The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Mixing Differential Inclusions with Markov Decision Processes

Nelson Gonçalves and João Sequeira

Abstract— **This paper describes a behavioral approach to the synthesis of controllers for the execution of tasks by mobile robots. The paradigm integrates two distinct methods for decision making under uncertainty, namely Markov decision processes and differential inclusions. The target application is surveillance systems where mobile robots can approach intruders to take their mugshot or block exit pathways. An hybrid model of the system is described and stability conditions discussed. The main ideas are exemplified in a search and pursue application and some preliminary experimental results with a navigation behavior are also presented.**

I. Introduction

This work proposes an approach to the synthesis of controllers for the execution of tasks by mobile robots. The paradigm is based on the mix of Markov decision processes to model the environment and differential inclusions to model robot behaviors. The target application is surveillance, where the robots must execute a set of heterogeneous tasks, e.g., patrolling the environment, searching for intruders and intercepting them, blocking their exit pathways, and moving in formation. The purpose of the latter is to increase the effectiveness of the robots. For instance, a group of robots moving in the adequate formation can block the exit of an intruder more effectively than a single robot.

The environment where surveillance takes place is adversarial, with unstructured dynamics and subject to uncertainties that affect its state. The execution of tasks by the robots is thus formulated as a problem of real-time decision making under uncertainty. The proposal in this work is to model the execution of tasks with two different, but complementary frameworks. The two resulting models are integrated then into a single, hybrid model that the robot will use to execute a task.

The framework of Markov decision processes (MDPs) is employed to model the dynamics of the environment, estimated from the outcomes of the behaviors by the robots during a task execution. This is a mathematically sound approach to sequential decision making under uncertainty [1]. It is also a natural choice because MDPs provide a first order approximation to the human decision making process [2].

Within MDPs, uncertainty is identified with the lack of knowledge on which environment state results from the actions of the robots. Uncertainty is represented through stochastic state transition functions. It can also be extended to include partial state observability [3] or poor knowledge of the state transition functions [4].

The kinematics of the robots, during the execution of the task, are modeled using differential inclusions (DIs) (see for instance [5]), representing robot behaviors (see [6] for an example of the synthesis of navigation behaviors). The idea is to constrain the robot pose to a convex and compact region while tracking a velocity reference. Reference poses and paths for the robot are not required to be known explicitly when modeling behavior by DIs. Instead, only bounds for the desired poses or path are used. The advantages of using DIs are twofold. First, surveillance applications usually do not require the precise specification of reference poses for robots; in fact when a robot is trying to block an exit the key factor is how intruders perceive the intention of the robot. Second, specifying robot behaviors through bounding regions implicitly enables the embedding of some the uncertainty and avoids requiring accurate world maps and localization.

Navigation behaviors are synthesized by specifying the constraints the robot trajectories should verify, instead of computing optimal trajectories. In a sense, behaviors can be synthesized with less information. For instance, the search for an intruder can simply be constrained to a bounded region instead of computing an optimal path from the sensors on the robot, the map of the environment and the motion model of the intruder.

The two frameworks are complementary as they model different aspects of task execution. The MDPs consider the dynamics of the environment while the robot kinematics and dynamics are addressed by the DIs. The two frameworks also account differently for uncertainty in task execution. In MDPs, it is not known which environment state will result from the actions of the robots. This uncertainty is structured because the state transitions, as function of the robot actions, are stochastic and with known distributions. In what concerns behavior modeling using DIs, the robot pose can take any value as long as it remains inside a convex and compact, bounding region, [6]. Thus in the DIs framework, uncertainty can be identified with ambiguity in the selection of a path for the robot.

The models obtained with the two frameworks are integrated into an hybrid model (see for instance [7] on hybrid systems). The discrete variables represent the state of the environment and the navigation behaviors which the robot can produce. The continuous variables represent the pose and velocity of the robot. They are determined by the navigation behavior which is currently active in the robot. The execution of tasks is thus modeled as the sequential activation of

N. Gonçalves is with Instituto Politécnico de Tomar, Tomar and Institute for Systems and Robotics, Lisbon, Portugal. J. Sequeira is with Instituto Superior Técnico and Institute for Systems and Robotics, Lisbon, Portugal. Email: {*ngoncalves*, *jseq*}@isr.ist.utl.pt

This work was supported by ISR/IST plurianual funding through the POS_Conhecimento Program that includes FEDER funds.

navigation behaviors with the purpose of producing a desired state of the environment.

The remainder of this paper is as follows. The synthesis of navigation behaviors for the mobile robots using DIs is detailed in Section II. It is followed by a brief description of the MDP framework in Section III. The two frameworks are integrated into a single, hybrid model for the execution of tasks in Section IV. In this section are also derived stability conditions for the hybrid task models. The approach is applied in Section V to a search and pursue problem. The paper is concluded with Section VI, with a brief summary and a discussion of the proposed behavioral approach .

II. Synthesis of Navigation Behaviors

The class of robots considered in this paper is composed by wheeled mobile robots moving on a 2D plane. Their kinematics are modeled by a differential equation

$$\dot{q} = f(q,u) \tag{1}$$

where $q \in \mathbb{R}^n$ is the robot pose in a frame of reference, $u \in \mathbb{R}^m$ are the control inputs and f is a Lipschitz continuous function. This is a suitable model for common wheeled robots, such as the unicycle, and vehicles with limited minimum turning radius. In practice, the control inputs available to the mobile robot are bounded, which is modeled by the differential inclusion (DI)

$$\dot{q} \in F(q,U) = \{p \in \mathbb{R}^n, u \in U \mid p = f(q,u)\} \tag{2}$$

where F is a set-valued map and U is a compact subset of $\mathbb{R}^m$. Assuming f to be Lipschitz continuous, from Proposition 2.4 in [5] it can be asserted that F is a Lipschitz continuous set-valued map. A strictly positive Lipschitz constant is assumed and the output values of F are convex, closed and contained in a ball centered at the origin. DIs with such set-valued maps are denoted *Lipschitzian*, [5]. In the remainder of the paper, the kinematics of the mobile robots are modeled by the differential inclusion (2).

The method for behavior synthesis is based in constraining the robot pose to a compact and convex region during a finite time interval $[t_0, t_1]$, $t_1 < \infty$.

Definition 1 (Viable Set): A viable set is represented by the output values of the Lipschitzian set-valued map $C(t)$, $C : \mathbb{R}^n \times [t_0, t_1] \to \mathbb{R}^n$. The robot pose is viable at time t if $q(t) \in C(t)$. The trajectory of a robot is viable if the pose is viable for all $t \in [t_0, t_1]$.

In mobile robotic applications, the obstacle free space surrounding a mobile robot is the natural candidate from where to extract viable sets. In another example, when the robot is pursuing an intruder, a natural viable set is the compact region in free space containing both the mobile robot and the intruder, if it exists. Other bounding regions include hallways and rooms, where surveillance missions are taking place. Depending on the mission requirements, the viable sets can be static or time-variant.

In general, a robot can execute an infinite number of viable trajectories. However, not all of them correspond to navigation behaviors useful to surveillance missions. Consider, for example, the navigation behavior of patrolling a region modeled by a static viable set, and assume that the initial pose of the robot is viable. In this case, the trivial trajectory, with $\dot{q} = 0$, is viable but is not suitable for the purpose of the task. For this reason the robots are also required to track (but not necessarily follow) a suitable velocity reference $\dot{q}_{ref}(t)$. Returning to the case of the patrolling behavior, this reference can lead the robot to produce a spiral shaped trajectory.

Definition 2 (Navigation Behavior): A navigation behavior is the tuple $B = (C, \dot{q}_{ref}, t_0, t_1)$ where $C(t)$ is the viable set, $\dot{q}_{ref}(t)$ is the desired velocity reference and $[t_0, t_1]$ is the temporal duration of the behavior.

The trajectory that the robot must follow is determined by projecting the desired velocity reference, $\dot{q}_{ref}$, in the set of viable directions of motion $V(.)$

$$\dot{q} \in V(q,U,C) \tag{3}$$

where the set-valued map $V(.)$ is such that any solution of the (3) is invariant in the set $C(t)$. That is, if initially $q(t_0) \in C(t_0)$ then all of the solutions of (3) remain inside $C(t), t \in [t_0, t_1]$.

Proposition 1 (Viable Directions of Motion): Assume that initially the robot pose is viable, $q(t_0) \in C(t_0)$. Let $\mathscr{D}C(t)(1,q)$ be the tangent derivative of $C(t)$ at point q and in the direction of increasing time. Then any solution of the differential inclusion

$$\dot{q} \in V(q,F,C) = F(q,U) \cap \mathscr{D}C(t)(1,q) \tag{4}$$

is viable for $t \in [t_0, t_1]$ if $F(q,U) \cap \mathscr{D}C(t)(1,q) \neq \emptyset$.

Proof: Assume the existence of solutions of (4) for any initial pose inside the viable set, $q(t_0) \in C(t_0)$. Let $\tilde{q}(t)$ be a non-viable solution of (4), with $\tilde{q}(t_0) \in C(t_0)$. Then, for this non-viable solution, there is a time instant $\tau \in [t_0, t_1]$ where it leaves the viable set. That is, at instant $\tau \in [t_0, t_1]$ a non-viable direction of motion is selected, $v \in V(.)$. However, by construction the tangent derivative contains only the viable directions of motion and therefore $v \notin \mathscr{D}C(\tau)(1,\tilde{q})$. Thus no solution of 4) can leave the viable set. The existence of solutions for (4) is asserted as follows. The tangent derivative is a map in the form $\mathscr{D} : grC \to \mathbb{R}^n$, where grC is graph of $C(t)$. By construction, the output values are closed cones. The set-valued map $F(q,U)$ is Lipschitzean with compact output values, hence closed. From Proposition 2.3 in [5] the set-valued map $V(.) = F(.) \cap \mathscr{D}(.)$ is upper semi-continuous for any point $(t,q) \in grC$. Now we are in the conditions of Theorem 5.4 in [5], and solutions to (4) exist if $F(q,U) \cap \mathscr{D}C(t)(1,q) \neq \emptyset$. ■

The expression of the set-valued map in (3) is thus the intersection between the set-valued maps of the robot kinematics (2) and the tangent derivative of $C(t)$. The latter can be understood as the set of directions of motion along which the robot can travel without leaving $C(t)$.

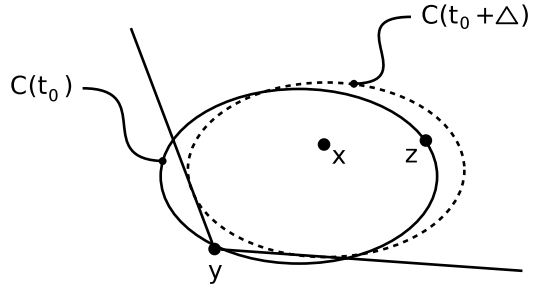

Fig. 1. Geometric interpretation of the tangent derivative.

Figure 1 depicts a geometric interpretation of the tangent derivative (check [5] for a formal definition). Let x, z and y be different viable robot poses at t_0 and let Δ be an arbitrarily small value. The tangent derivative is then the set of directions of motion $p \in \mathbb{R}^n$ such that the robot pose will remain viable for a small displacement of the viable set. If the robot pose is in the interior of the viable set, point x, then any direction of motion can be selected by the robot. However, if the robot pose is on the border of the viable set, points y or z, only the directions which belong to the cone containing $C(t_0+\Delta)$ can be selected. Since the point z is in the interior of $C(t_0+\Delta)$, the cone degenerates to $\mathbb{R}^n$.

When the viable set is static the tangent derivative is simply the tangent cone of the set, [5]. Any direction of motion is viable at the points in the interior of the set. For the points in the border of the set, only the directions that point to the interior of the set or are tangent to the border of the set.

At each time instant, the output value of $V(.)$ is then set of directions of motion such that the robot pose will remain viable. The direction of motion that the robot will follow is determined by projecting $\dot{q}_{ref}$ on to the output of $V(.)$:

$$v(t) = \begin{cases} 0, & if\ V(q,F,C) = \emptyset \\ Proj(\dot{q}_{ref}(t), V(q,F,C)), & otherwise \end{cases} \quad (5)$$

where $Proj(.)$ is the euclidean projection of a point to a set and $v(t)$ is the selected direction of motion. The output values of $V(.)$ are convex and compact by construction, therefore the direction $v(t)$ is always uniquely defined for $t \in [t_0, t_1]$.

Expression (5) reduces to the robot kinematics (2) if the robot pose is in the interior of the viable set. This is because any direction of motion is viable when $q(t) \in intC(t)$, since the tangent derivative degenerates and is equal to $\mathbb{R}^n$. Although in general it is not trivial to compute the tangent derivative, the main practical difficulty with this approach occurs when the robot pose is close to or on the border of the viable set. In practice, it is difficult to obtain an accurate estimate of the robot pose and only an estimative is available, affected by sensor noise. Thus, when it is close to the border of the viable set, the pose estimate can quickly switch between the viable and non-viable conditions. This is undesirable because the robot will stop instead of moving along the border of the viable set.

Since it is not possible to eliminate the noise and uncertainty present in the estimation of the robot pose, this problem will always be present. An alternative is to use the robot body frame of reference because the pose in this frame is always constant. Since the pose is known, the evaluation of (5) is not susceptible to the pose estimation errors. However, this is at the cost of limiting the range of behaviors to those that can be expressed in the robot body frame. That is, both $\dot{q}_{ref}$ and $C(t)$ must be expressed in the robot body frame.

Nevertheless, if the viable set is determined from the onboard sensors of the robot, such as range devices, it is also affected by noise and uncertainty. It is possible to verify the convexity and Lipschitzean conditions by filtering and smoothing the sensors inputs, but at the cost of additional processing time and more conservative estimates for the viable set.

III. Sequential Decision

The execution of a task by a mobile robot is understood as the sequence of actions it makes with the purpose of reaching a desired environment state. Only the tasks which can be executed without knowledge of the state past history are considered in this work. More complex tasks can be also be accomplished by first decomposing them into smaller subtasks of the type considered.

It is assumed that the robots know only the current state of the environment and the actions they can perform in that state. Furthermore, the actions of the robot produce a change of the state in the environment. This is modeled by a stochastic transition function, specific to each task and robot. Since the surveillance environments are dynamic, the robot must decide in real-time the action to perform given the current state it perceives.

The execution of a task by a robot is thus modeled by a Markov Decision Process (MDP), and a brief description of this framework is presented. A more inclusive description can be found in [1], for instance. An MDP is represented by the tuple,

$$m = (S, A, p, \varphi) \quad (6)$$

where S is the discrete, finite set of states of the environment, A is a finite set of actions, $p(.)$ is the state stochastic transition function and $\varphi(.)$ is the cost of transition between states using a given action. The environment dynamics are therefore modeled by the set of states S and the transition function $p(.)$, and include the interactions between the robots and the environment.

The environment state is an abstract concept, meaningful only in the specific context of each task. In general, the environment states can be mapped directly onto the environment features which the robot can perceive. For instance, the set of states can correspond to a region on a map or the relative position of an intruder with respect to the robot. Thus the set of environment states and the transition function can be understood as an instance of the information spaces described by Lavalle in [8].

The transition function $p(.)$ is defined as the probability of reaching state s' given that the action a is executed in state s, $p : S \times A \times S \to [0,1]$. The uncertainties about the environment dynamics are accounted for in this function. The cost function $\varphi(.)$ is defined as the immediate numerical value collected by the environment because action a was performed in state s, $\varphi : S \times A \to \mathbb{R}$. The purpose of the task is encoded in this function, since the robot must select the sequence of actions which minimize the expected sum of the cost of actions.

The actions of the robot are determined by a policy, $\pi : S \times A \to [0,1]$, that maps for each state $s \in S$ the probability of selecting each of the actions $a \in A$. Typically, the robot will use the policy, π^*, that minimizes the expected sum of the cost of the actions during the execution of the task. Although the policy can be a function of time and the state history, in this work only stationary and Markovian policies are admissible.

IV. Hybrid Task Model

In the MDP framework, the robots execute the tasks using a standard "sense-think-act" loop. First, the current environment state is determined using the robot sensors. An action is selected, and performed by the robot, based on the policy it is using. The robot then waits for the environment to change state and the procedure is repeated until either a finite number of actions were performed or the goal state was reached. In either cases, the robot is not allowed to perform any more actions and further state changes are ignored.

The execution of an action, in this work, is identified with the activation of a navigation behavior. That is, the desired velocity of the robot is determined by selecting a navigation behavior, Definition 2, and solving (5) with the corresponding values for $\dot{q}_{ref}$ and $C(t)$. The viable set and the velocity reference of the behavior are fixed and cannot change online. For instance, the robots cannot activate a generic "go to" behavior, where the viable set and the velocity reference are determined when it is activated. Instead, these parameters are fixed at the start of the robot operation.

The robots can only activate navigation behaviors for which their pose is viable. The reason is that if the pose is not viable for a given behavior, the robot remains stopped. Since the viable sets of the behaviors are time-variant, the set of actions available at each will also be time-variant. This is modeled in this work, by introducing uncertainty in the distributions of the state transition functions.

Let the instants at which an action is selected be indexed by the variable n. At each instant, the matrix with the transition probabilities for any pair of states $s, s' \in S$, under action $a \in A$ is P_n^a. The values in this matrix are known only to lie inside confidence intervals. That is, each row of matrix P_n^a is a subset of the probability simplex in $\mathbb{R}^{|S|}$, where $|S|$ is the number of environment states. This uncertainty model for the transition function $p(.)$ of an MDP was introduced in [4], and it can accommodate for different methods in estimating the confidence intervals.

The execution of a task is then modeled as the sequential activation of navigation behaviors until a desired environment state is observed. Since the navigation behaviors are synthesized with DIs and the selection of the behaviors with MDPs, the tasks are modeled by combining these two frameworks.

Definition 3 (Task Hybrid Model): A model of a task is the tuple $(S, q, \mathscr{P}, B, \varphi)$ where S is the set of environment states, $q \in \mathbb{R}^n$ is the robot pose, $\mathscr{P}$ is the set of confidence intervals for the transition matrices, B is the set of navigation behaviors and φ is the state transition cost function.

The task model is hybrid as the state is composed by both continuous time variables, $q \in \mathbb{R}^n$, and discrete-time variables, $s \in S$ (see for instance [7] on hybrid systems). It is a well known problem in hybrid systems that not all sequences of discrete state transitions result in stable continuous state trajectories. In the case of the task hybrid model, some of the policies can result in unstable trajectories of the robot pose. Thus, in addition to minimizing the expected cost during the execution of the task, the policy used by the robot must also verify a stability constraint for the robot pose.

A. Stability of Optimal Policies

The stability analysis of hybrid systems in general, requires that suitable Lyapunov functions are found for each of the continuous dynamic modes of the system, [7]. In the case of the task hybrid model, a Lyapunov function must first be determined for each navigation behavior. Then, a given discrete state trajectory is stable if the Lyapunov functions of the respective sequence of navigation behaviors are upper-bounded by positive definite and monotonously decreasing function (see for instance the theorem on *Weak Stability* in [5]). The main difficulty with this approach is the computation of suitable Lyapunov functions for each behavior. For this reason, a different approach is pursued in this work. We begin with an intuitive definition of stability for the hybrid task models. It is then show that this definition is useful to show practical stability, [9], for the hybrid task model without explicitly determining the Lyapunov function.

Since the navigation behaviors are synthesized by constraining the robot pose to a compact and convex region, it is natural to specify a similar region for the execution of the task. For instance, in a patrol task, the robot must remain inside the, possibly non-convex but compact, region containing the desired patrol route and check-points. In search tasks, the robot is required to remain inside the region suspected of containing intruders.

Definition 4 (Task Stability): Let $T(t)$ be a set-valued map with compact and connected output values containing the start and goal poses. A trajectory of the robot pose, $q(t)$ with $t \in [t_0, t_1]$, is *task stable* if and only if $q(t) \in T(t)$ is verified $\forall t \in [t_0, t_1]$.

This notion of stability can be found in the concept of Foot Rotation Indicator (FRI) [10], which is used to classify the stability of the gaits of biped robots. The FRI is a function of the robot dynamics and it is only required that the output FRI

point does not leave the convex hull of the foot support area. A given gait is stable, in the sense that the robot does not fall, if the robot state variables and their derivatives verify a convex constraint.

In the tasks for which the set-valued map $T(t)$ is constant, the task stability can be identified with traditional notion of Lyapunov stability for dynamical systems.

Proposition 2 (Bounded Stability): Let π be a policy of a task hybrid model. Then there is a compact and constant set-valued map $T(t) = T$, where initially $q(0) \in T$, such that the policy is task stable and also Lyapunov stable.

Proof: Let $\overline{b}$ be the largest bounding ball of all output values of any of the viable sets $C(t)$. By construction, this bounding ball always exists and it has a finite radius because there is only a finite number of behaviors. From Proposition 1 and (5), any trajectory of the robot pose $q(t)$ is either viable, that is constrained to a bounded set, or otherwise constant since $V(.) = \emptyset$ and in this case $v(.) = 0$. Then for any $\varepsilon > 0$ such that $\|q(0)\| < \varepsilon$ there exists a finite $\delta(\varepsilon) > 0$ such that $\|q(t)\| < \delta(\varepsilon)$ for all $t > 0$, where $\delta(\varepsilon)$ is greater than the radius of $\overline{b}$. Thus, by definition all of the robot pose trajectories $q(t)$ are stable in the Lyapunov sense. The set T is the bounding ball with radius $\delta(\varepsilon)$. ■

Thus, if the robot pose is only required to be constrained to a constant region T, it can be assessed that all policies are stable in the Lyapunov sense. However this result is not very useful in practice since the region T can be arbitrarily large.

It is possible to study the Lyapunov stability of policies for time-varying $T(t)$ by imposing some conditions on this set-valued map and also the navigation behaviors available to the robots. Consider a collection of compact and convex sets, each associated with a discrete environment state

$$\mathscr{T} = \{\forall s \in S \,:\, T_s\} \tag{7}$$

The navigation behaviors available to the robots either constrain their pose to the set $T_s \in \mathscr{T}$ of an environment state or allow the robots to move between the respective sets of any pair of states. That is, for any pair of environment states $s, s' \in S$, the velocity reference of the behaviors are in the form

$$\dot{q}_{s,s'} = \frac{T^0_{s'} - T^0_s}{t_1 - t_0} \tag{8}$$

where $T^0_{s'}, T^0_s$ are the center points of the respective sets and $(t_1 - t_0)$ is the duration of the behavior. The viable set is the Lipschitzean map $C_{s,s'}(t)$ such that

$$\begin{cases} C_{s,s'}(t_0) = co\{T_s, T_{s'}\} \\ \\ C_{s,s'}(t_1) = \overline{b}_\varepsilon(T^0_{s'}) \end{cases} \tag{9}$$

where $co\{.\}$ is the convex hull and $\overline{b}$ is a ball of radius ε centered at point $T^0_{s'}$. Thus the viable set can be understood as a region initially containing both T_s and $T_{s'}$, but shrinking with time to a ball around the center point of $T_{s'}$.

Proposition 3 (Behavior Practical Stability): Any viable trajectory of the mobile robot generated by the navigation behaviors in the form $B = (C_{s,s'}(t), \dot{q}_{s,s'}, t_0, t_1)$ verifies the conditions for uniform practical stability, [9].

Proof: By construction, any viable trajectory of the navigation behavior $B = (C_{s,s'}(t), \dot{q}_{s,s'}, t_0, t_1)$ will reach the ball $\overline{b}_\varepsilon(T^0_{s'})$ at time t_1. Since the viable trajectories are also bounded, they verify the conditions for uniform practical stability [9]. That is, for any initial viable robot pose $q(t_0) \in C_{s,s'}(t_0)$, the corresponding viable trajectory is bounded and is attracted to the ball $\overline{b}_\varepsilon(T^0_{s'})$. ■

This result agrees with the fact that the viable sets of the navigation behaviors $B = (C_{s,s'}(t), \dot{q}_{s,s'}, t_0, t_1)$ were designed to verify the conditions for practical stability. In practice, the radius of the ball $\overline{b}_\varepsilon(T^0_{s'})$ is dependent on the robot kinematics and the initial pose. Furthermore, the existence of viable trajectories also depends on these two factors. However, it is always possible to adjust the temporal duration of the navigation behavior and the viable sets such that $V(q, F, C_{s,s'}) \neq \emptyset$ for any $q \in C_{s,s'}$. That is, the robot is only required to move at a slow enough pace. With these navigation behaviors, it is possible to show that the robot pose trajectory is uniformly practically stable.

Proposition 4 (Policy Practical Stability): Let $s_g \in S$ be the goal environment state, $s_0 \in S$ the initial state and π^* the optimal policy. The available navigation behaviors are in the form $B = (C_{s,s'}(t), \dot{q}_{s,s'}, t_0, t_1)$. If s_g can be reached from s_0, then the robot pose trajectory is uniformly practically stable.

Proof: From Proposition 2 all trajectories of the robot are bounded by a ball $\overline{b}$ containing the initial pose and the center of the set T_{s_g}. In order for the robot pose trajectory to be uniformly practically stable, it must be bounded and attracted to a ball of radius ε centered at $T^0_{s_g}$. By construction, all of the viable sets are bounded and employing similar arguments to those of Proposition 2, all viable trajectories are bounded. Consider now a sequence of discrete states, produced by policy π^*, beginning at s_0 and ending in s_g. Using Proposition 3, for any two consecutive discrete states s_i, s_{i+1}, the robot pose begins in T_{s_i} and ends in a ball centered at $T^0_{s_{i+1}}$. Thus, for the last discrete state in the sequence before the goal, the behavior $B = (C_{s_{g-1},s_g}(t), \dot{q}_{s_{g-1},s_g}, t_0, t_1)$ leads the robot pose trajectory $q(t)$ to a ball centered at $T^0_{s_g}$ ■

The use of the viable sets $C_{s',s}(t)$ allowed the stability analysis of the robot pose trajectories without using Lyapunov functions. This was possible because the sets where designed such that the viable trajectories verify the properties required for uniform practical stability. This was the motive for proposing the concept of task stability. In general, the objective of stability analysis is to find a Lyapunov function, and a suitable controller, such that the dynamical system verifies the stability conditions. With the notion of task stability, instead the trajectories of the dynamical system are constrained to those that verify the stability conditions.

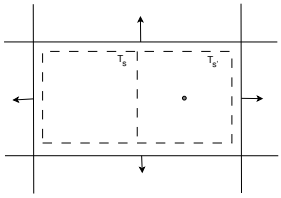

Fig. 2. Behaviors viable set

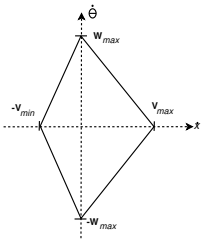

Fig. 3. Kinematics inclusion for a unicycle drive

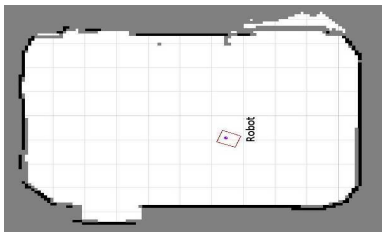

Fig. 4. Occupancy grid pap of the environment

V. Application Example

The proposed approach is applied to a search and pursue problem, where a mobile robot must search and intercept an intruder present inside a bounded region. The boundaries of the region are known, and it is subdivided in cells with a fixed size. The cell which contains the intruder and the probability distribution of obstacles over the cells are time-variant.

The formulation of the task hybrid model is divided in two parts. The first is the MDP where the environment states are the cells of the environment and the cells containing the robot and the intruder. It is assumed that the optimal policy has been computed and the reader is referred to [4] for details on the available algorithms. A similar problem is also described in [4], where a plane must avoid areas of turbulence. Their probability distribution is also time-variant and known only within some bounds.

The collection of sets $\mathscr{T}$ is composed by the cells of the map. Since the elements of $\mathscr{T}$ are only required to be compact and convex, any finite cover of the search region could have been used instead. The navigation behaviors are built using the cells of the map and their respective velocity references are obtained without effort. The main difficulty is computing the viable sets and their tangent derivatives, necessary for the evaluation of (5).

The convex sets are assumed to be described as the finite intersection of the hyperplanes represented in matricial form, $\{x \in \mathbb{R}^n : Ax \leq h\}$. The normal vector to each hyperplane is a row of matrix A and the respective element of vector h is the dot product of the normal vector with any point on the hyperplane, $a_i \cdot x - h_i = 0$.

This is a very convenient representation because the intersection of two convex sets is simply the concatenation of the respective matrices and vectors. The hyperplanes can also be moved along the direction of the normal vector, by scaling or adding the respective element of h with a positive value. Thus it is possible to generate the outputs of the set-valued map $C_{s,s'}(t)$ required by Proposition 3. An example is depicted in Figure 2, where the boundaries of the cells have dashed lines, the hyperplanes have solid lines and their normal vectors are pointing outwards. When the robot is in the same cell as the intruder, a similar viable set is used for the interception behavior.

The tangent derivative is the entire space $\mathbb{R}^n$ if the robot is not on the boundary of $C_{s,s'}(t)$. Otherwise it is expressed also as the intersection of hyperplanes in the form $\{x \in \mathbb{R}^n : A^t x \leq h^t\}$. The rows of A^t are the normal vectors of the hyperplanes in the viable set which the robot pose is touching, $a_i \cdot x - h_i = 0$. The components of vector h^t are $(-a_i \cdot \eta)\Delta t$, where a_i is the normal vector of the hyperplane, η the velocity of the viable set and Δt is the time between consecutive evaluations. Thus, the tangent derivative imposes lower bounds on the velocity of the robot when it is on the boundary of the viable set.

Finally, the differential inclusion that models the kinematics of the robots is also expressed as the intersection of hyperplanes. The polyhedron that models the kinematics inclusion for the P3-AT unicycle robot is represented in Figure 3, in the robot body frame. The maximal linear and angulat velocities are W_{max} and V_{max} respectively, while V_{min} is the maximum negative linear velocity. Since robots with unicycle drives cannot move sideways, the linear velocity along y is always constrained to be zero and is not represented in the figure. The polyhedron encodes the natural constraints of bounded control inputs on a unicycle robot. For instance, if both motors rotate in the same direction, at the maximal input voltage, then the linear velocity is maximal and the angular velocity is constrained to zero.

A. Navigation Behavior

A preliminary experiment was conducted on a rectangular environment to evaluate the proposed approach for a simple navigation behavior. A Pioneer P3-AT robot is placed inside a bounded rectangular space, with approximately 6 meters meters by 11 meters. The Player/Stage software framework, [11], was used to build the occupancy grid map with a resolution of $0.1m$, represented in Figure 4. The pose estimation was obtained using the particle filter method available from the same software framework. The purpose of the experiment is to evaluate the performance of a navigation behavior together with a standard approach for estimating the pose of the robot under almost ideal conditions, a static and known environment without obstacles.

The two viable sets for the navigation behavior are represented in Figure 5, with the bottom and left hyperplane

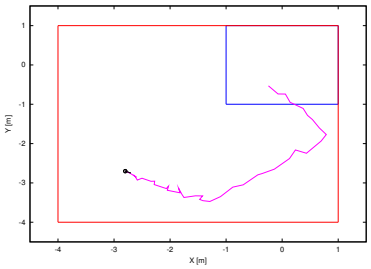

Fig. 5. Navigation behavior results

of the larger viable square moving towards the smaller one at the rate of $0.1m/s$, during 40 seconds. The robot initial pose is $(-2.8, -3.7, -0.2)$, represented in the figure by a circle at the left bottom of the image. The desired velocities, expressed in the robot body frame, are respectively $0.25m/s$ and $0.05rad/s$. Since the pose estimation is not accurate, the pose is assumed to be touching an hyperplane if it is at less than $0.2m$ away, which is twice the resolution of the grid map.

The robot moves, initially in the direction of the bottom hyperplane which is simultaneously moving upwards. When it is closer than $20cm$, the bottom hyperplane imposes a constraint on the motion of the robot, forcing it to move tangent to the hyperplane with a smaller linear velocity. Since it has also an angular velocity, eventually it is oriented towards the right hyperplane and the constraint of the bottom hyperplane is no longer active. The robot then moves freely until being close enough to the right hyperplane for the constraint to become active. Again it will slowdown and rotate until the motion constraint becomes inactive, then moving towards the interior of the smaller viable set.

Since a reasonable degree of accuracy is required in the pose estimative, a large amount of particles was used. This introduced a non-negligible time delay in the estimative that cause a sluggish reaction to when constraints became active. Another difficulty is that pose estimative exhibited abrupt changes in the values because of the estimation variance. This forced the use of a tolerance margin to the proximity with the hyperplanes and also limited their velocity.

This experiment suggests that the features of the pose estimation methods have a strong influence on the navigation behaviors. A possible solution is to employ multiple viable sets for the same navigation behavior and select the one where robot is further away from the border. Another possible approach is to use the robot body frame, since the pose is known and constant in this case. This is the future direction of research for this work.

VI. Conclusions

A behavioral approach for the synthesis of controllers for mobile robots was presented in this work. The tasks the robots must execute are modeled with two different frameworks, MDPs and DIs. In short, the synthesis of navigation is accomplished with DIs while MDPs are used to plan the sequence of behaviors such that the task goals are accomplished. The resulting models are hybrid, containing both continuous and discrete time variables. The concept of task stability was proposed, in which the robot pose is *task stable* if it is always constrained to a compact and connected time-variant region. The region can be understood as generalization of the viable sets used to synthesize navigation behaviors, without the convexity and Lipschitzean requirements .

The stability of the robot pose trajectories, during the execution of a task, was studied with the help of the task stability concept. It was shown that for any policy of the task model, the robot pose trajectories are bounded and stable in the Lyapunov sense. Also, by properly designing the viable sets of the navigation behaviors it is possible to show uniform practical stability of the robot pose trajectories. The strategy is to determine the collection of sets that is a finite cover of the time variant region where task takes place. The collection is then used to build the viable sets, and the velocity references, for each navigation behavior. The stability analysis was accomplished without the need to explicitly compute suitable Lyapunov functions for each task hybrid model. This was possible because the viable sets for the synthesis of navigation behaviors can be designed according to the desired stability requirements.

A fundamental requirement of the behavior synthesis method is the existence of viable trajectories for each navigation behavior. That is, for any behavior there must always exist at least one viable direction of motion. It can thus be understood as a controllability requirement for the design of the viable set.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.

[2] J. R. Hollerman and W. Schultz. Dopamine neurons report an error in the temporal prediction of reward during learning. *Nature neuroscience*, 1(4):304–309, August 1998.

[3] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[4] Arnab Nilim and Laurent El Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Operations Research*, 53(5):780–798, 2005.

[5] Georgi Smirnov. *Introduction to the Theory of Differential Inclusions (Graduate Studies in Mathematics)*. American Mathematical Society, 1st edition, 2001.

[6] N. Goncalves and J. Sequeira. Modeling robot behaviors with hybrid automata. In *8th Portuguese Conference on Automatic Control, 2008. Controlo 2008*, pages 706–711, July 2008.

[7] J. Lygeros. An overview of research areas in hybrid control. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 5600–5605, 2005.

[8] Steven M. Lavalle. *Planning Algorithms*. Cambridge University Press, May 2006.

[9] Antoine Chaillet and Antonio Loría. Uniform semiglobal practical asymptotic stability for non-autonomous cascaded systems and applications. *Automatica*, 44(2):337 – 347, 2008.

[10] Ambarish Goswami. Foot rotation indicator (fri) point: A new gait planning tool to evaluate postural stability of biped robots. In *IEEE International Conference on Robotics and Automation*, pages 47–52, 1999.

[11] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *ICAR 2003*, pages 317–323, Coimbra, Portugal, June 2003.