This work-in-progress paper was presented as part of the main technical program at IEEE ETFA'2011

An heuristic for trajectory generation in mobile robotics

Pedro Fonseca, António Neves, José Luís Azevedo and João Silva
IEETA / Univ. de Aveiro
Campus Universitario
3810-193 AVEIRO, Portugal
{pf,an,jla,joao.m.silva}@ua.pt

Abstract

We present an heuristic to compute the points in a trajectory in a 2D space based on the trapezoidal velocity profile, where the computed trajectory is subject to the constraints of initial position and velocity, final position and velocity (defined as vectors) and the values for acceleration and plateau speed (defined as scalars).

The proposed heuristics are directed at omnidirectional holonomic robots, i.e., robots that are capable of, amongst others, manoeuvring without affecting the orientation.

These algorithms are currently being applied to the CAMBADA team RoboCup MSL robots.Although without a formal proof, numerical experiments have shown that the algorithms converged to a viable solution when the data fulfils the necessary conditions.

1. Introduction

In mobile wheeled robots, the problem of generating a trajectory for the robot to follow is a major task to perform. In some cases, it is acceptable to coarsely define a trajectory, simply by means of a target point (a destination), where the robot moves to, while avoiding obstacles in the way. No special constraints are posed to the robot movement during displacement.

In some other situations, such a coarse definition for the robot's trajectory is not acceptable and a more finely grained definition is required. This occurs, for instance, in the Robot Soccer Middle-Size League (MSL) Robocup competitions, when the players try to perform dribbling: manoeuvering the ball in its possession while progressing towards the goal and avoiding the opponents. This action requires a precise and fine-grained definition of the robot's movement.

The CAMBADA robotic MSL team recently reached a point where a more sophisticated way of generating the robot's trajectories was required. For that, we developed an heuristic for generating the robot trajectories, defined as a sequence of points in space that define the robot's position at regularly space, consecutive time instants and that are fed to the robot's control system at a fixed rate (currently, every 33ms). As the only constraints posed on the robot path are those defined by the robot's dynamics, the proposed heuristics are directed at omnidirectional holonomic robots, i.e., robots that are capable of, amongst others, manoeuvring without affecting the orientation [3].

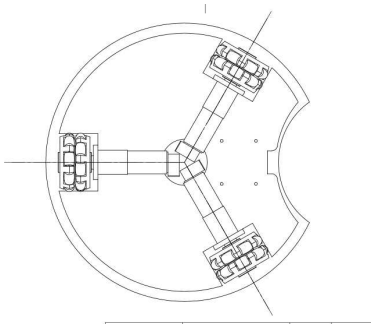

Figure 1. CAMBADA's locomotion system.

2. The CAMBADA robots

CAMBADA is the MSL robot soccer team from the University of Aveiro, in Portugal [2]. CAMBADA was MSL World Champion in 2008 and ranked third in the 2009 and 2010 competitions.

The robot's locomotiom system is based on 3 omnidirectional wheels, aligned at 120º (figure 1). These wheels, also called Swedish wheels [1], have rollers that allow for unconstrained movement along the wheel axis (perpendicular to the wheel's motion). This configuration of the robot wheels allows for holonomic motion.

3. Assumptions

We assume that the robot has some kind of location system that allows it to correctly estimate its position and orientation with respect to a global reference frame with a finite error bound. The algorithm computes and provides the robot a time sequence of spatial positions as points in this global reference frame that the robot will follow.

The proposed algorithm is directed at omnidirectional holonomic robots. This allows us to consider separately

978-1-4577-0018-7/11/$26.00

the computation of the robot's location and orientation and to decouple the problem of trajectory estimation in two separate problems: the computation of the trajectory in the XY plane and the computation of the orientation Θ. Even though, we will require these two movements (position, in 2D, and orientation, in 1D) to be synchronised, meaning that they start and finish at the same time.

The trajectories generated by the algorithm are a variant of the well known and widely used trapezoidal velocity profile: an initial acceleration phase (phase 1), a phase of travelling at constant speed (phase 2) and a final deceleration phase (phase 3).

The movement to be computed is defined by:

- the initial conditions: position, p_i, and velocity, v_i;
- the target or final conditions: position, p_f, and velocity, v_f.
- the absolute value, or norm, for the plateau speed, v_p^*;
- the absolute value, or norm, for the accelerations during phase 1, a_1^*, and phase 3, a_3^* m

We will use the convention of noting with an asterisk * the symbols for quantities expressed as the absolute value or the norm of the corresponding physical quantity. For instance, the plateau speed is specified as v_p^* (a real, positive scalar); the algorithm will compute the plateau velocity v_p, such that $|v_p| = v_p^*$.

We assume that the accelerations in phases 1, a_1^*, and 3, a_3^*, can be different. This can correspond, for instance, to the fact that friction acts in opposition to the force exerted by the motors when the robot increases speed and acts reinforcing this force when it decreases speed.

The problem to be solved is thus: given the initial and final values for position and velocity, p_i, v_i, p_f, v_f; given the scalar values for the plateau speed, v_p^*, and for the accelerations a_1^* and a_3^*, compute a sequence $p_0, p_1, \ldots, p_n$ of $n+1$ points in the XY space, equally space in time and corresponding to instants separated by a time interval T_a, such that:

- $p_0 = p_i$: start point for the movement;
- $p_n = p_f$: end point for the movement;
- $p_1 - p_o = v_i T_a$: initial speed;
- $p_n - p_{n-1} = v_f T_a$: terminal speed;
- $|v_{k+1} - v_k| = a_1^* T_a$, for $k \leq k_1$: acceleration during phase 1 (k_1 is the last sampling moment of phase 1)
- $|v_{k+1} - v_k| = a_3^* T_a$, for $k \geq k_2$: acceleration during phase 3 (k_2 is the last sampling moment of phase 2)
- $|p_{k+1} - p_k| = v_p^* T_a$ for $k_1 < k < k_2$: plateau speed

The robot will attempt to travel the distance s in a movement in three phases:

- in phase 1, it will change its velocity to align with a plateau speed, subject to an acceleration a_1^*;
- in phase 2, it will travel at constant, plateau speed; and
- in phase 3, it will change its velocity from plateau velocity to the target velocity, subject to an acceleration a_3^*.

When phase 3 ends, and the robot velocity is the target velocity, the robot's position is the target destination.

4. Heuristic for path computing

The main point in the heuristic is to find the value for the plateau velocity, v_p. Consider p_α as the point where the robot enters phase 2 and p_ω the point where robot leaves phase 2. In between these two points, the robot travels in a straight line, at constant speed $v = v_p$. It is easy to see that v_p is aligned with $p_\omega - p_\alpha$.

The problem here lies in the fact that v_p is defined by p_α and p_ω, which, in turn, depend on v_p (different values of v_p will yield different values of p_α, for the same v_i; the same applies to p_ω).

To solve this interdependency, *mov2d* (algorithm 1) works iteratively to find a solution for v_p. A first estimate for v_p is to align v_p with $p_f - p_i$, resulting in $v_p^{(0)}$. From p_i, v_i, p_f, v_f and $v_p^{(0)}$, the algorithm computes p_α and p_ω. p_α and p_ω now yield v_π, an estimate for v_p, such that $v_\pi \parallel (p_\alpha - p_\omega)$ and $|v_\pi| = v_p^*$. v_π is combined with the previous estimate using the convergence function h, and the process continues until convergence. In this algorithm, the counter i can be used to terminate the algorithm if it does not converge within a finite number of iterations.

Convergence criteria is defined by function ϕ. This function evaluates to TRUE when the values of v_p, p_α and p_ω provide a "sufficiently good" path for the robot to follow. Algorithm 2 presents one possible definition for ϕ, where the function computes an estimate for Δs, the difference between the travelling in phase 2 for the velocities $v_p^{(i-1)}$ (the estimate for plateau velocity in the previous iteration) and v_π (the estimate in the current iteration), and compares it to a given threshold, using the fact that the triangle with sides $v_p^{(i-1)}$ and v_π is similar to the triangle with sides $s_p^{(i-1)}$ and s_π.

Algorithm 3 presents one possible implementation for h, the function that computes the new estimate for v_p based on the previous estimate and the temporary estimate v_π. The value k can be read as: "the ratio of the length of the projection of s_π in s to the length of s". The role of factor k is to reduce the weight of the vector v_π when the distance travelled in the direction of the overall movement is short. In these situations, the direction of v_π is very sensitive to small changes in p_α and p_ω, increasing instability and preventing convergence of the algorithm.

Algorithm 1 *mov2d*: Heuristic for 2D movement

Require: $s \cdot v_i \geq 0, s \cdot v_f \geq 0$
  $s \leftarrow p_f - p_i$;
  $v_p^{(0)} \leftarrow \hat{s} \cdot v_p^*$ {First estimate for v_p; $\forall u, \hat{u} = \triangleq \frac{u}{|u|}$}
  continue $\leftarrow$ true
  while continue **do**
    {Estimate p_α}
    $\Delta v_1 \leftarrow v_p^{(i)} - v_i$
    $a_1 \leftarrow \hat{\Delta v_1} \cdot a_1^*$
    $s_1 \leftarrow \frac{\Delta v_1}{a_1} \cdot \frac{v_i + v_p}{2}$
    $p_\alpha = p_i + s_1$
    {Estimate p_ω}
    $\Delta v_3 \leftarrow v_f - v_p^{(i)}$
    $a_3 \leftarrow \hat{\Delta v_3} \cdot a_3^*$
    $s_3 \leftarrow \frac{\Delta v_3}{a_3} \cdot \frac{v_p + v_f}{2}$
    $p_\omega \leftarrow p_f - s_3$
    {Estimate displacement at plateau speed}
    $s_\pi \leftarrow p_\omega - p_\alpha$
    $v_\pi \leftarrow \hat{s}_\pi \cdot v_p^*$
    if $\phi(v_p^{(i-1)}, v_\pi, s_\pi)$ **then**
      {ϕ defines the convergence criterion}
      {Final value reached}
      continue $\leftarrow$ false
    else
      {Update estimate}
      $v_p^{(i)} \leftarrow h(s, v_p^{(i-1)}, s_\pi, v_\pi)$
    end if
    Increment i
  end while

Algorithm 2 Definition of convergence criteria function ϕ

  return $|v_p^{(i-1)} - v_\pi| \frac{|s_\pi|}{|v_\pi|} < \epsilon$

Although there is so far no proof of convergence, the algorithm was tested and the results showed that it converged in all situations where:

1. the condition $s \cdot v_f \geq 0$ holds, or, in other words, v_f is not directed opposite to the movement direction;
2. p_i and p_f are sufficiently apart and the robot has enough space to change direction at both ends of the movement (an analytical expression for the minimum distance between p_i and p_f has not yet been determined).

The case where p_i and p_f are not sufficiently separated is detected by the condition $s \cdot s_\pi < 0$, which means that,

Algorithm 3 Definition of h function

Require: s, $v_p^{(i-1)}$, s_π, v_π
  $k \leftarrow \frac{s \cdot s_\pi}{|s|^2}$
  return $kv_\pi + (1-k)v_p^{(i-1)}$

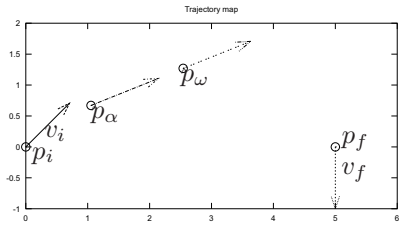

Figure 2. Elements computed by *mov2d*

after computing the estimates for p_α and p_ω, the projection of vector $s_\pi = p_\omega - p_\alpha$ in the direction of $s = p_f - p_i$ points in the direction opposite to s. This situation, as is, has no solution by our algorithm. In this case, reducing the value of v_p^*, and thus reducing the length of vector v_π, has shown to allow for convergence of the algorithm. In the actual implementation, the code in algorithm 4 is inserted in algorithm 1 immediately after computing the value of v_π.

Algorithm 4 Heuristic for degenerate movement

  if $s \cdot s_\pi < 0$ **then**
    $v_p^* \leftarrow \lambda v_p^* \quad \{0 < \lambda < 1\}$
    if $v_p^* < v_{p,\min}^*$ **then**
      {Could not find a viable value for v_p^*}
      return with fail code
    end if
    $v_p^{(i)} \leftarrow \hat{s} \cdot v_p^*$
    Increment i
    continue {Restart loop with new value of v_p^*}
  end if

Algorithm 1 computes the parameters that define the movement, namely p_α, p_ω and v_p. The actual computation of the points in the trajectory is performed by the algorithm *motion_comp*, described in algorithm 5. At the end of *motion_comp*, *pp* is a vector that contains all points in the trajectory.

Figure 2 presents the major elements that define the movement:

- p_i, v_i, p_f and v_f, that were input to *mov2d*;
- p_α, p_ω and v_p, that were computed by *mov2d*.

Figure 3 corresponds to the same trajectory, after superimposing the points computed by *motion_comp*.

The result is a set of points that the robot's control system can track an follow, as shown in figure 4.

5. Conclusions

We present here an heuristic to compute the points in a trajectory in a 2D space based on the trapezoidal velocity profile, where the movement occurs in three phases:

Algorithm 5 *motion_comp*: compute points in a trajectory

{Elements for phase 1}
$\Delta v_1 \leftarrow v_p - v_i$ {Δv in phase 1}
$t_1 \leftarrow \frac{|\Delta v_1|}{a_1^*}$ {Time to complete phase 1}
$n_1 \leftarrow \lceil \frac{t_1}{T_a} \rceil$ {Number of cycles in phase 1}
$a_1 \leftarrow \widehat{\Delta v_1} a_1^*$
{Elements for phase 3}
$\Delta v_3 \leftarrow v_f - v_p$ {Δv in phase 3}
$t_3 \leftarrow \frac{|\Delta v_3|}{a_3^*}$ {Time to complete phase 3}
$n_3 \leftarrow \lceil \frac{t_3}{T_a} \rceil$ {Number of cycles in phase 3}
$a_3 \leftarrow \widehat{\Delta v_3} a_3^*$
{Start computing the points}
$p \leftarrow p_i$; $v \leftarrow v_i$; {Initial values}
{Phase 1}
for $n = 1$ to n_1 **do**
  $v \leftarrow v + a_1 T_a$
  $p \leftarrow p + v T_a$
  $pp \leftarrow p$ {$\mathbf{A} \leftarrow b$: append b to $\mathbf{A}$}
end for
{Phase 2}
$p \leftarrow p_\alpha$; $v \leftarrow v_p$ {Initial point for phase 2}
$pp \leftarrow p$
for $n = 1$ to n_2 **do**
  $p \leftarrow p + v T_a$
  $pp \leftarrow p$
end for
{Phase 3}
$p \leftarrow p_\omega$ {Initial point for phase 3}
$pp \leftarrow p$
for $n = 1$ to n_3 **do**
  $v \leftarrow v + a_3 T_a$
  $p \leftarrow p + v T_a$
  $pp \leftarrow p$
end for
$p \leftarrow p_f$ {Last point in the movement}
$pp \leftarrow p$

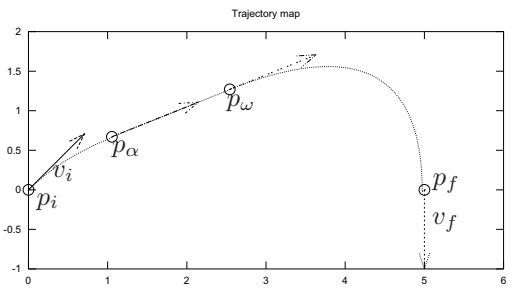

Figure 3. Trajectory computed by *motion_comp*

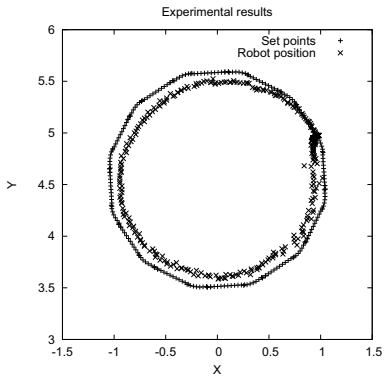

Figure 4. Robot's actual trajectory

initial acceleration from initial velocity to a plateau speed (phase 1), travelling in uniform motion at plateau velocity (phase 2) and acceleration from plateau velocity to final velocity (phase 3). The computed trajectory is subject to the constraints of initial position and velocity, final position and velocity (defined as vectors) and the values for acceleration and plateau speed (defined as scalars).

The results show that these algorithms are capable of producing the correct result in most situations. In some situations (for example, large differences between initial and final velocities in short trajectories) the heuristics are unable to find a solution for the problem.

At current point, there is no formal definition for the conditions that guarantee the existence of a viable solution or the algorithm's success in finding such a viable solution. These involve iterative procedures, for which the proof of convergence is still an open issue. Although without a formal proof, numerical experiments have shown that both algorithms converged to a viable solution when the data fulfils the necessary conditions.

References

[1] J. F. Blumrich. Omnidirectional wheel, Feb. 1974. US Patent 3789947.

[2] A. Neves, J. Azevedo, M. Cunha, N. Lau, A. Pereira, G. Corrente, F. Santos, D. Martins, N. Figueiredo, J. Silva, B. Ribeiro, L. Almeida, L. Lopes, J. Rodrigues, and A. Pinho. CAMBADA2010: Team description paper. In *Proceedings Robocup 2010*, 2010.

[3] R. Siegwart and I. R. Nourbakhsh. *Introduction to autonomous mobile robots*. MIT Press, Cambridge Mass., 2004.