A HYBRID CONTROL APPROACH TO ACTION COORDINATION FOR MOBILE ROBOTS [1]

M. Egerstedt, X. Hu and A. Stotsky

{magnuse, hu, stotsky}@math.kth.se
Div. of Optimization and Systems Theory
Royal Institute of Technology
SE - 100 44 Stockholm, Sweden

Abstract: In this paper, the problem of how to integrate different robotic actions, within a behavior-based framework, is investigated. The case we study is the obstacle negotiation problem, and our approach is based on planned reference trajectories. These trajectories are chosen in such a way that the combination of a robust path-following behavior, designed in a model independent way, and a reactive obstacle-avoidance behavior would move the robot around an obstacle at a predefined safety distance. This is done while keeping the value of a cost functional low, defined for evaluating the performance of the robot. We exemplify and evaluate this approach on a Nomad 200 mobile robot platform.

Keywords: Mobile robots, Hybrid systems, Motion control, Path planning

1. INTRODUCTION

In a *behavior-based* robotic system, the idea is that different behaviors are identified as action responses to sensory inputs. A behavior could, for instance, be *obstacle avoidance* in which sonar information about a close obstacle results in a movement away from that obstacle. This way of identifying different distinct robot behaviors has the advantage of making the system modular. New behaviors can be added without causing any major complexity increase in the system at the same time as the control problem can be divided into subparts, which simplifies the design process.

The problem that will be investigated in this paper concerns *action coordination*. How should different behaviors be combined so that the resulting robot response is satisfactory from both the safety and performance perspective? One way of addressing this is to use an arbitration mechanism, where behaviors with higher priorities simply overrule other behaviors. For instance, when moving a robot toward a given goal point, not bumping into things is probably considered to be more important than actually reaching the goal point at any cost.

In this paper, however, we choose to work with a slightly different strategy, where many behaviors could be active simultaneously since we believe that there is much to gain in terms of performance if the robot both tries to avoid obstacles and move toward the goal point at the same time. The question then becomes that of determining the weights on each individual behavior in order to make the total robot system behave in a satisfactory way. (Large *et al.*, 1997; Shöner and Dose, 1992)

Naturally, this obstacle negotiation problem has been studied extensively (Arkin, 1998; Brooks, 1986) but the main advantage with our proposed hybrid control approach is that it explicitly deals with questions concerning safety and optimality. Instead of the traditional approaches, using poten-

[1] This work was sponsored in part by the Swedish Foundation for Strategic Research through its Centre for Autonomous Systems at KTH.

tial field type methods, we are able to produce a robot behavior that satisfies the safety constraints at the same time as the solution is close to optimal with respect to a given performance evaluation functional.

In Section 2, we illustrate how a *hybrid system* can be used for solving this problem in a systematic way. A hybrid control system is a system whose continuously controlled states are affected by the occurrence of discrete events and in our case, these events would correspond to the detection of an obstacle, resulting in a need for a new control strategy in order to make the robot avoid this obstacle.

In Section 3, a reference output trajectory is produced, leading the robot around the obstacle. Based on a stable and model independent path following behavior, derived in Section 4, we proceed, in Section 5, by proposing a solution to the weighted action problem. We then investigate how to combine this behavior with an obstacle avoidance behavior in such a way that the resulting control moves the robot around the obstacle in a safe manner, at a predefined safety distance, while keeping the curvature of the path small.

We conclude by implementing our ideas on a Nomad 200 platform in order to show that our approach not only works in theory but also works well on some real systems.

2. THE OBSTACLE NEGOTIATION PROBLEM

The specific problem that will be investigated in this article is how to come up with a control that moves a robot between two points. This *point-to-point motion* should be done so that the detection of an obstacle could be dealt with in such a way that the robot will not be closer to the obstacle than a desired safety distance, C_s.

We assume that both the longitudinal velocity, v, and the heading of the robot, ϕ, can be controlled directly. We thus get that

$$\dot{\phi} = \omega. \tag{1}$$

2.1 *Obstacle Avoidance*

The behavior that we will focus on initially is a so called *reactive* obstacle avoidance behavior. The word reactive, a commonly used one in the robotics community, is used here since it is a behavior that can be thought of as a reflex. When the robot moves too close to an obstacle, it is forced to change the motion to avoid hitting the obstacle. This is a reasonable safety strategy, since the robot may move around in an unstructured world, where the occurrence of unpredicted, or unmodeled obstacles is very likely.

If the sonars on the robot, with center of gravity at (x, y) and heading ϕ, detect a point-obstacle at (x_{ob}, y_{ob}), the reactive control response will be,

$$\omega = C_{OA} W_{OA}(d)(\tilde{\phi} - \phi), \tag{2}$$

where

$$d = \sqrt{(x - x_{ob})^2 + (y - y_{ob})^2},$$
$$W_{OA}(d) = \begin{cases} \dfrac{(d_{OA} - d)}{(d + \epsilon)^3} & \text{if } d < d_{OA} \\ 0 & \text{if } d \geq d_{OA} \end{cases}$$

and

$$\tilde{\phi} = \pi + \text{atan2}(y_{ob} - y, x_{ob} - x). \tag{3}$$

Here, d_{OA} is the fixed distance from the obstacle where the behavior becomes active. The constant C_{OA} is a weight that will be determined further on in the article in order to make the overall system behave in a satisfying way.

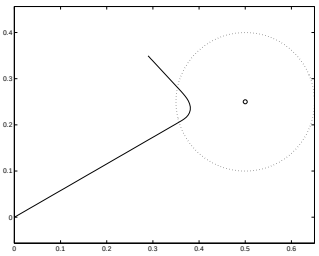


Fig. 1. The robot movement is shown (solid), when driving with a constant velocity with only the obstacle-avoidance behavior determining the robot's response. The dotted circle around the point-obstacle is the border where the behavior becomes active.

Since a real, extended obstacle cannot be considered to be a point, in the actual implementation of the avoidance behavior we calculate the desired heading angle $\tilde{\phi}$ as the orientation of the sum of the weighted vectors that each individual sonar reading contribute with. For the Nomad 200 this corresponds to taking the sum over 16 elements since the Nomad is equipped with 16 ultrasonic sensors.

A conventional control strategy for incorporating this obstacle-avoidance behavior would be

$$\dot{\phi} = \begin{cases} u(t), & \text{if no obstacle is within range} \\ C_{OA} W_{OA}(d)(\tilde{\phi} - \phi), & \text{if an obstacle is within range} \end{cases}$$

for some predefined control $u(t)$.

This gives a switching hybrid control system. However, the drawback of this strategy is that

it makes trajectory tracking inefficient and thus affects the performance of the robot in terms of goal achievement.

Instead, we propose the following control strategy

$$\begin{aligned}\dot{\phi} = & \; s(d)C_{OA}W_{OA}(d)(\tilde{\phi} - \phi) \\ & + \; (1 - s(d))u(t),\end{aligned} \tag{4}$$

where $s(d)$ is a smooth step, defined as

$$\begin{aligned}s(d) = \frac{1}{2}\Bigg(&\tanh(k(\frac{d}{d_{OA}} + 1)) \\ & - \; \tanh(k(\frac{d}{d_{OA}} - 1))\Bigg),\end{aligned}$$

where $k > 0$. The difference here is that we do not use a real step function since we want to fuse the controllers together so that more than one behavior can be active at the same time. The reason why we want to leave the obstacle avoidance term in (4) is due to the fact that we want to maintain the modular, behavior-based structure of the system.

2.2 *A Hybrid Control Approach*

Our method of combining a continuous control system with the occurrence of discrete events, such as detection of obstacles, is inspired by the work of Tomlin *et al.* (1998). The main idea is to view the hybrid control problem as an optimal control problem.

If we let our admissible controls be $u \in \mathcal{U}$, and define a safety functional

$$\mathcal{J}_s(u) = \min_{t \geq 0} \left\{ (x(t) - x_{ob})^2 + (y(t) - y_{ob})^2 \right\}, \tag{5}$$

then the set of controls, $\mathcal{U}_s(C)$, that make the robot move at least a distance C away from the obstacle, can be defined as

$$\mathcal{U}_s(C) = \{u \in \mathcal{U} \; : \; \mathcal{J}_s(u) \geq C\}. \tag{6}$$

It should be mentioned that both $\mathcal{J}_s$ and $\mathcal{U}_s$ depend on the robot's initial position but for the sake of notational simplicity, we leave that out from the definitions.

The next step is to define another cost functional that penalizes high curvature of the chosen path. This is a reasonable performance criterion since it penalizes paths that make the robot move in sudden, abrupt ways, resulting in uncomfortable robot motions.

The control candidates for minimizing this functional will be chosen from the set of safe controls, $\mathcal{U}_s(C_s)$, where C_s is our preferred safety margin.

Unfortunately it turns out that this is a very hard problem to solve numerically (not to mention analytically), but the underlying approach could suggest a way for producing a solution to the obstacle negotiation problem that is both safe, computationally feasible, and makes the system behave in a satisfactory way with respect to keeping the curvature of the produced path small.

Our idea that will be pursued in the following sections, is that instead of focusing on the hard optimal control problem we should concentrate on just producing geometrical trajectories that lead around the obstacle. This way we would not have to deal with the actual kinematics of the robot in the optimization formulation. Instead we add the kinematics when we track the produced path. This would mean that we cannot be sure that we actually find the optimal controller, but rather that we find one that is reasonably close to the optimal one.

3. PATH PLANNING

One first observation is that for a path produced by a scalar function $y_d = f(x_d)$, the curvature is given by

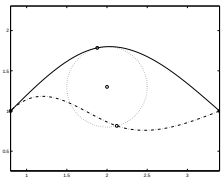
$$\kappa(x_d) = \frac{f''(x_d)}{(1 + f'(x_d)^2)^{3/2}}, \tag{7}$$

where the subscript d stands for the desired robot position.

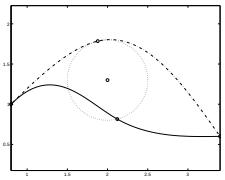
If we minimize $f''(x_d)^2$ instead of $\kappa(x_d)^2$, we would make $\kappa(x_d)^2$ small automatically, which was a desired feature in our planned paths, as seen in the previous section.

If an obstacle is detected, we identify two points on the boundary of the safety region around that obstacle and calculate both the trajectory that goes above and the one that goes below the obstacle. Since we are minimizing the L_2-norm of the second derivative, the resulting curve will be a spline. This is a fortunate fact since it means that we will not be forced to relay on extensive world information or to do any heavy computations on-line, which tends to be the case when using more sophisticated planning algorithms. (Krogh and Thorpe, 1986; Latombe, 1991; Stenz, 1994)

It should be pointed out that these trajectories actually do not stay away from the safety region around the obstacle at all times since we only interpolate through one distinct point. The reason for choosing such a solution will become clear further on when we fuse the path-following with the obstacle-avoidance behavior. To put it simply here, it is because the combination of these two behaviors will force the robot away from the safety region even if the planned path leads through it.



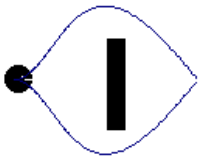
(a) A case when it is optimal to go above the obstacle.



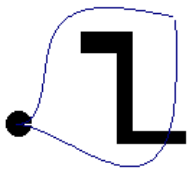
(b) A case when it is optimal to go below the obstacle.

Fig. 2. In these two figures, the proposed path is shown, and since we have two choices (over or under the obstacle) we simply choose the best of these two paths (solid). The dash-dotted line shows the path that we did not choose.

When using this approach on real, extended obstacles, instead of interpolating around point obstacles, we actually identify the boundary of the obstacles by for example sonars, and then interpolate through these points, as seen in Figure 3.



(a) The two paths planned around the obstacle.



(b) Replanning is needed since the whole obstacle cannot be seen by the sonars.

Fig. 3. Here, the planner is implemented on the Nomad 200-simulator, the `Nserver`, and in the second figure, replanning is needed if the lower path is found to be optimal. This is, of course, due to the fact that new information about the extension of the obstacle affects the way the robot should move.

4. TRACKING

We now have a method for producing paths that lead around the obstacle, and make the curvature small, and hence our next task is to find a good tracking algorithm so that the robot follows the proposed path robustly.

We devote this section to the development of such a control algorithm. We let the general reference path, parameterized by s, be given by

$$\begin{aligned} x_d &= p(s) \\ y_d &= q(s), \end{aligned} \quad 0 \le s \le s_0 \tag{8}$$

and we assume that the desired path is smooth, i.e. $p'^2(s) + q'^2(s) \neq 0, \quad \forall s \in [0, s_0]$.

In other words, what we require is that

$$\lim_{t \to \infty} \rho(t) = \epsilon \tag{9}$$

where ϵ is a positive number that can be arbitrarily small, and

$$\rho(t) = \sqrt{(x_d - x)^2 + (y_d - y)^2} \tag{10}$$

where (x,y) is the actual position of the reference point on the robot.

In this paper we propose an approach in which a reference point on the reference path is chosen and then a simple control algorithm is used to steer the robot toward that point. What is different in our approach from other similar methods is that the evolution of the reference point is governed by a differential equation which contains the position error. One of the advantages of our approach is that it is quite robust with respect to measurement errors and external disturbances.

We should point out that in Canudas de Wit *et al.* (1996) and Jiang and Nijmeier (1997), tracking controls are designed so that the tracking errors tend to zero globally. However, their controls are more complex and very much model dependent, while our control is quite intuitive and model independent.

From (8) we directly get that

$$\begin{aligned} \dot{x}_d &= p'(s)\dot{s} \\ \dot{y}_d &= q'(s)\dot{s}, \end{aligned} \tag{11}$$

which implies that if the car (x, y) would track the path perfectly we would have

$$\dot{s} = \frac{p'(s)}{p'^2(s) + q'^2(s)}\dot{x} + \frac{q'(s)}{p'^2(s) + q'^2(s)}\dot{y}. \tag{12}$$

In this case, we would then have $\dot{x} = \dot{x}_d$ and $\dot{y} = \dot{y}_d$.

If we denote $v = \sqrt{\dot{x}^2 + \dot{y}^2}$, and assume $\dot{s} > 0$, then (12) becomes $\dot{s} = v/\sqrt{p'^2(s) + q'^2(s)}$

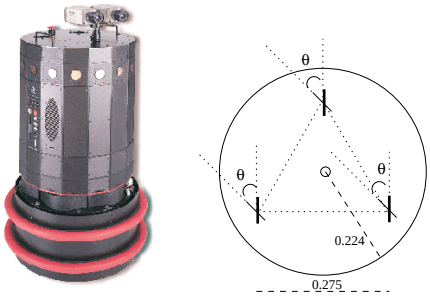
On the other hand, (12) does not contain any position error feedback, which is important for the robustness. Therefore we propose our dynamics for the reference point as follows:

$$\dot{s} = \frac{ce^{-\alpha\rho}v_0}{\sqrt{p'^2(s) + q'^2(s)}}, \tag{13}$$

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

later. Consequently, we need to control both the speed and the steering of the vehicle.

Since we will implement our ideas on a Nomad 200, this platform serves as the basis for the robot model we use for the actual control design.



(a) The Nomad 200.

(b) The synchro-drive wheel system.

Fig. 4. The platform on which the proposed algorithm is implemented.

The Nomad 200 is a synchro-drive tri-wheeler, where the two slave wheels are mechanically turned in the same direction as the actively controlled master wheel, as seen in Figure 4. If we ignore side-slips or mechanical delays between the wheels, we get the well known unicycle model

$$\begin{aligned} \dot{x} &= v\cos\phi \\ \dot{y} &= v\sin\phi, \end{aligned} \tag{14}$$

which is quite similar to the kinematic model of a car, except for the lack of a constraint on the maximal steering angle.

We design our control algorithm as follows:

$$\begin{aligned} v &= \gamma\rho \\ \omega &= ke_\phi + \dot{\phi}_d, \quad k > 0, \end{aligned} \tag{15}$$

where both γ and k are positive, $e_\phi = \phi_d - \phi$ and $\phi_d = \arctan2(e_y, e_x)$.

In Egerstedt *et al.* (1999) it is shown that with this control (15), the steady tracking error ρ can be made as small as one wants and ϕ tends to ϕ_d exponentially.

We thus have a way of both producing and tracking paths, and we now combine these two together into a path following behavior.

5. ACTION FUSION

We now move on to the major point of this paper, namely the action fusion. In the previous sections, we have developed a path-following behavior, produced by the controller $u_{PF}(t)$, and the problem thus becomes that of combining two behaviors so that the robot moves safely, and not too far from optimal with respect to curvature, around a detected obstacle.

Our solution is going to be based on the assumption that there is a minimal distance at which we can detect the obstacle. In an environment that is not extremely dynamic, this is probably a reasonable assumption and it makes it possible for us to identify worst case scenarios where we detect the obstacle at this minimal distance. These scenarios are of interest since they are the ones where the suggested path has the longest part of its curve going through the safety region. The trick now is to choose C_{OA} in (4) numerically in such a way that the combination of our two behaviors makes the robot stay away from this region, and thus we can be sure that it will stay away from this region when we detect the obstacle further away. The results from this idea can be seen in Figure 5.

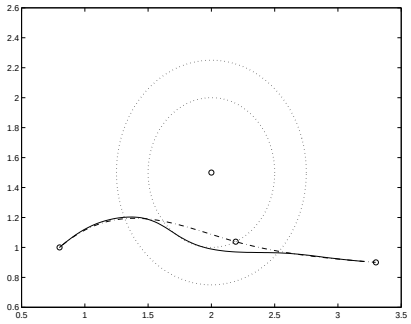


Fig. 5. Numerical tuning of the weight gave us that C_{OA} should be set to 0.9. The inner dotted circle is the safety region, while the outer is the border at which obstacle-avoidance becomes active. The dash-dotted line corresponds to the planned trajectory and the solid line is the actual robot response.

If, however, the robot do encounter obstacles that are closer than the assumed minimal distance, this will not in general be a problem since the obstacle-avoidance will just repel the robot away from that obstacle.

As seen in Figure 6, the method seems to work well.

6. CONCLUSION

In this paper, we have outlined a strategy for solving the action fusion problem that arises when moving a robot, within a behavior-based framework, between two points. Our method treats the avoidance of obstacles in such a way that we planned a geometric reference trajectory around the obstacle for the robot to follow. This path is chosen so that it makes the robot move at

(a) (b)

Fig. 6. Here, the result of implementing our ideas on the Nomad 200 can be seen. The reason why the sonar readings seem rather inaccurate is due to the fact that the robot has some drift in the odometry at the same time as the sonar resolution is rather coarse.

a safe distance from the robot, at least at one distinct point, while keeping the curvature of the path small, since this is a reasonable performance criterion. We have also shown how the robot should track this planned trajectory in a stable way, which results in a path following behavior. We then combine this behavior with a reactive obstacle avoidance behavior in such a way that the safety distance requirement is satisfied by choosing the weight on the obstacle-avoidance response based on a worst case scenario.

7. REFERENCES

Arkin, R.C. (1998). *Behavior-Based Robotics*, The MIT Press, Cambridge, Massachusetts.

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

Canudas de Wit C., B. Siciliano and G. Bastin (1996). Theory of Robot Control, *Springer Verlag.*

Egerstedt M., X. Hu and A. Stotsky (1999). Control of Mobile Platforms Using a Virtual Vehicle Approach. Preprint. Submitted to *IEEE Transactions on Automatic Control.*

Jiang Z. and H. Nijmeijer (1997). Tracking Control of Mobile Robots: A case study in Backstepping, *Automatica*, vol.33, N7,pp. 1393-1399.

Krogh B. and C. Thorpe (1986). 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.

Large E., H.I. Christensen and R. Bajcsy (1997). Dynamic Robot Planning: Cooperation through-Competition, in *IEEE International Conference on Robotics and Automation 1997*, Vol. 3, pp. 2306–2312.

Latombe J.C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers.

Schöner G. and M. Dose (1992). A Dynamical Systems Approach to Task-Level Integration Used to Plan and Control Autonomous Vehicle Motion, *Robotics and Autonomous Systems*, Vol. 10, pp. 253-267.

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

Tomlin C.J., G.J. Papas, J. Košecká, J. Lygeros and S.S. Sastry (1998). 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.