

A Time and Energy Optimal Controller for Mobile Robots

Sebastien Ancenay¹ and Frederic Maire²

¹ Institut National des Sciences Appliquees, Batiment 502, 20 Avenue Albert Einstein,
69621 Villeurbanne, France.

² Smart Devices Laboratory, School of SEDC, IT Faculty,
Queensland University of Technology, 2 George Street, GPO Box 2434,
Brisbane QLD 4001, Australia.

s.ancenay@student.qut.edu.au sebastien.ancenay@insa-lyon.fr
f.maire@qut.edu.au

Abstract. We present a time and energy optimal controller for a two-wheeled differentially driven robot. We call a *mission* the task of bringing the robot from an initial state to a desired final state (a state is the aggregate vector of the position and velocity vectors). The proposed controller is time optimal in the sense that it can determine the minimum amount of time required to perform a mission. The controller is energy optimal in the sense that given a time constraint of n seconds, the controller can determine what is the most energy efficient sequence of accelerations to complete the mission in n seconds.

1 Introduction

Robotic soccer [1] is a challenging research domain which involves teams of agents that need to collaborate in an adversarial environment. The fast paced nature of robotic soccer necessitates real time sensing coupled with quick behaving and decision-making, and makes robotic soccer an excellent test-bed for innovative and novel techniques in robot control. For example, the behaviours and decision making processes can range from the most simple reactive behaviours, such as moving directly towards the ball, to arbitrarily complex reasoning procedures that take into account the actions and perceived strategies of team-mates and opponents. In order to be able to successfully collaborate, agents require robust basic skills. These skills include the ability to go to a given place on the playing

field, the ability to avoid obstacles and the ability to dribble the ball. This paper concentrates on the design of a low level controller to perform elementary missions. We show how to compute off-line optimal trajectories using quadratic programming and how to use these results to build a real time controller.

The remainder of the paper is organized as follows. In Section 2, review related work. In Section 3, we describe our quadratic programming approach. In Section 4, and Section 5, we present two methods to solve the quadratic programming optimization problem of Section 3. In Section 6, we describe experimental results.

2 Previous Work

Many low-level robot controllers create some virtual potential to guide the motion of robots. Recent work based on repulsive potential fields by researchers from LAAS-CNRS [3] is representative of this approach. These potential methods handle well obstacle avoidance and path planning. Each obstacle produces an additive virtual potential field. The robot follows the gradient vectors to travel in the lower valleys of the potential field. The non-holonomic path deformation method [6] is a generic approach of the on-line trajectory deformation issue. It enables to deform a trajectory at execution time so that it moves away from obstacles and that the non-holonomic constraints of the system keep satisfied. Perturbations are also represented with potential fields.

Potential fields can be stacked up. For robot soccer [9], a base field is built where potential values decrease towards the opponent goal to force robots to play closer to that area. A robot's position field encourages the robots to remain in their role positions to provide better robot dispersion around the playing field. Another field is used to represent obstacles and clear path to the ball.

Low-level navigation controllers can be integrated in SLAM (Simultaneous Mapping and Localisation) method [4,5]. In [4], large scale non-linear optimization algorithms and extended Kalman filters were used to compute trajectories.

In [10], a new path planning technique for a flexible wire robot is presented. The authors introduced a parametrization designed to represent low-energy configurations and three different techniques for minimizing energy within the self-motion manifold of the curve.

In [2], a differential evolution algorithm is used to solve the path planning problem by finding the shortest path between start and goal points. This is the closest method to ours. But, the method presented in [2] does not guarantee the optimality of the returned solution (as it relies on an evolution algorithm).

3 A quadratic Programming Approach

Energy efficient path planning can be formulated as a quadratic problem [7]. As the source of power of a mobile robot is an on-board battery, energy is a limited resource that should be consumed sparingly over a whole soccer game. By minimising the sum of the accelerations subject to some constraints, we obtain the most economical (energy-efficient) sequence of accelerations. An added benefit of this approach is that the robot trajectory is very smooth.

The dynamics of a punctual robot follow Newton's laws. A trajectory of such a robot is completely determined by the initial state of the robot (position and velocity) and subsequent sequence of accelerations. The variables P , V and A will denote respectively the position vector, velocity vector $V = \dot{P}$ and acceleration vector $A = \dot{V}$. The trajectory is discretized into n time steps of duration Δ . The derivation of P^i , the position vector at time i , and V^i the velocity vector at time i is straightforward;

$$\begin{aligned} & \begin{cases} V^i = V^0 + \Delta \sum_{j=1}^i A^j \\ P^i = P^0 + \Delta \sum_{j=1}^i V^j \end{cases} \\ & P^i = P^0 + \Delta \sum_{j=1}^i \left(V^0 + \Delta \sum_{k=1}^j A^k \right) \\ & = P^0 + \Delta i V^0 + \Delta^2 \sum_{j=1}^i \sum_{k=1}^j A^k \\ & = P^0 + \Delta i V^0 + \Delta^2 \sum_{j=1}^i (i - (j - 1)) A^j \end{aligned}$$

Treating the x and y coordinates separately presents several computational benefits including a dimension reduction. A 2D path is a linear combination of two 1D paths (see Fig. 1 and Fig. 2). A *mission* is the task of bringing in an energy efficient way the state of the robot from an initial state (P^0, V^0) to a desired final state (P^f, V^f) in a given number n of time steps. Two problems can be distinguished. The first problem is to find the minimum n_0 such that the mission can be completed in n_0 time steps given the physical limitations of the robots (maximum possible speed and acceleration). The second problem is, given $n \geq n_0$, find the sequence of

accelerations that minimizes the energy consumption. The cost function to be

optimized is $\min \sum_{k=1}^n \|A^k\|^2$.

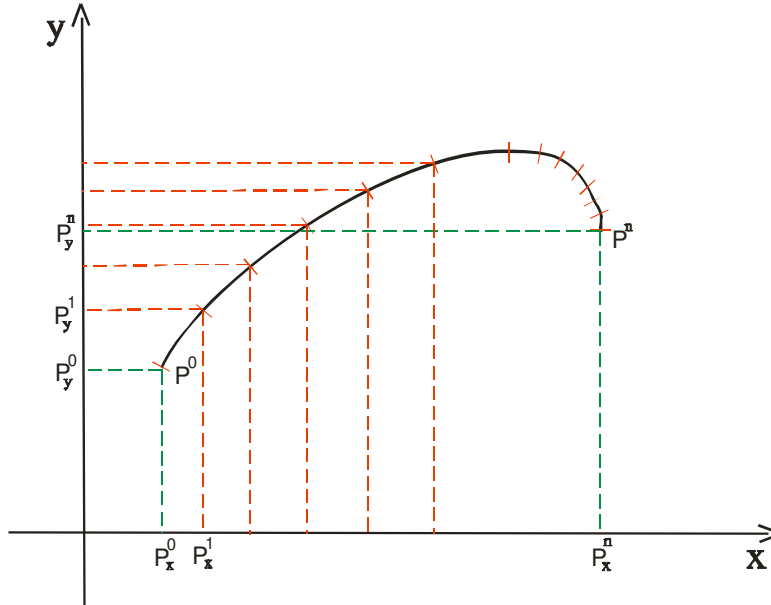


Fig. 1. A 2D path is a linear combination of two 1D paths

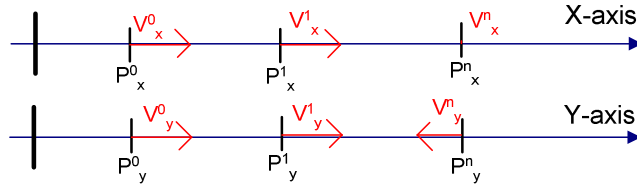


Fig. 2. The two 1D paths corresponding to the 2D path of Fig. 1.

To solve a 2D mission, we solve two 1D missions (see Fig 2). Considering the initial state S^0 represented by the couple of values (P^0, V^0) , we want to find out the sequence of accelerations to reach the final state $S^n=(P^n, V^n)$ while minimizing the energy consumed. In order to be able to recombine the two 1D solutions, the two 1D solutions must have the same number of steps. Fig. 3 sketches the recombination process. After computing the two initial 1D solutions, we must ensure that they have the same number of steps $N_x = N_y$ in order to recombine be able to merge them. We recompute the shortest (with respect to the number of time steps) solution with the maximum of N_x and N_y . If there exists a feasible solution in N_0 time steps,

then a feasible solution exists for every $N \geq N_0$. In the rest of the paper, we only consider 1D mission.

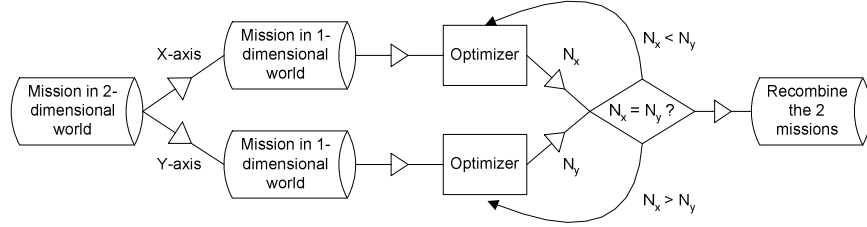


Fig 3. 2D solution are derived from 1D solutions.

There are three different constraints that must be satisfied. The final state S^f (position P^f , velocity V^f) of the robot must be reached after a given number of steps n . The norm of velocity vector must be bounded at any time to reflect the physical capabilities of the robot. This constraint is itself translated into a constraint on the acceleration vectors.

$$\forall i \in [1; n], \quad \|A^i\| \leq A_{\max}$$

$$\forall i \in [1; n], \quad \|V^i\| \leq V_{\max}$$

$$P^n = P^f$$

$$V^n = V^f$$

4 The Quadratic Programming Optimizer

The first method, we investigate uses quadratic programming. We will translate the constraint that the robot must be in state S^f at time n , and the bounded velocity constraints into a system of linear inequalities. Notice, that for ease of computation, we will use the norm 1 instead of the Euclidian norm.

The position constraint expressing that the robot must be in state S^f at time n yields $\sum_{j=1}^n (n - (j - 1))A^j = \frac{P^f - P^0 - n\Delta V^0}{\Delta^2}$. This equality is of the form

$$MA = b, \text{ where } M = [n \quad n-1 \quad \dots \quad 1] \text{ and } b = \frac{P^f - P^0 - n\Delta V^0}{\Delta^2}.$$

Similarly, the final velocity constraint yields that $V^n = V^0 + \Delta \sum_{j=1}^n A^j = V^f$. That

is, $\sum_{j=1}^n A^j = \frac{V^f - V^0}{\Delta}$. Again, a constraint of the form $MA = b$.

The bounded velocity constraints yields a system of inequalities of the form

$$MA \leq b, \text{ where } M = \begin{bmatrix} -1 & 0 & \cdots & 0 & 0 \\ -1 & -1 & & & 0 \\ \vdots & & \ddots & & \vdots \\ -1 & & & -1 & 0 \\ -1 & -1 & \cdots & -1 & -1 \end{bmatrix} \text{ and } b = \begin{bmatrix} \frac{V_{\max} + V^0}{\Delta} \\ \vdots \\ \frac{V_{\max} + V^0}{\Delta} \\ \Delta \end{bmatrix}$$

To estimate the minimum number of steps n_0 needed to perform the mission, a binary search is used (see Fig. 4). Starting with a small value for n , we first search an upper bound that give a feasible solution. Then, we perform a standard binary search.

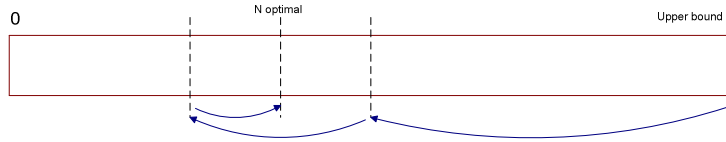


Fig 4. Binary search for the minimal n_0

5 The Graphical Optimizer

In this section, we reduce the search for an optimal sequence of accelerations to a shortest path problem in a graph. The shortest path can be computed using Dijkstra algorithm. Let discretize the state space. The vertices of the graph are the possible states of robots. Some states are not be reachable from the current state. The current velocity determines the range of states we can reach the next time step (see Fig 5). We set the length of the arcs (when they exist) between states with the square acceleration values.

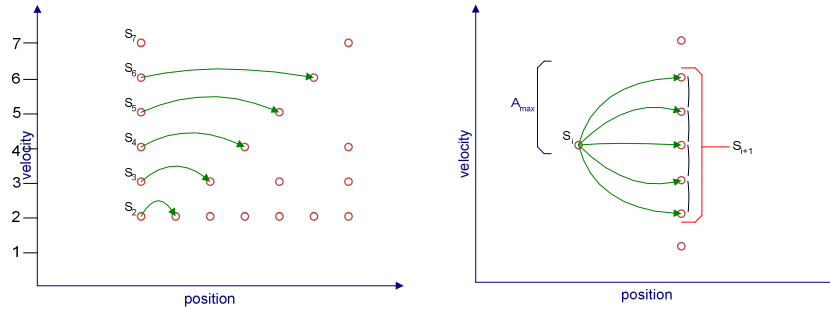


Fig 6 The discretized state space. The next position is determined by the current velocity

The minimal number of steps required for a mission is the number of arcs of the shortest path.

7 Experiments

In the modelling, we assume that the robot is a punctual point. Velocities and accelerations are applied on that point. However our real robot has two wheels. We have to relate the acceleration of the punctual robot and the wheel speed commands of the real robot.

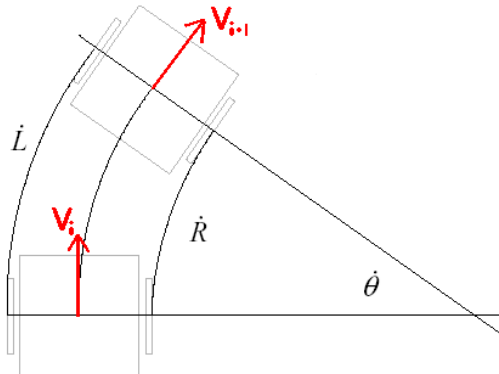


Fig 7. The left and right wheel speeds are related to the velocity of the centre of mass of the two-wheeled robot.

The optimizer provides the next acceleration vector to apply that the centre of mass of the robot should have. The wheel speeds \dot{L} and \dot{R} (left and right) must satisfy $\dot{L} = \|V_{i+1}\| - d \times \dot{\theta}$ and $\dot{R} = \|V_{i+1}\| + d \times \dot{\theta}$, where d is the distance between the centre of the robot and a wheel, and $\dot{\theta}$ is the angular speed. We have

$|\dot{\theta}| = \cos^{-1} \left(\frac{V_{i+1} \cdot V_i}{\|V_{i+1}\| \cdot \|V_i\|} \right)$. The sign of $\dot{\theta}$ is determined with cross product between V_{i+1} and V_i .

We used Matlab optimisation toolbox to implement the quadratic programming method.

In the example below, the initial position is at (0; 0) with a velocity of (0; 0.4). The final state is at position (0.4; 0) with the same velocity (0; 0.4).

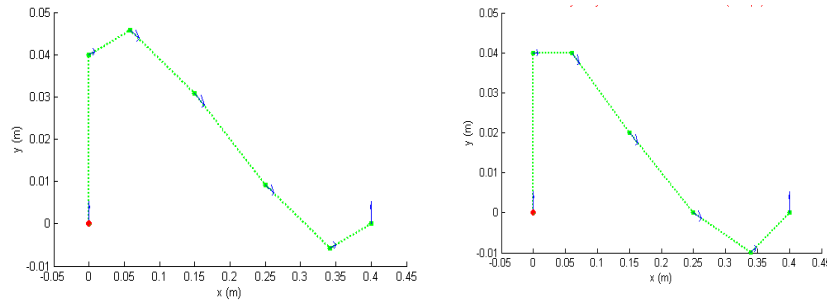


Fig 8. $\Delta = 0.1s$, $V_{\max} = 1m.s^{-1}$, $A_{\max} = 40m.s^{-2}$. Left; quadratic programming solution. Right; dynamic programming (graph) solution.

Both methods return the same optimal number of steps although the optimal paths are slightly different due to the discretization of the graphical method. In Fig. 9, the number of time steps is twice as large that of Fig. 8. The trajectory is smoother. The other parameters are the same except for the resolution of the state graph.

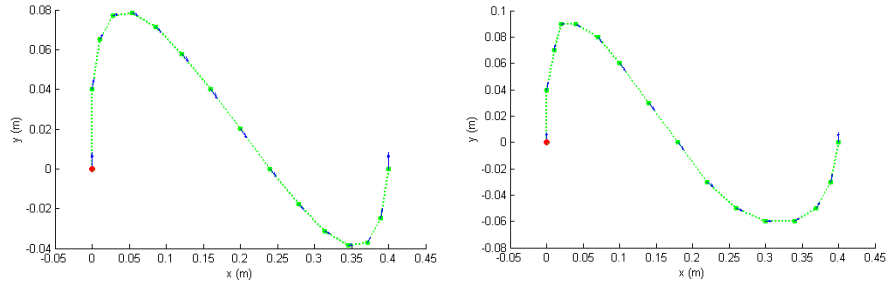


Fig 9. $\Delta = 0.1s$, $V_{\max} = 1m.s^{-1}$, $A_{\max} = 40m.s^{-2}$. Left; quadratic programming solution. Right; dynamic programming (graph) solution. The number of time steps is twice that of Fig. 8.

Experiments were also done on a Mirosoft soccer field (dimensions set by FIRA [1]). The length of the field is about one meter and a half. The control system, running on

a 1.6 GHz computer, is sending wheel speed commands to a remote controlled robot. The only sensor of the system is an overhead camera above the playing field. A vision system tracks the positions of the moving objects on the fields.

Our experimental method requires tracking the position of the robot at each step for a given mission. At each time step, the control system determines the closest mission amongst the one computed off-line by the optimizer and retrieves the next acceleration vector and applies it to the robot. To sort and access data, we use a fast tree indexing system that was introduced in [8]. The mission is continuously revised.

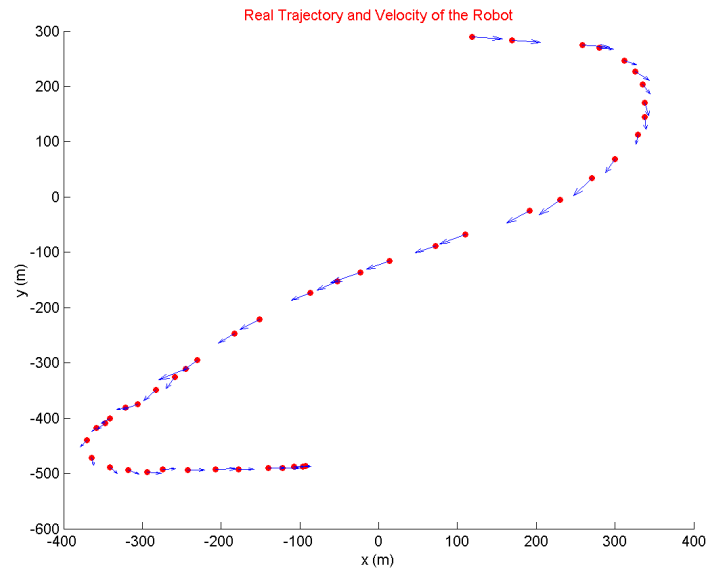


Fig 10. Trajectory of the robot with $\epsilon=4\text{cm}$

In the example shown in Fig. 10, we have similar conditions at those in the mission as those of the mission of Fig. 9. For that, the initial velocity of the robot is set to $(200,0)$. The same 'S' shape is observed as expected.

The position and velocity estimates of the robot returned by the vision system used were noisy. But, as the mission is updated at each frame, the controller can handle inaccurate estimates. A more robust approach would be to use a Kalman filter to track the robot.

Conclusion and Future Work

The quadratic programming formulation for robot control was first introduced in [7] for simulated robots. At that time, we had not realized that the 2D missions could be

reduced to 1D missions. The other innovation of the present paper is the resolution of the optimization problem with dynamic programming (Dijkstra algorithm). This is also the first time, that we have applied the control system to a real robot (and demonstrated that it works as well as in simulation).

Our approach can be easily extended to 3D problems (for aircrafts or submarines). For the future, we plan to implement the object interception behaviours that we described in [7] on a real robot.

References

- [1] FIRA, Federation of International Robot-soccer Association, <http://www.fira.net/> Mirosot robot-soccer competition, <http://www.fira.net/soccer/mirosot/overview.html>
- [2] Hélder Santos, José Mendes, *Path Planning Optimization Using the Differential Evolution Algorithm*, Universidade de Tras-os-Montes e Alto Douro, Departamento de Engenharias, <http://robotica2003.ist.utl.pt/main/Docs/Papers/ROB03-S1-4.pdf>, 2003
- [3] Florent Lamiroux, David Bonnafous, Carl Van Geem, *Path Optimization for Nonholonomic Systems : Application to Reactive Obstacle Avoidance and Path Planning*, CNRS, France, 2002.
- [4] P. Newman, J. Leonard, *Pure Range-Only Sub-Sea SLAM*, Massachusetts Institute of Technology, USA, 2002.
- [5] Robert Sim, Gregory Dudek, Nicholas Roy, *Online Control Policy Optimization for Minimizing Map Uncertainty during Exploration*, 2003.
- [6] Olivier Lefebvre, Florent Lamiroux, Cedric Pradalier, *Obstacles Avoidance for Car-Like Robots Integration And Experimentation on Two Robots*, CNRS-INRIA, France, 2002.
- [7] Frederic Maire, Doug Taylor, *A Quadratic Programming Formulation of a Moving Ball Interception and Shooting Behaviour and its Application to Neural Network Control*, CITI, Faculty of IT, QUT, Australia, 2000.
- [8] Sebastian Bader, Frederic Maire, *A Fast And Adaptive Indexing System For Codebooks*, ICONIP'02, November 18-22, 2002,
- [9] Gordon Wyeth and Ashley Tews, *Using Centralised Control and Potential Fields for Multi-robot Cooperation in Robotic Soccer*, University of Queensland, Australia, 1998.

[10] Mark Moll, Lydia Kavraki, *Path Planning for Minimal Energy Curves of Constant Length*, Department of Computer Science, Rice University, Houston, USA, 2003.