Sensor-Based Navigation Coordination for Mobile Robots

Xiaoming Hu, David Fuentes Alarcón and Tove Gustavi
Optimization and Systems Theory
Royal Institute of Technology
SE-100 44 Stockholm, Sweden
{hu,gustavi}@kth.se

***Abstract*— In this paper, the problem of navigation coordination is studied. New results are given for robustly integrating behavior based control systems. In the studied problem formulation, one behavior (path following) is a priori planned, the other (obstacle avoidance) is reactive. The methods are based on the virtual vehicle approach and a regularized automaton.**

Keywords: Mobile robots, navigation control, obstacle avoidance, sensor feedback

I. Introduction

For mobile robots the basic functionality is to navigate in an at least partially unknown environment without collision with any obstacle. A popular way in robotics for dealing with such problems is within the *behavior based* control architecture. (See for example [1], [9].) The main methodology is to identify different controllers for a complex control task with individual robot behaviors. The main advantage of this approach is that it makes the system modular, which both simplifies the design process as well as offers the possibility to add new behaviors to the system without causing any major increase in complexity, as pointed out in [2].

From a control point of view, there are two issues within this framework that need to be further studied. One is how to design each individual behavior (controller) in a robust way, based on available sensor data, and the other is how to fuse those behaviors.

The problem of navigation through an unknown environment we consider in this paper can be decomposed into two behaviors: path following and obstacle avoidance. Each of them has been a central problem in mobile systems and there is a vast literature on them (see, for example, [12] and the references therein). One key issue here is that given a reactive (sensor-based) obstacle-avoidance behavior, how should it be integrated with a path following behavior so that it incorporates the fact that the two are going to run concurrently? In [4] a hybrid control approach is used for integrating these behaviors, where a regularized automaton is proposed to reduce the chattering that might be caused by hard switches. In this paper, we further develop the results in [4] and [5] by using a virtual vehicle approach to coordinate the two behaviors. Furthermore, a sensor-based linear controller is designed for obstacle avoidance that can be easily integrated with the path following controller.

This paper is organized as follows. In Section 2, we give the problem formulation. In Sections 3 and 4, we discuss the design of individual behaviors. In section 5, the integration issue is discussed. In section VI we discuss the application of our results to a Khepera robot.

II. Problem formulation

The specific problem we consider is how to move a robot from an initial position to the goal in a planar environment via a planned path. This path following behavior should be interrupted when an obstacle is detected and be resumed when the obstacle is by-passed. Since the obstacle-avoidance behavior is a necessary safety measure and thus has higher priority than the path following one, it should be designed as a *reactive* behavior.The word reactive used here is to suggest that the behavior should be thought as a reflex. Thus real-time constraints are critical in designing such a controller.

In this paper we assume that we can control the robot's translational and rotational velocities. Naturally, for platforms that do not give direct control over the velocities, one needs to design an actuator control so that these velocity controls are realized. The implementation could be just a static mapping, as in the car case, or a dynamic regulator. In those cases, the velocity controllers can be viewed as higher-level controllers.

Based on our assumptions, we use the following unicycle model for the robot:

$$\begin{aligned} \dot{x} &= v\cos\phi \\ \dot{y} &= v\sin\phi \\ \dot{\phi} &= \omega, \end{aligned} \tag{1}$$

where $(x, y) \in \mathbb{R}^2$ is the position of the robot, ϕ is its heading, and v and ω are the controlled translational and rotational velocities respectively.

Our goal is to design a feedback controller (v, ω) that implements path following and obstacle avoidance, and is robust to disturbances and uncertainties.

III. Path following

Given an initial position and a final position, there are many ways to design a goal reaching behavior. In this paper we assume that the robot is required to follow a planned path. This might be the case, for example, if the robot is required to pass through a set of way points before reaching the goal. For various path planning algorithms, one can refer to, for example, [11], [12], [16], [6].

We now assume that the planned reference path is given by

$$\begin{array}{l} x_d = p(s) \\ y_d = q(s) \end{array} \quad 0 \le s \le s_f. \tag{2}$$

In this paper the path following control we use is based on the work in [5]. For the sake of completeness, we review the main ideas here and then in section V we will modify the controller in order to have it better integrated with the obstacle avoidance control.

In [5] a virtual vehicle approach is proposed where the idea is to let the motion of the reference point on the planned path be governed by a differential equation containing error feedback. It can be viewed as a combination of the conventional trajectory tracking, where the reference trajectory is parameterized in time, and a dynamic path following approach [15], where the criterion is to stay close to the geometric path, but not necessarily close to an *a priori* specified point at a given time. This approach makes the algorithm robust to measurement errors and external disturbances since, if both the tracking errors and disturbances are within certain bounds, the reference point moves along the reference trajectory while the robot follows it within a prespecified look-ahead distance. Otherwise, the reference point should slow down and "wait" for the robot.

The control objectives here are

$$\begin{array}{l} \limsup_{t\to\infty} \rho(t) \le \epsilon_\rho \\ \limsup_{t\to\infty} |\phi(t) - \phi_d(t)| \le \epsilon_\phi, \end{array} \tag{3}$$

where ϵ_ρ and ϵ_ϕ are positive numbers that can be made arbitrarily small, $\rho(t) = \sqrt{(x_d - x)^2 + (y_d - y)^2}$, where ($x$,$y$) is the actual position of the robot, and ϕ and ϕ_d are actual and desired robot orientations.

From (2) we directly get that $\dot{x}_d = p'(s)\dot{s}, \dot{y}_d = q'(s)\dot{s}$, which implies that if the robot would track the path perfectly, we would have

$$\dot{s} = \frac{p'(s)}{p'^2(s) + q'^2(s)}\dot{x} + \frac{q'(s)}{p'^2(s) + q'^2(s)}\dot{y}, \tag{4}$$

since this corresponds to $\dot{x} = \dot{x}_d$ and $\dot{y} = \dot{y}_d$. On the other hand, (4) does not contain any position error feedback, which is important for the robustness. Therefore we propose our dynamics for the reference point as follows:

$$\dot{s} = \frac{ce^{-\alpha\rho}v_0}{\sqrt{p'^2(s) + q'^2(s)}}, \tag{5}$$

where v_0 is the desired speed at which one wants the vehicle to track the path, and α and c are appropriate, positive numbers.

Remark: If the reference path is only given as a collection of dense way points $\{(x_d(s), y_d(s)) : s = 1, 2, \ldots, N\}$, then (5) can be modified as

$$s_{k+1} = s_k + \sigma(\rho_k), \tag{6}$$

where

$$\sigma = \begin{cases} 0 & \text{if } \rho_k > \epsilon_\rho \\ 1 & \text{if } \rho_k \le \epsilon_\rho \end{cases}$$

We now let our control algorithm be as follows:

$$\begin{array}{l} v = \gamma\rho\cos(e_\phi) \\ \omega = ke_\phi + \dot{\phi}_d, \quad k > 0, \end{array} \tag{7}$$

where both γ and k are positive, $e_\phi = \phi_d - \phi$, and $\phi_d = \arctan2(y_d - y, x_d - x)$.

Remark: *e_ϕ is not defined at $\rho = 0$ since ϕ_d is not defined. In implementation, one can replace ϕ_d by*

$$\tilde{\phi}_d = \begin{cases} \phi_d & \text{if } \rho > \epsilon \\ \frac{\phi_d(-2\rho^3 + 3\epsilon\rho^2) + \theta_r(-2(\epsilon-\rho)^3 + 3\epsilon(\epsilon-\rho)^2)}{\epsilon^3} & \text{if } \rho \le \epsilon, \end{cases} \tag{8}$$

where ϵ is a small positive number and θ_r is the orientation angle of the tangent to the reference curve at s. It is easy to see that $\tilde{\phi}_d$ is well defined at $\rho = 0$ since $\lim_{\rho\to 0} \phi_d(-2\rho^3 + 3\epsilon\rho^2) = 0$.

An intuitive interpretation for this control algorithm is that the robot is steered toward the reference point on the desired path with a speed proportional to the tracking error and whenever it reaches the steady tracking error (the desired looking-ahead distance) it will track the path with an almost constant speed. It is easy to see that even if a path is planned beyond the dynamic or kinematic constraints of the system, this control algorithm can still be used to track the path the best it can.

Lemma 3.1: For a robotic system (1), when the path following control (5), (7) and (8) are applied, the tracking error is bounded globally. Furthermore, the tracking error can be made arbitrarily small as $t \to \infty$, by tuning the parameters in the control. In particular, if we let $c = e^{\frac{\alpha v_0}{\gamma}}$, increasing γ will reduce the steady state tracking error, and v_0 will approximately be the speed in the steady state.

Remark: Naturally, when we decrease the steady state tracking error, the on-line computation will go up.

IV. Obstacle avoidance

Sensor-based obstacle avoidance has been a very important problem in mobile robotics. There are basically two groups of techniques for avoiding obstacles. One is reactive (online) and the other is re-planning (mostly offline) [8], [13], [12]. When designing a reactive controller, one important factor one should take into account is the constraints on communication and computation bandwidth.

In this section we propose a simple reactive controller that is directly based on sensor feedback, and can be easily integrated with the path following controller. Used alone, the proposed control will prevent the robot from running into an obstacle, but it will not make any attempt to steer the robot in accordance to a pre-defined path. However, in combination with the path-following control it shows a robust path following behavior, as will be discussed in Section 5.

Adding the controls from the obstacle avoidance behavior and the path following behavior will, with properly chosen parameters, result in a trajectory approximately tangential to the surface of the obstacle.

We assume that the sensors available for obstacle detection are an array of infra-red sensors that are placed symmetrically. The readings from each pair of such sensors would give a linear approximation of the obstacle contour.

Once such a linear approximation of obstacle contour is detected and is decided to be close to the robot, the objective of the obstacle avoidance control is to steer the robot away from the obstacle. The proposed control function is implemented as a linear combination of the sensor readings with parameters chosen in order to steer the robot away from the obstacle by making a smooth turn. Once the robot has turned away from the object it will continue to move in a direction normal to the contour.

To illustrate the design of the obstacle avoidance control we assume the total number of sensors is eight.

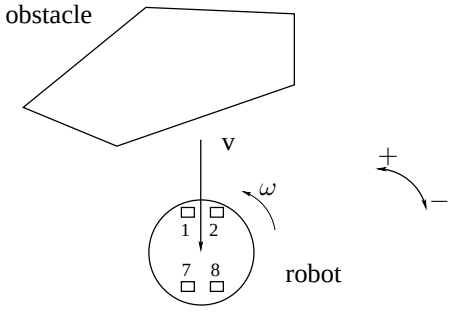


Fig. 1. Robot with two head-on sensors

for the sensor pair $(1, 2)$ (Figure 1) the reactive control is

$$v = -k_1(y_1 + y_2) \quad (9)$$
$$\omega = p_1(-y_1 + y_2) \quad (10)$$

where k and p are positive parameters, and y_1 and y_2 are normalized readings from sensor 1 and 2. Naturally one has to tune the parameters for a given platform.

Control for pair $(7, 8)$ can be designed similarly

$$v = -k_7(y_7 + y_8) \quad (11)$$
$$\omega = p_7(-y_7 + y_8) \quad (12)$$

Remark: When $y_1 = y_2$, i.e. when the robot is facing an obstacle straight on, a situation may occur where the obstacle avoidance and the path following controls cancel out. However, it happens rarely since this orientation is unstable for obstacle avoidance control (see Proposition 4.1), and the solution is discussed in Section 5.

For pair $(3, 5)$ (Figure 2) the reactive control is

$$v = -k_3y_3 + k_5y_5 \quad (13)$$
$$\omega = -p_3y_3 - p_5y_5 \quad (14)$$

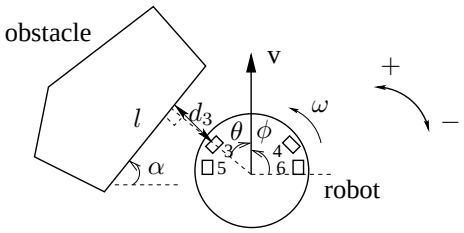


Fig. 2. Robot with two pairs of side sensors.

Here we should have $k_3 < k_5$ and $p_3 > p_5$. To motivate this choice of parameters, let us consider the case in Figure 2 where the robot is approaching an obstacle. If the sensor readings y_3 and y_5 are equal the desired behavior of the robot should be to turn to the right while moving forward with a positive velocity, v. This implies that $k_5 > k_3$. $p_3 > p_5$ follows from the fact that a signal from sensor 5 indicates only that the robot is passing beside an object and not necessarily that it has to turn away to avoid the obstacle.

Similarly for pair $(4, 6)$

$$v = -k_4y_4 + k_6y_6 \quad (15)$$
$$\omega = p_4y_4 + p_6y_6 \quad (16)$$

Thus if we use $y = (y_1, \ldots, y_m)^T$ to indicate the normalized sensor readings, we will end up with a reactive controller

$$v = Ky \quad (17)$$
$$\omega = Py \quad (18)$$

Remark: Here we assume that the sensor reading is linear to the distance. In reality, an infra-red sensor like the ones used in Khepera may have a very nonlinear characteristics:

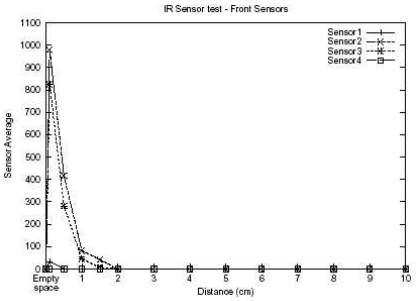


Fig. 3. Khepera infra-red sensor outputs

Then, one may need to use some $f(y)$ instead of y in the controller for higher accuracy. The readings might also be quite noisy. In this case one might have to use an EKF first.

Proposition 4.1: Suppose the linear contour l (see Figure 2) is sufficiently long, the sensors are identical and the output y is monotonically decreasing with the reflecting

distance d, and $y(d) > 0$ for all $d > 0$. Then, with the reactive control (9) to (16)

$$\dot{\phi} = \omega = Py$$

has two equilibria: $\phi_1 = \frac{\pi}{2} + \alpha$ $(\theta = 0)$ and $\phi_2 = -\frac{\pi}{2} + \alpha$ $(\theta = \pi)$. Furthermore, ϕ_1 is unstable and ϕ_2 is asymptotically stable.

Remark: The proof is fairly straight forward and omitted here. This result implies that if the environment consists of only convex polyhedrals, then the robot would not get stuck in any local minima since it is unstable and there is always some noise.

V. Navigation coordination

When adding an obstacle-avoidance behavior to the path following behavior, the overall system may need to be coordinated by a hybrid automaton. In fact, we may get two different possible hybrid automata (see for example [14]). This depends on whether the two behaviors are active simultaneously or not.

If one treats concurrently active behaviors as sub-controllers for the overall system, different controllers affect the system simultaneously, resulting in a smooth overall performance, as shown in [10]. However, with this method the analysis of the system is becoming significantly more difficult as new behaviors are added.

The other possible solution to the coordination problem, corresponding to hard switches between the different behaviors, is more scalable, but has the disadvantage that it increases the risk of introducing chattering into the system and the overall transient response may not be satisfactory.

In this paper we follow the method proposed in [4], where the idea is to use a regularized automaton, which improves the transient performance by adding some intermediate nodes to the automaton.

We use the problem we study here to illustrate the idea. When an obstacle is closer to the robot than a priori given distance d_{OA} (for example, the range of the infra-red sensors), the obstacle-avoidance behavior becomes active. If we would use hard switches between the two behaviors, under some scenarios this may lead to a chattering situation. On the other hand, if we consider the overall system with these two behaviors as a discontinuous differential system, there exists a sliding surface under some assumptions. Since the repulsive potential field from the obstacle-avoidance behavior is orthogonal to the surface on which the behavior becomes active, the vector field on the sliding surface could be just

$$f_S = \alpha_s f_{OA} + (1 - \alpha_s) f_{PF}, \tag{19}$$

where $\alpha_s \in [0, 1]$ is so that f_S is orthogonal to f_{OA}. Then adding this as an intermediate node would give better transient response and avoid the so-called "Zeno phenomena" [7]. This discussion can be summarized as follows:

Proposition 5.1: Suppose the obstacle in consideration is either a circle or a line segment, and the norm of each of the vector fields corresponding to the path following and obstacle avoidance behaviors is proportional to the respective distance, then at each point of the sliding surface, the resulting sliding solution is orthogonal to the obstacle avoidance vector field.

Proof: The conclusion follows from the fact that at each point of the sliding surface, the projection of the path following vector field to the normal direction of the obstacle has to cancel the obstacle avoidance vector field.

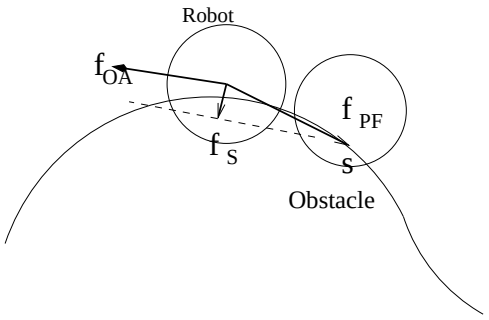


Fig. 4. The fusion of Path following and obstacle-avoidance.

In practice, what we do is to switch the system into the following intermediate node once an obstacle is detected:

$$f_s = \alpha f_{OA} + (1 - \alpha) f_{PF}, \tag{20}$$

for some $\alpha \in [0, 1]$. Different choice of α will result in different distance to the obstacle from the resulting sliding surface.

Since we use the virtual vehicle approach for goal reaching, a remaining issue here is when the obstacle-avoidance is active, how one should update the virtual vehicle. After all, the updating of the virtual vehicle will eventually decide if the robot will get stuck on the sliding surface or move on after it has passed the obstacle. We propose the following modification to (5),

$$\dot{s} = \begin{cases} \frac{ce^{-\alpha\rho}v_0}{\sqrt{p'^2(s)+q'^2(s)}} & \text{if } \rho > d_{OA} \\ \frac{v\cos(\phi-\theta_r)}{\sqrt{p'^2(s)+q'^2(s)}(1-\kappa(s)\rho)} & \text{if } \rho \leq d_{OA} \end{cases} \tag{21}$$

where $\kappa(s)$ is the curvature of the reference path at s and the rest of the variables are defined in Figure V. What the second component suggests is that once the obstacle-avoidance is active, s should switch to be the orthogonal projection of the robot position onto the path [3].

Remark: In order to keep the second term in (21) bounded, the curvature has to be sufficiently small everywhere. In practice, one can use a saturation function to make the term always bounded.

Arbitration

Let us denote the path following control action as u_{PF} and the obstacle-avoidance action as u_{OA}. If properly scaled, the

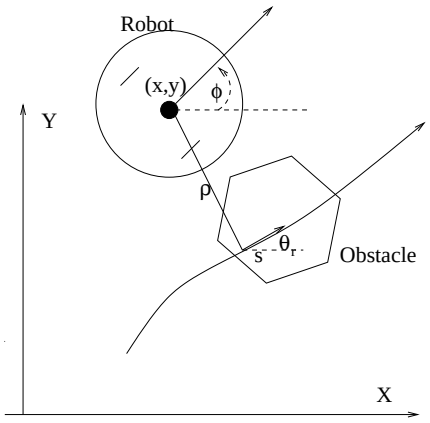


Fig. 5. Update of s when obstacle avoidance is active

regularized automaton would lead the overall control action to

$$u = u_{PF} + u_{OA}.$$

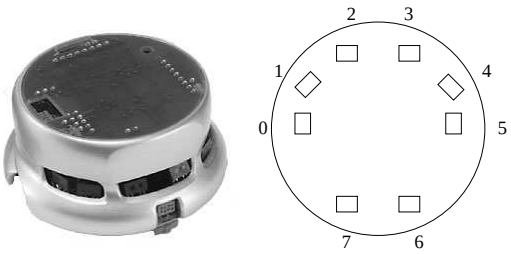
In order to avoid singular cases and possible local minima since the environment does not always consist of only convex objects, we propose the following arbitration control

$$\omega = \delta, \quad \text{if } u_{PF} = -u_{OA} \neq 0.$$

VI. Experiments

In this section we use some experimental results on a Khepera robot to illustrate our methods.

Khepera is a small mobile robot (55 mm in diameter), which was designed as a research and teaching tool. It allows testing algorithms, behaviors and control strategies theoretically developed for trajectory planning, obstacle avoidance and hypotheses on behavior processing, among others.



(a) (b)

Fig. 6. Khepera and its infra-red sensors

Each wheel of the robot is driven by a DC motor coupled with it. The two motors are independent, which leads to decoupled translation and rotation. Thus (1) is a good model for it.

The robot has two encoders (one in each wheel) for sensing the position of each wheel and then, the distance covered by the wheels. The encoder readings are the only data we used in our experiments for position feedback. Naturally the accuracy can be improved by incorporating other sensors.

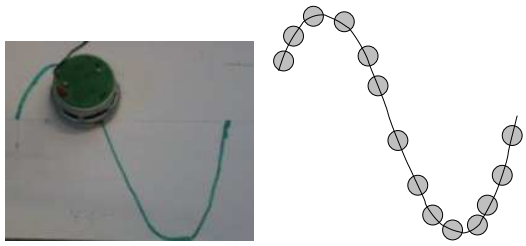
The sensors we used for detecting obstacles are the infra-red proximity sensors on Khepera. Eight such sensors are placed around the robot and are positioned as four in front, one in each side and two in back.

In our experiments, the controllers proposed in the previous sections are implemented. For obstacle avoidance, the following controller is used:

$$\begin{pmatrix} \omega \\ v \end{pmatrix} = 0.1 \begin{pmatrix} -0.2 & -1 & -1 & 1 & 1 & 0.2 & 0 & 0 \\ 0.5 & -0.2 & -0.3 & -0.3 & -0.2 & 0.5 & 0.1 & 0.1 \end{pmatrix} \begin{pmatrix} y_0 \\ \vdots \\ y_7 \end{pmatrix},$$

where $y_i's$ are the infra-red sensor outputs. The coefficients are obtained after some trial and error. When the path is generated by a bitmap file, the virtual vehicle (6) is used instead.

In Figure 7 the following of a path by the Khepera is shown, where the path is generated by a cubic spline.



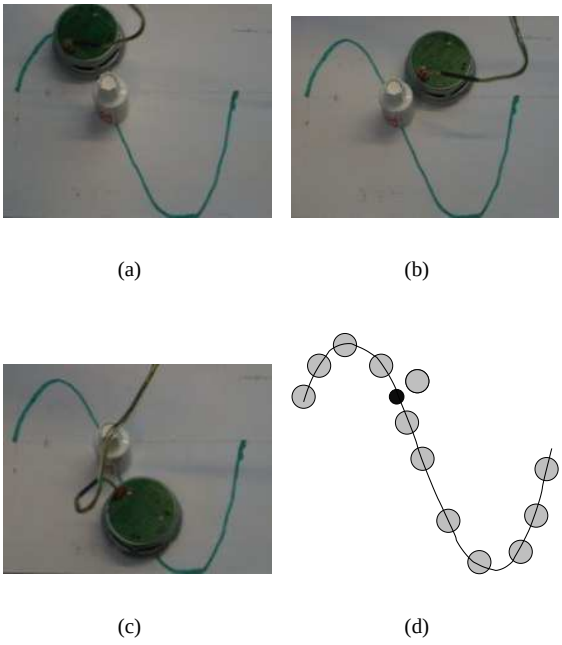
(a) (b)

Fig. 7. Path following

In Figure 8 the same path following and obstacle avoidance is shown. We should point out that the obstacle was placed on the path randomly after the robot had started to move on the path.

VII. Conclusions

In this paper a method for integrating behavior based control systems, based on a virtual vehicle approach, is presented. In particular we discuss the algorithm for updating the virtual vehicle when the obstacle avoidance control is active, and a modification of the results in [4] is proposed. Also, a new reactive controller for obstacle avoidance is suggested. The strength of this control is that it is simple to implement, robust and does not require a great deal of computational resources. The results of this work has been verified experimentally on a Khepera robot and the results of the experiments are presented in the paper. Furthermore,



(a) (b)

(c) (d)

Fig. 8. Path following and obstacle avoidance

this controller for a reactive behavior can be easily integrated with the controller for the planned path following behavior.

Acknowledgment This work is in part sponsored by the Swedish Research Council and by the European IST project RECSYS.

VIII. REFERENCES

[1] R.C. Arkin. *Behavior-Based Robotics*. The MIT Press, Cambridge, Massachusetts, 1998.

[2] R. Brooks: A Robust Layered Control System for a Mobile Robot, *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14-23, 1986.

[3] C. Canudas de Wit, Bruno Siciliano and G. Bastin, Theory of Robot Control. *Springer Verlag, 1996*.

[4] M. Egerstedt and X. Hu, A Hybrid Control Approach to Action Coordination for Mobile Robots, *Automatica vol. 38* no. 1, 2002.

[5] M. Egerstedt, X. Hu and A. Stotsky, Control of Mobile Platforms Using a Virtual Vehicle Approach, *IEEE Trans. Aut. Control vol. 46* no. 11, 2001.

[6] M. Egerstedt and C. Martin, Trajectory planning in the infinity norm for linear control systems, *Internat. J. Control 72*, no. 13, 1999.

[7] K. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. Regularization of Zeno Hybrid Automata. *Systems and Control Letters*, Vol. 38, pp. 141-150, 1999.

[8] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. robot Res*, 90-98, 1986.

[9] D. Kortenkamp, R.P. Bonasso, and R. Murphy, Eds. *Artificial Intelligence and Mobile Robots*. The MIT Press, Cambridge, Massachusetts, 1998.

[10] J. Košecká: *A Framework for Modeling and Verifying Visually Guided Agents: Design, Analysis and Experiments*. Dissertation, Grasp Lab, March 1996.

[11] B. Krogh and C. Thorpe. Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles, Proceedings of the *1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 1664-1669, 1986.

[12] J.C. Latombe. *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[13] T. Lozano-Perez, A simple motion-planning algorithm for general robot manipulators, *IEEE Trans. Robot. Automat.*, June, 1987.

[14] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for Reachability Specifications for Hybrid Systems. *Automatica*, Vol. 35, No. 3, March 1999.

[15] N. Sarkar, X. Yun, and V. Kumar. Dynamic Path Following: A New Control Algorithm for Mobile Robots. *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, Texas, Dec. 1993.

[16] A. Stenz. Optimal and Efficient Path Planning for Partially Known Environments, Proceedings of the *1994 IEEE International Conference on Robotics and Automation*, 1994.

[17] C. Tomlin, G. Papas, J. Košecká, J. Lygeros, and S.S. Sastry. Advanced Air Traffic Automation: A Case Study in Distributed Decentralized Control, *Control Problems in Robotics*, Lecture Notes in Control and Information Sciences 230, Springer-Verlag, London, 1998.

[18] P. Ögren, M. Egerstedt, L. Petersson and X. Hu, Reactive Mobile Manipulation Using Dynamic Trajectory Tracking: Design and implementation, in the proc. of CDC 2000.

[19] J. Tan and N. Xi, Integrated sensing and control of mobile manipulators, IEEE International Conference on Intelligent Robots and Systems, v 2, 2001, p 865-870.

[20] K. Arras, J. Persson, N. Tomatis, and R. Siegwart, Real-time obstacle avoidance for polygonal robots with a reduced dynamic window Proceedings - IEEE International Conference on Robotics and Automation, v 3, 2002, p 3050-3055.

[21] H. Noborio, I. Yamamoto and T. Komaki, Sensor-based path-planning algorithms for a nonholonomic mobile robot, IEEE International Conference on Intelligent Robots and Systems, v 2, 1997, p 510-517.