mean frequency probability vector
        $p^k = \frac{1}{2}(p^{k-1} + q)$
  endw
  $\tilde{\pi}(q_i^{\mathscr{I}}, \cdot) = p^k$;
end
```

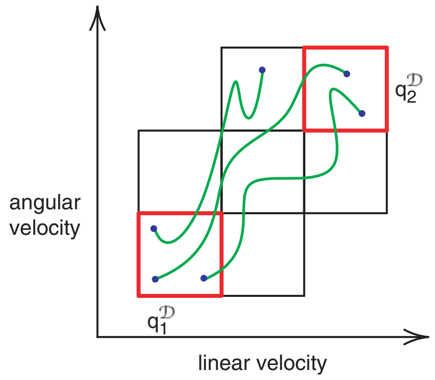

Figure 3. System trajectories with the identical initial decision state.

4.2 *Computation of the χ-vector*

As described in Section 3, computation of the optimal supervisor requires the characteristic vector χ as a time-dependent control specification that is updated as new sensor data becomes available. In general, χ can be a function of the current velocity $\overline{v}(t)$, local obstacle map $\Theta(t)$, goal heading $\Gamma(t)$ and a global map $\Omega(t)$ if available. The characteristic vector χ reflects how the supervised plant handles the sensor data. By suitable modifications, the plant behaviour can be made more or less aggressive. For the robotic experiments at hand, χ is selected to be a function of the observed local obstacle map. Specifically, the weight of a decision state is assigned a particular value based on the current intermediate state $q_j^{\mathscr{I}}$. All $q_j^{\mathscr{I}}$s reachable from $q^{\mathcal{D}}$s with the same command $c \in \mathscr{C}$ are assigned the same weight. If a global map and a goal is available, the characteristic weights (i.e., χ-values of the decision states) that take the system away from the goal are further penalised. Furthermore, external directional commands are incorporated from a higher level in the hierarchy, while still maintaining obstacle avoidance, by introducing selective bias on the characteristic weights of the decision states. Once the transition matrix parameters and the characteristic weights are available, an optimal supervisor that maximises the measure $\mu(\Pi, \chi)$ is computed and imposed. Thus, the trajectory is modified by selectively disabling the generated commands.

The proposed scheme is illustrated in Figure 4, where a change in the local environment of the robot causes the instantaneous control specification to change, thereby altering the supervisory decisions. The underlying assumption is that the environment does not change faster than the time required to compute the optimal supervisor at each step. This is not restrictive since the number of iterations required for the convergence of Algorithm 2 is sub-linear in n

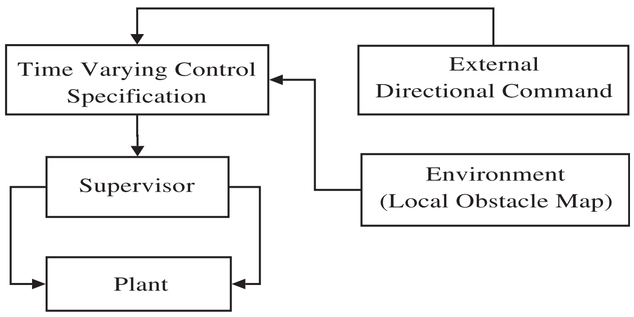
Time Varying Control Specification
External Directional Command
Supervisor
Environment (Local Obstacle Map)
Plant

Figure 4. Schematic representation of supervised navigation.

Chattopadhyay and Ray (2007), where n is the number of states in the automaton) and therefore can be executed efficiently. For example, the PLAYER server (Gerkey et al. 2003) executes data reporting at a frequency of 10 Hz (mostly to accommodate the standard laser range finders) while the described optimisation can be carried out at speeds well over 100 Hz even on a moderately powerful PC104 architecture.

4.3 *Livelock avoidance*

This section addresses the livelock avoidance (including deadlock avoidance). For a finite state automaton, livelock means existence of a set of unmarked states with no transition out of this state set. To detect the presence of a livelock, a (non-regular-language-based) observer is built to observe the events allowed by the obstacle avoidance controller. Whenever a string of events is accepted by this observer, the navigation automaton is said to be in a livelock. The observer is constructed with the following four basic ingredients.

1. A non-empty finite alphabet Σ of event symbols
2. A binary set $\{q_0, q_1\}$ of attributes, where q_0 indicates that an event does not belong to a livelocked string and q_1 indicates that an event does belong to a livelocked string
3. A special counting symbol B where $\Sigma \cap \{\mathrm{B}\} = \emptyset$
4. An event-driven state transition function δ

Definition 15: A livelock observer is defined by the following three-stack pushdown automaton:

$$\mathscr{O}_l = \{Q, \Sigma, \Gamma_1, \Gamma_2, \Gamma_3, \delta, Q_m\}$$

where $Q = \{q_0, q_1\} \times \Sigma$ is the set of states and $Q_m = \{q_1\} \times \Sigma$ is the set of all accepted states; Σ is the event alphabet; $\Gamma_1 = \Sigma$ is the first stack alphabet; $\Gamma_2 = \{\mathrm{B}\}$ and $\Gamma_3 = \{\mathrm{B}\}$ are the second and third stack alphabets; and $\delta\colon (Q \times \Gamma_1^\star \times \Gamma_2^\star \times \Gamma_3^\star) \times \Sigma \to (Q \times \Gamma_1^\star \times \Gamma_2^\star \times \Gamma_3^\star)$ is the set of transitions defined as follows:

$$([q_0\ \sigma], \omega, B^k, B^n) \xrightarrow{\eta} ([q_0\ \eta], \omega\eta, B, B^{n+1}) \quad \sigma \neq \eta \text{ and } n < N$$

$$([q_0\ \sigma], \omega, B^k, B^n) \xrightarrow{\sigma} ([q_0\ \sigma], \omega, B^{k+1}, B^n) \quad k < K \text{ and } n < N$$

$$([q_0\ \sigma], \omega\gamma, B^k, B^n) \xrightarrow{\sigma} ([q_0\ \sigma], \omega, \varepsilon, B^{n-1}) \quad k = K \text{ and } n > 0$$

$$([q_0\ \sigma], \omega, B^k, B^n) \xrightarrow{\eta} ([q_1\ \eta], \omega\eta, B, B^{n+1}) \quad \sigma \neq \eta \text{ and } n \geqslant N$$

$$([q_1\ \sigma], \omega, B^k, B^n) \xrightarrow{\sigma} ([q_1\ \sigma], \omega, B^{k+1}, B^n) \quad k < K \text{ and } n \geqslant N$$

$$([q_1\ \sigma], \omega\gamma, B^k, B^n) \xrightarrow{\sigma} ([q_1\ \sigma], \omega, \varepsilon, B^{n-1}) \quad k = K \text{ and } n \geqslant N$$

$$([q_1\ \sigma], \omega\gamma, B^k, B^n) \xrightarrow{\sigma} ([q_0\ \sigma], \omega, \varepsilon, B^{n-1}) \quad k = K \text{ and } n < N$$

where the parameters K and N are chosen based on the problem at hand and need to be tuned to control the behaviour of the robot.

It is well known that a push-down automata with more than one stack has the same expressive power as a Turing machine (Hopcroft et al. 2001). The livelock observer is modelled by three stacks to represent what actually happens in the computation procedure. The first stack, known as the event stack, stores the events and the length of this stack determines whether a livelock is detected or not. The second stack, called event multiplicity stack, counts the occurrence of the last event while the third stack, called the counting stack, counts the number of elements in the event stack. The observer remembers the last occurred event using the composite state in Q. The transitions of the observer perform the following functions in the order listed in Definition 15.

1. As a new event ($\eta \neq \sigma$) is added to the *event stack*, the counter stack is incremented by one by pushing the counting symbol B. The event multiplicity stack is cleared and B is pushed on to it. The observer changes the state from $\{q_\ell, \sigma\}$ to $\{q_\ell, \eta\}$, $i = 0$ or 1, to remember the last event.
2. If new event is the same as the last event, the multiplicity of the last event is incremented

by pushing the counting symbol $B \in \Gamma_2$ on to the second stack. Note that stacks 1 and 2 remain unchanged.

3. If the multiplicity of the last event is $k = K$, then the topmost element of the event stack is popped out, the event multiplicity stack is cleared, and the size of event stack is decremented by popping out B from the counter stack.
4. If the length of the event stack is $n = N$, then the observer switches its state to q_1 and accepts the string (i.e livelock is detected). It remains in q_1 as long as $n \geqslant N$ and switches back to q_0 whenever the stack length becomes less than N. All other transitions from q_1 are similar to those when the observer is in q_0.

The livelock observer records all new events occurring in the event stack and accepts the string if the event stack's length is greater than N. Thus, it essentially captures alternating events as in the string 01010101010. Since it is undesirable to have strings like $0^n1^n0^n1^n$ for large n, the top symbol of the event stack is erased if the multiplicity of the last symbol is greater than K. For example, let the event alphabet be $\Sigma = \{0, 1\}$, $K = 3$, and $N = 5$. The string 0010011001010 would be accepted by the machine representing a livelock whereas the string 00000001111111000000001111111100000000 would not be accepted. Once a livelock is detected it is a simple matter to avoid it by randomly choosing one of the controllable events in the live-lock string and writing it as an external command to the obstacle avoidance controller. The main results on livelock observer are summarised below.

- Livelock detection and monitoring in real time: The decisions are made from the observer states.
- Recovery from a livelock: Upon detection of a livelock, the navigation algorithm is executed to circumvent the livelock; this feature also includes avoidance of forthcoming livelocks.

It is important to note that the above methodology is not exclusive in the sense that there exist parallel approaches for effective detection and recovery from decision saddle points. The above details demonstrate that decision livelocks can be easily addressed in practical implementations.

5. Implementation on a Segway RMP

The model of a Segway robotic mobile platform (RMP) was obtained by using the system identification toolbox for MATLAB. The robot is excited with a psuedo-random input in its typical work environment and the output measured by the onboard sensors was recorded. The input is constrained as follows.

- $1\,\text{sec} \leqslant t_d \leqslant 6\,\text{sec}$;
- $-0.8\,\text{m/s} \leqslant v_c \leqslant 0.8\,\text{m/s}$;
- $-0.4\,\text{rad/s} \leqslant \omega_c \leqslant 0.4\,\text{rad/s}$;

where v and ω are the commanded linear and angular velocities of the robot which are held constant for a period of t_d seconds. The measured outputs for linear velocity v and angular velocity ω of the onboard sensors were recorded. Since the Segway has the dynamics of an inverted pendulum, an onboard balancing controller is always active which introduces a lot of high frequency disturbances in the response. The subspace method for system identification (Hayes 1996) is chosen to largely attenuate these disturbances. To choose the model order we follow standard techniques such as the Akaike's information criterion (AIC) (Hayes 1996). A fourth order model was found to be a good balance between prediction error and modelling complexity as shown in Figure 5. The resulting state space model with the input vector $u = [v_c \quad \omega_c]^T$ and the output vector $y = [v \quad \omega]^T$ is given by

$$x_{n+1} = Ax_n + Bu_n \tag{8a}$$

$$y_n = Cx_n + Du_n, \tag{8b}$$

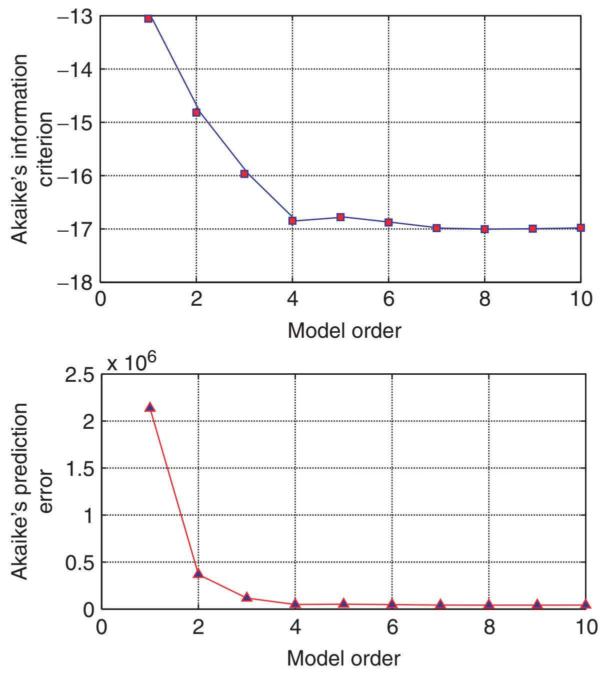

Figure 5. Model order selection.

where the matrices A, B, C, D are as follows:

$$A = \begin{pmatrix} 0.9624 & -0.0263 & 0.1190 & 0.0198 \\ -0.0293 & 0.6130 & 0.0242 & 0.2175 \\ -0.1277 & -0.0798 & 0.7695 & -0.2559 \\ -0.0052 & -0.4793 & -0.2170 & -0.4911 \end{pmatrix}$$

$$B = \begin{pmatrix} -0.0082 & -0.0003 \\ 0.0013 & 0.0852 \\ 0.0368 & 0.0807 \\ -0.0240 & 0.4032 \end{pmatrix}$$

$$C = \begin{pmatrix} 5.3435 & -0.3961 & -0.1457 & 0.0926 \\ 0.1204 & 1.4111 & 0.0013 & -0.2697 \end{pmatrix}$$

$$D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Figure 6 shows the input, measured response and the response of the 4th order model that was fit to the data.

Simulation results show that 8 decision states are found to be sufficient for the navigation automaton. The central points of the corresponding equivalence classes are shown in the first three columns of Table 1. The commands or the velocity set points are chosen to be the central points of the classes as shown in the fifth column. Hence we had 8 dynamic events, 8 commands, 8 decision states resulting in a total $8+8\times 8=72$ states for the navigation automaton. The transition probabilities were estimated as described in Algorithm 4. The first few rows of the $\tilde{\pi}$ matrix in Table 2 show transition probabilities of the dynamic events corresponding to intermediate states. Row 9 (in bold) shows probabilities of commands from the decision state 1. The $\widetilde{\Pi}$ matrix is partially listed in Table 2; the complete matrix could not be shown due to space limitation.

The χ vector is chosen based on the local obstacle map as sensed by the onboard sonars. The segway RMP has 3 sonars in the front and 3 sonars in the back arranged as shown in Figure 7. The weight for decision states $q_i^{\mathcal{D}}$ is chosen to be zero since a decision state is reached through intermediate states $\{q_i^j\}$, $i=1,\ldots,|Q^{\mathcal{I}}|$ via uncontrollable transitions. The weight for each intermediate state q_j^i $\forall i = 1,\ldots,|Q^{\mathcal{D}}|-1$ is set to the same value as q_j^0. However, the weights for q_j^i and q_k^i $j\neq k$ are different. The weights for states $\{q_j^i\}$ for $i=1$ and $j=0\ldots7$ are as shown in Table 3 (see Table 1 for nomenclature of commands $\mathscr{C}$), where the distances $d_0\ldots d_5$ are the instantaneous readings of sonars 1 … 6 of the Segway RMP and TH=2.0 meters and the tertiary operator "?:" is the same as in C/C++ languages. For example, result = (condition?x:y) means result = x if condition is true, otherwise result = y.

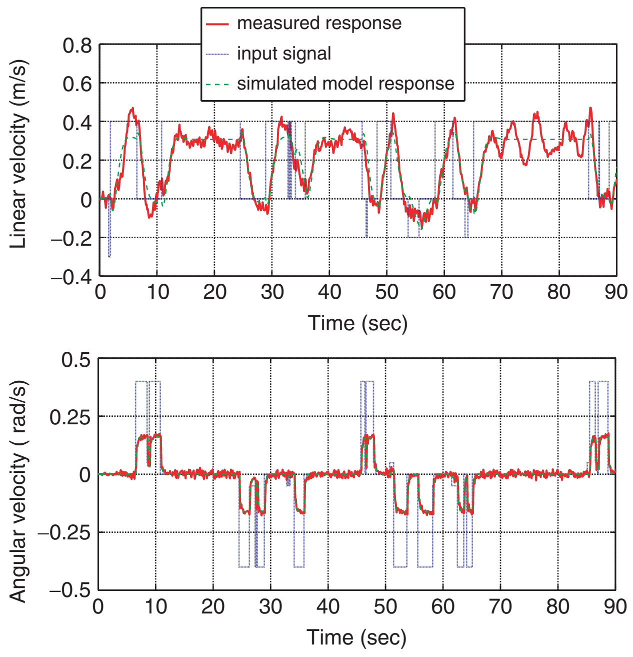

Figure 6. Response plots for system identification.

Table 1. Decision states and the respective commands.

$q_i^{\mathcal{D}}$	v_i	ω_i	c_i	Commands $\mathscr{C}\subset\Sigma$	Description
$q_0^{\mathcal{D}}$	0	0	c_0	X	Stop
$q_1^{\mathcal{D}}$	0.2	0	c_1	F	Forward
$q_2^{\mathcal{D}}$	0.5	0	c_2	FF	Fast forward
$q_3^{\mathcal{D}}$	−0.2	0	c_3	B	Back
$q_4^{\mathcal{D}}$	0	0.25	c_4	L	Turn left
$q_5^{\mathcal{D}}$	0	0.5	c_5	LL	Turn fast left
$q_6^{\mathcal{D}}$	0	−0.25	c_6	R	Turn right
$q_7^{\mathcal{D}}$	0	−0.5	c_7	RR	Turn fast right

Table 2. Sample entries of $\widetilde{\Pi}$.

	σ_0	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7
q_0^1	0.959	0.000	0.000	0.001	0.000	0.000	0.000	0.000
q_1^1	0.024	0.935	0.000	0.001	0.000	0.000	0.000	0.000
q_2^1	0.016	0.015	0.927	0.002	0.000	0.000	0.000	0.000
q_3^1	0.023	0.001	0.000	0.936	0.000	0.000	0.000	0.000
q_4^1	0.004	0.000	0.000	0.000	0.956	0.000	0.000	0.000
q_5^1	0.004	0.000	0.000	0.000	0.001	0.955	0.000	0.000
q_6^1	0.004	0.000	0.000	0.000	0.000	0.000	0.956	0.000
q_7^1	0.004	0.000	0.000	0.000	0.000	0.000	0.001	0.955
	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7
$q_1^{\mathscr{D}}$	**0.120**	**0.120**	**0.120**	**0.120**	**0.120**	**0.120**	**0.120**	**0.120**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

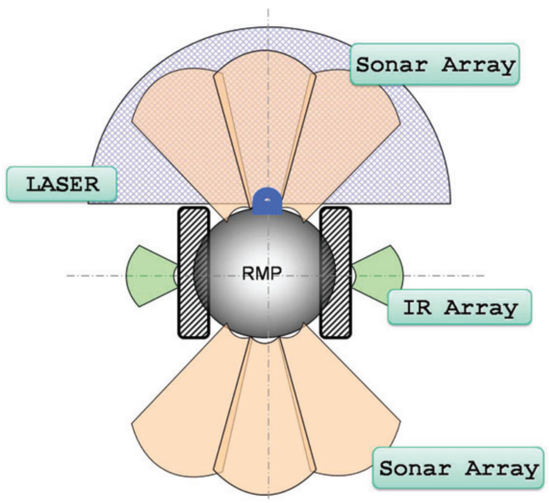

Figure 7. Sensor array on Segway RMP.

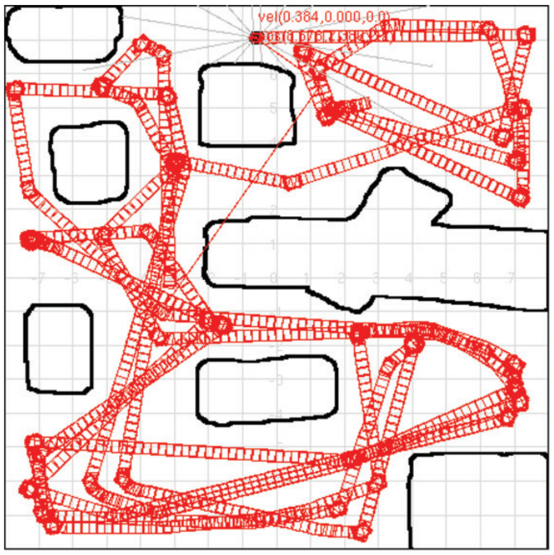

Figure 8. Simulated Segway robot motion on a STAGE simulator.

Table 3. Weights (χ) for few sample states.

q	Characteristic weight $\chi(q)$	Outcomes
q_0^1	-8.0	High neg wt for X
q_1^1	$(d_1 > TH?(d_1 - 0.3):(d_1 - 0.3)*1.8)$	high pos wt for F
q_2^1	$(\min(d_1, d_0, d_2) > TH?d_1*3.0:0.0)$	pos wt for FF
q_3^1	$d_4 - 2.2$	B
q_4^1	$d_0 - 0.05$	L
q_5^1	$\min(d_0, d_3) - 0.01$	LL
q_6^1	$d_2 - 0.06$	R
q_7^1	$\min(d_2, d_5) + 0.01$	RR

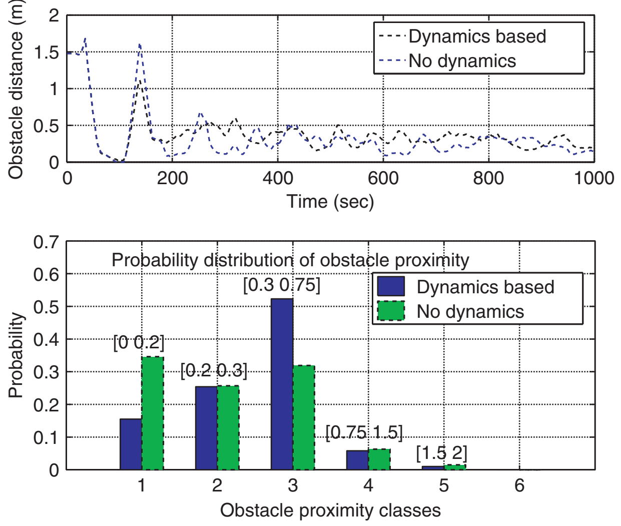

Figure 9. Effect of considering vehicular dynamics and otherwise.

5.1 *Simulation and implementation results*

The proposed algorithm has been extensively simulated on the stage simulator with `player` robot server (Gerkey et al. 2003). Figure 8 shows typical trajectories in an environment with obstacles. The algorithm has been successfully implemented on several robotic platforms including a Segway RMP robot that is instrumented with a variety of sensors.

Figure 9 illustrates the effects of including vehicular dynamics, where normalised histograms of probability distribution are compared based on the same data set from 100 runs for the two cases: (i) vehicular dynamics included, and (ii) vehicular dynamics not included. The results indicate that the probability of being in the obstacle vicinity (e.g., within 0.2 m) reduces to half if the effects of vehicle dynamics are accounted for.

Figures 10(a) and 10(b) respectively show the performance of obstacle avoidance without and with livelock avoidance. The plots in Figure 10 show the position (x, y, θ) of the robot with time. Without any livelock avoidance, the robot may become stuck in a livelock (e.g., the robot keeps moving forward and backward as indicated by unchanging θ as seen in Figure 10(a)). In contrast, with the livelock observer monitoring the robot's behaviour, the livelocks are completely avoided as seen in Figure 10(b). The experimental results establish the following features of the navigation algorithm and the livelock observer.

- Robustness under disturbances: This feature includes sensor faults (e.g., noisy and intermittent sensors).

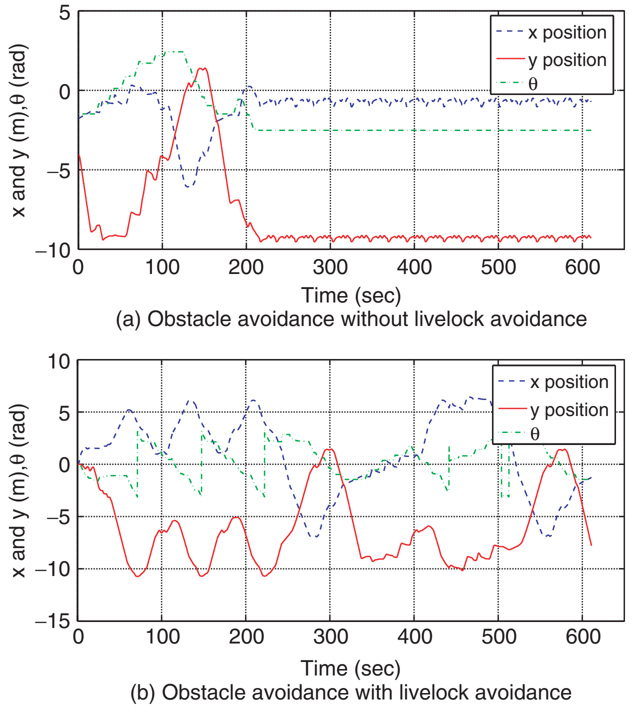

Figure 10. Robot motion profile: x position, y position, and angle θ. Top plate (no livelock observer): Livelock from 210 sec onwards. Bottom plate (with livelock observer): Normal operation.

- Insignificant computational cost: This feature ensures superior robot behaviour in real time.
- Local convergence: This feature alleviates the problem of being 'stuck in local optimisation minima', which is typical for purely continuous techniques.
- Fine control of robot behaviour: This feature is made possible through assignment of the characteristic vector, which is superior to 'safety distance' assignment that is usually adopted in robot control.

6. Summary, conclusions, and future research

This paper formulates and experimentally validates a novel concept for autonomous intelligent navigation of mobile robotic platforms, especially in unmapped dynamic environments. The navigation algorithm is formulated based on language-measure-theoretic optimal control of probabilistic regular languages (Chattopadhyay and Ray 2007). Real-time sensor data and model-based information on the robot's motion dynamics are fused to construct a probabilistic finite state automaton model that dynamically computes a time-dependent discrete-event supervisory control policy. Emphasis is laid on reduction of computational load and algorithmic complexity in the synthesis of the discrete-event supervisory control algorithm. The basic behaviour of mobile robotic platforms has been modelled as a probabilistic finite state automaton (PFSA). At each step, its future performance is computed in advance based on (possibly) time-dependent characteristic weights. The robust supervisory control algorithm is updated and exercised in real time according to specifications that may be dynamically updated.

The paper also addresses the issue of livelock avoidance by constructing an observer to monitor the behaviour of the navigation algorithm. In the event that a livelock is detected, the observer sends external commands to the navigation algorithm to circumvent the livelock. The performance and robustness of the autonomous intelligent navigation algorithm have been experimentally validated on Segway RMP robotic platforms.

The future research plan includes application of language-theoretic formalism for global path planning as an extension of reactive local path planning. Initial studies suggest a possible hierarchical structure, where the robot environment is partitioned into grids. In this approach, a global path planner at the upper level would plan paths from grid A to grid B in the map, where the lower lever may execute a navigation algorithm, such as the one presented in this paper, for local path planning. The two levels may operate independently except when the global planner commands the lower level to execute the plan. Future research may also incorporate feedback from the lower level during the computation of optimal paths.

Acknowledgements

This work has been supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant Nos. W911NF-07-1-0376 and W911NF-06-1-0469.

References

Brock, O., and Khatib, O. (1999), 'High-speed Navigation using the Global Dynamic Window Approach', *ICRA*, Detroit, MI, Vol I, 341–346.

Chattopadhyay, I., Chakraborty, S., and Ray, A. (2005), 'Intelligent Navigation in Space', *AIAA Aerospace Infotech Proceedings*, pp. AIAA–2005.

Chattopadhyay, I., and Ray, A. (2006), 'Renormalised Measure of Regular Languages', *International Journal of Control*, 79, 1107–1117.

Chattopadhyay, I., and Ray, A. (2007), 'Language-measure-theoretic Optimal Control of Probabilistic Finite-state Systems', *International Journal of Control*, 80, 1271–1290.

Fox, D., Burgard, W., and Thrun, S. (1997), 'The Dynamic Window Approach to Collision Avoidance', *IEEE Robotics & Automation Magazine*, 4, 23–33.

Fujimura, K. (1991), *Motion Planning in Dynamic Environments*, New York, NY, USA: Springer-Verlag.

Garg, V. (1992) 'An Algebraic Approach to Modeling Probabilistic Discrete Event Systems', *Proceedings of 1992 IEEE Conference on Decision and Control*, Tucson, AZ, pp. 2348–2353.

Garg, V. (1992) 'Probabilistic Languages for Modeling of DEDs', *Proceedings of 1992 IEEE Conference on Information and Sciences*, Princeton, NJ, pp. 198–203.

Gerkey, B., Vaughan, R., and Howard, A. (2003), 'The Player/Stage Project: Tools for Multi-robot and Distributed Sensor Systems', *Proceedings of the International Conference on Advanced Robotics (ICAR 2003)*, Coimbra, Portugal, June 30–July 3, pp. 317–323.

Hayes, M. (1996), *Statistical Digital Signal processing and Modelling*, New York, NY, USA: John Wiley & Sons.

Hopcroft, J.E., Motwani, R., and Ullman, J.D. (2001), *Introduction to Automata Theory, Languages, and Computation* (2nd ed.), Boston, MA, USA: Addison-Wesley.

Hwang, Y.K., and Ahuja, N. (1992), 'Gross Motion Planning—a Survey', *ACM Computing Surveys*, 24, 219–291.

Kokotovic, P., Khalil, H., and O'Reilly, J. (1999), *Singular Perturbation Methods in Control Analysis and Design*, Philadelphia, PA: Society of Industrial and Applied Mathematics.

Koren, Y., and Borenstein, J. (1991) 'Potential Field Methods and their Inherent Limitations for Mobile Robot Navigation', *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1398–1404.

Ljung, L. (1999), *System Identification* (2nd ed.), Upper Saddle River, NJ, USA: Prentice Hall PTR.

Lozano-Perez, T. (1983), 'Signal Planning–a Configuration Space Approach', *IEEE Transactions on Computers*, C-32, 108–120.

Mallapragada, G., Chattopadhyay, I., and Ray, A. (2006) 'Autonomous Navigation of Mobile Robots Using Optimal Control of Finite State Automata', *IEEE Conference on Decision and Control (CDC)*, San Diego, pp. 2400–2405.

Ray, A. (2005), 'Signed Real Measure of Regular Languages for Discrete-event Supervisory Control', *International Journal of Control*, 78, 949–967.

Ray, A., Phoha, V., and Phoha, S. (2005), *Quantitative Measure for Discrete Event Supervisory Control*, New York: Springer.