Online Trajectory Generation for Mobile Robots with Kinodynamic Constraints and Embedded Control Systems

Marcello Bonfè∗ Cristian Secchi∗∗ Enea Scioni∗

∗ Engineering Department, University of Ferrara, 44122 Ferrara, Italy.
E-mail: marcello.bonfe@unife.it, enea.scioni@unife.it.
∗∗ DISMI, University of Modena and Reggio Emilia, 42122 Reggio Emilia, Italy. E-mail: cristian.sechhi@unimore.it

Abstract: The paper describes trajectory generation and tracking control algorithms, respectively based on nonlinear filtering and dynamic feedback linearization, for mobile robots. A main feature of proposed algorithms is that they are suitable for the implementation on embedded systems with limited computational resources. The trajectory generator is based on nonlinear filters and logic-based management of reference inputs and dynamic constraints, allowing online smoothing of straight-line reference paths. Sparse via-points along a path can be assigned by a global planner based on obstacle avoidance algorithms and can be changed at any time during motion. Moreover, the trajectory generated by the nonlinear filter can be fed into a control loop based on the dynamic model of the robot, so that accurate tracking can be achieved. The paper includes practical remarks for efficient fixed-point implementation of the proposed trajectory generator.

Keywords: Mobile robots, Nonlinear filters, Feedback linearization, Robot dynamics, Embedded systems.

1. INTRODUCTION

Generation and accurate tracking of smooth motion trajectories, subject to kinematic and dynamic constraints, are fundamental issues in robotics. Motion planning is usually separated into the geometric problem (path planning) and the actual trajectory planning, in which the timing law for a given path is designed. Path planning for wheeled mobile robots must take into account holonomic constraints and/or obstacles, while constraints involved by dynamics laws and bounded velocity/acceleration must be addressed during the design of either timing plans or tracking control algorithms, see Siciliano et al. (2009) (Ch. 4,7,8). The literature on path planning describes many solutions to avoid collision with static or moving obstacles, see LaValle (2006). If static obstacles are considered, robot motion can be planned in advance and efficient, but computationally demanding, interpolation methods can be applied. However, when the tasks of the robot or the positions of obstacles are not fully known a priori, paths must be adapted or re-planned online, within hard real-time constraints. Moreover, any path must be associated with a feasible timing law to become a trajectory compatible with the dynamic features of the robot.

The trajectory generation solution first introduced by Bonfè and Secchi (2010) is specifically designed for online execution, thanks to its limited computational demand and to its discrete-time behavior. The output of the proposed trajectory generator is fully specified (in cartesian coordinates) with respect to time and has continuous curvature, so that it is compatible with with kinodynamic constraints (i.e. bounded linear/angular velocity and acceleration) of unicycle-like robots. The input of the trajectory generator can be either a sequence of fixed via-points, along a straight-line path, or a time-varying reference point, which may be provided by global planning algorithms implementing obstacle avoidance, but not necessarily precise geometric path design, since the smoothing action of the trajectory generator compensates path discontinuities. This paper extends the contribution of Bonfè and Secchi (2010) by analysing the geometric features of generated trajectories and by presenting some obstacle-avoidance use cases (Section 4). Moreover, Section 5 contains practical guidelines for the implementation of the trajectory generator on a low-cost DSP or microcontroller and reports real experimental results.

2. RELATED WORK

The problem of motion planning for wheeled robots in presence of kinematic and dynamic constraints has been addressed since seminal papers of Shiller and Gwo (1991) and Donald et al. (1993). Many authors have addressed the problem of generating motor commands or velocity profiles to obtain time-optimal motion along a geometric path, either fully specified in advance or resulting from smoothing algorithms applied to a straight-line path. In fact, it is well-known (see Fraichard and Scheuer (2004)) that even if the minimum length path, connecting two configurations of a wheeled robot in a planar workspace, is a sequence of straight lines and circular arcs, such path cannot be perfectly tracked by a mobile robot, because of discontinuity in the acceleration. Several path smooth-
ing techniques, producing a final path with continuous curvature, are available from the literature. For example, Connors and Elkaim (2007) used B-splines, while Lau et al. (2009) used Bezier splines. An issue related to such a kind of spline curves is the fact that moving a single control-point may affect several adjacent segments, which makes quite difficult their online modification. With the approach proposed in this paper, the trajectory is inherently generated online, as the output of a nonlinear smoothing filter. The latter receives as input a target point, which is the equivalent of a control-point for spline interpolation, with the key difference that the target point of the nonlinear filter can be changed at any time during motion and the filter will generate a trajectory reaching smoothly the newer target. Moreover, as will be described in Section 4, if the target point is changed according to a properly designed sequence, the trajectory generated by the nonlinear filter has geometric features similar to those of the paths described by Fraichard and Schuener (2004).

The search for time-optimal motion plans on specified paths, compatibly with kinodynamic constraints, is another topic that has been studied by many authors, like Lau et al. (2009); Sprunk et al. (2011); Howard and Kelly (2007). One of the contributions of mentioned and related references is certainly the definition of computationally efficient optimization algorithms. On the other hand, experiments described in those papers are made with high-performance PCs. The trajectory generator proposed in this paper can instead be executed by a low-cost 16-bit microcontroller in less than 700 μs (see Section 5). Moreover, the one-dimensional nonlinear filter described at the beginning of next section guarantees that the reference input is reached in minimum time, compatibly with dynamic constraints. This means that the two-dimensional trajectory generator proposed by Bonfè and Secchi (2010) applies the highest admissible linear and angular velocities at any time. Formal proofs of such properties are based on the theory of Sliding Mode control (Utkin (1977)).

To conclude, real-time path tracking is the last issue to address for efficient motion control of a mobile robot. Accurate tracking requires error-feedback controllers running in a real-time loop not affecting the trajectory itself, which is the most common scheme in manipulation robotics. Control design techniques based on dynamic inversion or feedback linearization have been proposed by d’Andrea Novel et al. (1995) and Oriolo et al. (2002). In these works, linear and steering velocities are used as robot control inputs, which are usually the only commands accepted by commercial mobile robots. The algorithms proposed in this paper have been tested using an in-house developed platform, allowing to specify driving force and steering torque as control inputs. Therefore, it has been possible to extend the feedback linearization method of Oriolo et al. (2002), taking into account the dynamics of the robot and achieving smoother control and accurate tracking performances.

3. TRAJECTORY GENERATION AND TRACKING

Nonlinear smoothing filters

The nonlinear filtering approach to online trajectory generation, for one dimensional motion, has been exploited first by Zanasi et al. (2000), in a paper that describes the design of a Variable Structure (VS) dynamic system acting as a smoothing filter for rough (steps, discontinuous ramps, etc.) position reference signals. The filter achieves perfect tracking of the reference signal in minimum time, compatibly with constraints on the first and second-order derivative of the filter output. The VS system is composed by a chain of two integrators and a nonlinear controller that guarantees the requirement on bounded output derivatives and minimum time response. The block diagram of the filter, in the discrete-time case 1, is shown in Fig.1.

![Fig. 1. Block diagram of a nonlinear filter for trajectory generation](image)

The VS controller of the filter receives at each sampling instant $nT$ the following inputs: the position reference $r_n$ and its time derivative $\dot{r}_n$, the current outputs of the integrators $(\hat{x}_n$ and $\hat{x}_n)$, the bounds $U$ on the acceleration/deceleration and $\hat{x}_M$ on the velocity absolute value. Therefore, such bounds can be changed in real-time. The control law proposed by Zanasi et al. (2000) is a Sliding Mode (SM) controller (see Utkin (1977)), ensuring that the resulting trajectory always reaches a sliding surface in the error phase plane (with coordinates $y_n = x_n - r_n$, and $\dot{y}_n = \dot{x}_n - r_n$), in minimum time, without overshooting and without exceeding the bounds $U$ and $\hat{x}_M$. Once reached the surface, the SM brings the filter towards a perfect tracking condition, which is also achieved in minimum time. Since the VS controller is designed directly in the discrete-time domain, no chattering is introduced by the nonlinear filter (see Zanasi et al. (2000)).

The extension of this filtering method to generate trajectories in the two-dimensional operational space of a nonholonomic mobile robot has been presented by Bonfè and Secchi (2010). The key idea is to generate with two separate nonlinear filters, both structured as shown in Fig. 1, linear velocity and orientation of the trajectory, compatibly with dynamic constraints, and then take into account kinematic constraints to obtain first-order time derivatives of the trajectory in the cartesian space, whose subsequent numerical integration defines the desired position vector. Considering the class of differential-drive (i.e. unicycle-like) wheeled robots, kinematic constraints require the robot configuration vector $[x, y, \theta]^T$, being $\theta$ the orientation of the robot w.r.t. the fixed cartesian frame, to comply with the following differential equation:

$$\begin{align*}
\dot{x} &= v \cos \theta \\
\dot{y} &= v \sin \theta \\
\dot{\theta} &= \omega
\end{align*}$$

1 All the nonlinear filters described in the paper are discrete-time. The sampling instant $nT$ is dropped in all subsequent equations, to simplify notation, and differentiation/integration are improperly denoted as the equivalent continuous-time operations.
in which \(v\) and \(\omega\), respectively driving velocity and steering velocity, are in most applications assumed as the control inputs. Considering only the first two rows of Eq. (1), it is clear that the generation of two sufficiently smooth signals \(v(t)\) and \(\theta(t)\) and the integration of \(\dot{x}\) and \(\dot{y}\) equals to generate a trajectory compatibly with the kinematic model of the unicycle. Dynamic constraints in the generation of \(v(t)\) and \(\theta(t)\) can be taken into account recalling that the acceleration of a planar trajectory is given by the sum of tangential and radial acceleration orthogonal vectors, whose lengths are respectively \(a_t = \dot{v}\) and \(a_r = \dot{\theta} = v \omega\). Both components must be bounded to preserve from lateral slipping and to keep proportionality between wheels velocity and driving velocity. Limiting radial acceleration by reducing (without zeroing) the driving velocity when \(\omega \neq 0\), involves a limitation also on the scalar curvature of the path, since the latter is \(\kappa = \omega/v = \omega^2/a_r\).

Minimization of time and space required to reach a given target position can be achieved by maximizing linear velocity, within actuator limitation, when the robot is oriented towards the target, and instead set driving speed as the ratio between maximum allowed radial acceleration and maximum steering velocity, when the robot must change its orientation to point towards the target. The generation of this change of orientation can be obtained following the so-called planar pursuit-evasion approach and applying the equations describing the geometric relationship between a moving target, whose position and linear velocity are denoted as \([x_t, y_t]^T\) and \(v_t\) in Fig. 2, and a tracking point, characterized by \([x_d, y_d]^T\) and \(v_d\) in the figure. Such pursuit-evasion equations can be found in Lin (1992).

Fig. 2. Target interception geometry

From now on we will assume \(v_t = 0\) (i.e. fixed target positions), but the approach can be easily extended to consider moving targets, as will be shown in Section 4. If the orientation of the trajectory generated by the filter is different from \(\theta_c\), then the trajectory should be modified by introducing a turning phase. During the turning phase necessary to align with the target, steering velocity should be the highest possible, while driving velocity must be set to maximum value compatible with the bound on radial acceleration. Once that target alignment is achieved, the final position can be exactly reached by triggering a deceleration phase with zero final velocity, as soon as the distance from the target is equal to the space required to stop with given bounds on \(\dot{v}_d\) and \(\ddot{v}_d\).

The block diagram of a nonlinear smoothing filter that generates trajectories for a unicycle-like robot, in the way just described, is shown in Fig. 3.

Fig. 3. Block diagram of the nonlinear smoothing filter for mobile robotics

The block Velocity Nonlinear Filter of Fig. 3 is a filter with the structure of Fig. 1, but its output is a driving velocity \(v_d\) (instead of a desired position), that perfectly tracks the discontinuous reference velocity \(v_t\) with bounds on first and second-order derivatives (\(|v_d| \leq v_M\) and \(|\dot{v}_d| \leq U_v\)). The Orientation Nonlinear Filter differs from the previous one only in the calculation of the tracking error, which is limited in the interval \([-\pi; \pi]\). The output of the filter is the desired orientation \(\dot{\theta}_d\) tracking at best \(\theta_c\) with bounded derivatives (\(|\dot{\theta}_d| \leq \dot{\theta}_M\) and \(|\ddot{\theta}_d| \leq \ddot{\theta}_M\)).

The Switching Logic is a finite state machine that sets the reference signals \(v_r\), \(\theta_r\) and \(\theta_c\) (\(\dot{v}_r = 0\) at any time) and the bounds of the orientation filter. The state machine contains four states, namely Turning, Aligned, Stopping and Limit Speed, and it formalizes the target approaching sequence previously described (see Bonfè and Secchi (2010) for the full details of the state machine).

The behavior of the Switching Logic and, therefore, the resulting geometric properties of the trajectories generated by the filter, depend on the following parameters:

- \(v_M\): maximum allowed driving velocity. It is set as a reference for the Velocity Nonlinear Filter only when the trajectory is aligned with the target (state Aligned).
- \(A_{RM}\): maximum allowed radial acceleration. It affects the resulting curvature of the path and it is used to calculate the driving velocity limit in the states Turning and Limit Speed. The latter state is required to slow down before starting a curve.
- \(\dot{\theta}_M\): maximum allowed steering velocity. It is used to limit the rate of change of the orientation along a curve (i.e. Turning state), when the Orientation Nonlinear Filter is forced to track \(\theta_c\).
- \(R_{stop}\): distance required to decelerate from \(v_1 = v_M\) to \(v_r = 0\). This value, used in the guard condition to switch from state Aligned to Stopping, can be easily calculated since the output of the velocity filter is a standard profile with trapezoidal first-order derivative (acceleration, in this case). \(R_{stop}\) is obtained integrating further this velocity profile.

The outputs of the velocity and orientation nonlinear filters must be combined as follows:
\[\dot{x}_d = v_d \cos \theta_d \]
\[\dot{y}_d = v_d \sin \theta_d \]
\[\text{and integrated to finally obtain } [x_d, y_d]^T. \]
Higher order time derivatives \(\dot{x}_d, \dot{y}_d, \ddot{x}_d, \ldots\) can be easily calculated from the full output vectors of the two nonlinear filters, with equations obtained by differentiation of Eq.(2).

**Remark 1.** Target position \([x, y]^T\) can be abruptly changed at any time. This event causes \(\theta_d \neq \theta_e\) and, therefore, forces a reduction of driving velocity, set to \(A_{RM}/\theta_B\) before starting the turning phase. In this way, radial acceleration is guaranteed to be bounded by \(A_{RM}\).

**Remark 2.** The second-order time derivatives of velocity and orientation are bounded, so that also third-order derivatives \([\ddot{x}_d, \ddot{y}_d]^T\) are limited. Moreover, this implies that the curvature of the resulting path is continuous.

**Dynamic feedback linearization**

Perfect trajectory tracking can be achieved, theoretically, by means of dynamic inversion or feedback linearization techniques, provided that the model of the controlled system is perfectly known. This control design method has been applied to the kinematic model of unicycle-like mobile robots by Oriolo et al. (2002). The approach can be extended to obtain feedback linearization of the dynamic model, including mass and inertia of the robot. In particular, Oriolo et al. (2002) showed that the kinematic model of Eq.(1) can be I/O feedback linearized by choosing \(\eta = [x, y]^T\) as the output and adding an integrator before the original input \(v\), so that the I/O decoupling matrix (see Isidori (1995), Chap. 5) becomes nonsingular. This property is maintained if the state of the integrator \(\xi\), which is part of the dynamic compensator, is kept different from zero. Setting \(a = \xi\) and using \(u = [\alpha, \omega]^T\) as the input vector, exact linearization can be achieved by means of a control law based on the inverse of the I/O decoupling matrix, so that the closed-loop dynamics is \(\ddot{\eta} = \nu = [\nu_1, \nu_2]^T\).

This procedure can be applied to linearize the dynamics of a unicycle-like robot, whose model can be written as follows (see Siciliano et al. (2009), Par. 11.4):

\[\begin{align*}
\dot{x} &= v \cos \theta \\
\dot{y} &= v \sin \theta \\
\dot{\theta} &= \omega \\
\dot{\omega} &= \tau_s / J
\end{align*}\]

in which \(\tau_d\) is the driving force, \(\tau_s\) is the steering torque, \(m\) is the total mass and \(J\) is the moment of inertia around the vertical axis. Applying the linear input transformation \(a = \tau_d/m\) and \(\alpha = \tau_s/J\) and adding an integrator before \(a\), such that:

\[a = \xi, \quad \dot{\xi} = j\]

the feedback linearization between the input \(u = [j, \alpha]^T\) and the output \(\eta = [x, y]^T\) can be obtained with the following control law:

\[u = B(v, \theta)^{-1} [\nu - F(v, \eta, \theta, \omega)] \]

in which \(B(v, \theta)\) is the I/O decoupling matrix and \(F(v, \eta, \theta, \omega)\) contains nonlinear terms to be cancelled (see Bonf\'e and Secchi (2010)), so that the linear system \(\ddot{\eta} = \nu\), with \(\nu\) as a new input (which can be exploited to design an additional linear control loop), is obtained. Notice that \(B(v, \theta)\) is invertible as long as \(v \neq 0\), but this issue can be easily addressed following the implementation guidelines described by Oriolo et al. (2002).

**Remark 3.** The state in normal coordinates of the linearized system is \(z = [x, \dot{x}, x, y, \ddot{y}]^T\). The value \(z_d\) of this vector for a desired motion can be calculated online explicitly from the outputs of the previously described trajectory generator. Assuming that \(z\) can also be calculated in real-time, using estimation algorithms based on sensor-fusion of wheel encoders and inertial measurements (which can be obtained from modern low-cost MEMS accelerometers and gyroscopes), accurate trajectory tracking can be achieved applying the following error feedback plus feedforward control law:

\[\nu = [\ddot{x}_d, \ddot{y}_d] + K (z_d - z) \]

in which \(K\) is any \(2 \times 6\) constant matrix that stabilizes the control-loop on the linearized system.

4. VIA-POINT COMMUTATION AND APPLICATION CASE-STUDIES

The role of the discrete-time nonlinear filter described in Fig. 3 is to calculate online a smooth trajectory approaching and reaching a target cartesian position. At each sampling instant, the output vector of the filter is updated according to the error between the current orientation of the trajectory and the one that actually allows to reach the target point, compatibly with kinodynamic constraints. Therefore, the resulting path depends inherently on the possibly time-varying setting of target position. If the aim is to generate a trajectory among obstacles, a key issue that needs to be solved is the adequate placement of a sequence of via-points and, in addition, the definition of conditions for selecting online one of these via-points as the target of the nonlinear smoothing filter. Of course, these tasks could be executed by a higher-level planning algorithm, that is aware of the location of obstacles and of the geometric properties of the trajectory generated by the filter.

In particular, there are two features of the proposed nonlinear filter that must be considered. First of all, the design of the Switching Logic guarantees that radial acceleration is bounded by \(A_{RM}\), by means of a Limit Speed state that precedes any turn. The bound on \(A_{RM}\) implies the bound:

\[\kappa_M = \frac{\dot{\theta}^2_B}{A_{RM}} \]

on the curvature of the trajectory in the Turning state. Moreover, since \(\dot{\theta}_B\) can only vary linearly with a constant rate \(U_{\theta}\), the time derivative of the curvature is also piecewise constant and bounded by:

\[\dot{\kappa}_M = \frac{U_{\theta} \dot{\theta}_B}{A_{RM}} \]

Therefore, the trajectories generated with the proposed approach have the same properties of Continuous-Curvature paths (CC-paths) described by Fraichard and Scheuer (2004) or Szadeczky-Kardoss and Kiss (2008), namely they are composed of linear segments, clothoid arcs (i.e. arcs with linearly increasing curvature) and circular arcs of
radius $\kappa_M^{-1}$. The key differences are that the trajectory generator described in previous section does not compute explicitly the clothoids, since the latter are inherently the output of the nonlinear filter, and that the resulting clothoid arcs are specified w.r.t time, instead of arc length. Moreover, since the driving velocity is exactly $v_d = A_{RM}/\dot{\theta}_B$ in any curved part of the trajectory, by design, there is no need for a time-scaling law to guarantee the geometry of the path or to comply with dynamic constraints, which is instead the approach of Szadeczky-Kardoss and Kiss (2008). The geometric features of CC-paths (see Fraichard and Scheuer (2004)) can be revisited for the case under study considering the trajectory plotted in Fig.4, including a curve near a via-point denoted with $A$. In the figure, the current state of the nonlinear filter Switching Logic is highlighted by the color of the path. As can be seen, the first part of the curve is a clothoid arc, more precisely an arc whose sharpness is:

$$\sigma_M = \frac{\kappa_M}{v_d} = \frac{U_0}{v_d^2}. \quad (9)$$

$$x_{B'} = \sqrt{\frac{\pi}{\sigma_M}} C_F \left( \sqrt{\frac{\kappa_M^2}{\pi \sigma_M}} \right)$$

$$y_{B'} = \sqrt{\frac{\pi}{\sigma_M}} S_F \left( \sqrt{\frac{\kappa_M^2}{\pi \sigma_M}} \right) \quad (10)$$

in which $C_F$ and $S_F$ denote the Fresnel integrals, and that:

$$\theta_{B'} = \frac{\kappa_M^2}{2\sigma_M} \quad (11)$$

It is then possible to calculate the length $BC$, finding the coordinates of $C$ as follows:

$$x_C = x_{B'} - \kappa_M^{-1} \sin \theta_{B'}$$

$$y_C = y_{B'} + \kappa_M \cos \theta_{B'}$$

which means that:

$$BC = \sqrt{x_C^2 + y_C^2} \quad (13)$$

Finally, knowing that the angle between the initial orientation of the trajectory along the clothoid arc and the tangent in $B$ to the circle of radius $BC$ is:

$$\mu = \arctan \left( \frac{x_C}{y_C} \right), \quad (14)$$

that $\gamma_2 = \pi/2 - \mu$, that $\gamma_3$ is half of the angle between the two segments connecting three consecutive via-points and applying known theorems on triangles, we obtain:

$$AB = BC \frac{\sin \gamma_3}{\sin \gamma_1} \quad (15)$$

These results can be extended to consider also the case in which the maximum curvature arc is absent and the curve is composed only by two symmetric clothoid arcs, which happens if the angle between two segments connecting three via-points is larger than a given limit.

It is important to note that, apart from Eq.15, all the other results depend only on the bounds applied to the nonlinear filter. If such bounds are used as constant parameters, the only value that should be calculated online and for each via-point is $AB$. Otherwise, since the bounds of the nonlinear filter can be changed in real-time, this additional degree of freedom can be used to increase the curvature of the trajectory in case of narrow passages, in which the via-point should be approached as much as possible.

**Case study 1: path smoothing among static obstacles**

Obstacle-avoiding paths among static obstacles can be obtained by means of a properly defined interaction between the nonlinear filter and a global planner. The latter can place a set of via-points, using any collision-avoidance algorithm, among the obstacles and can calculate for each via-point the commutation distance (which is actually the sum of $AB$ plus the distance required to reduce linear velocity from $v_M$ to $A_{RM}/\dot{\theta}_B$), so that continuous-curvature trajectories smoothing straight-line paths remain collision-free. The complete sequence of via-points, together with related commutation distance, can be stored in a queue and can be managed online by a simple algorithm, extracting from the queue a via-point to be selected as the current target point of the filter and continuously updating the distance from the output of the filter to the via-point to promptly detect the commutation condition. As an example, Fig. 5 shows
a trajectory generated applying a sequence of fixed via-points, marked by stars, along a path composed of straight lines among obstacles (grey rectangles). The target point of the nonlinear filter is always changed exactly when the filter output is at a distance from the current via-point such that at the end of the turning phase, triggered by the target commutation, the trajectory is exactly oriented along the straight line connecting the current via-point with the subsequent one. The simulation parameters are $v_M = 0.25 \text{ m/s}$, $\dot{v}_M = 0.4 \text{ m/s}^2$, $U_e = 0.6$, $A_{RM} = 0.1 \text{ m/s}^2$, $\theta_M = 0.6 \text{ rad/s}$, $U_0 = 2 \text{ rad/s}^2$.

Fig. 5. Straight-line path among obstacles, with via-points (stars) and resulting smooth trajectory generated by the nonlinear filter (red)

**Case study 2: avoiding moving obstacles**

The target point for the nonlinear filter can also have a given linear velocity. A practical case that may be addressed using this feature of the proposed nonlinear filter is a highway-like context, in which slower vehicles move in the same direction of the controlled mobile robot. If the robot has to pass in front of the slower vehicle, it is possible apply the following target point commutation strategy:

1. when the robot is behind the slow vehicle, the target is set as a point aligned with the back of other vehicle, but shifted at the left or at the right (if admissible) by a given safety distance, and moving at the same speed of the other vehicle;
2. once that the previous target is reached, the target is shifted forward of a given distance and is kept moving with a given velocity, slightly greater than that of the slower vehicle;
3. once the the previous target is reached, the target is set as a point in front of the other vehicle, and is kept moving with a given velocity, slightly greater than that of the slower vehicle.

At the end this maneuver, the robot has completed the pass and has safely avoided the moving obstacle with a smooth trajectory. Moreover, since nonlinear smoothing filter limits the derivative of linear acceleration, the trajectory would also be suitable for real autonomous vehicles, providing comfortable human transportation. An example of a trajectory achieving this behavior is shown in Fig. 6, in which the three different moving target points, as previously described, are denoted with green triangles. Notice that motion of the controlled robot and of the slower vehicle seem to overlap at the end of the passing, but of course the slower vehicle is actually behind the robot and the two do not collide.

![Fig. 6. Smooth trajectory (blue) to pass a slower robot (red) moving in the same direction](image)

5. IMPLEMENTATION AND EXPERIMENTS

The proposed nonlinear filter has been tested on a in-house built differential-drive platform, with custom electronics. The trajectory generation algorithm and the closed-loop controller described in previous sections have been implemented on a motion control card based on a dsPIC30F by Microchip Technology, which is a 16-bit fixed-point Digital Signal Controller (DSC) executing up to 30 MIPS.

The performance of the DSC have been optimized thanks to the use of fixed-point computations (with resolution augmented to 32-bit). Particular care has been given to proper scaling of the variables in the code, to avoid as much as possible the use of division operations. In fact, as is well-known by fixed-point DSP programmers, divisions by powers of two can be translated into left shift operations, which can be executed more efficiently (i.e. one instruction cycle vs. 17 instruction cycles on a dsPIC30F). As an example, the implementation of the SM control law for the nonlinear filter of Fig.1 requires divisions by $U$ (bound on second-order derivative) and $T$ (sampling-time) as denominator, so that setting these values as powers of two reduces dramatically the execution time. The full implementation of the trajectory generator of Fig.3 is computed by the DSC in less than 700 $\mu$s, which allowed to safely set the sampling frequency of the full control system at 256 Hz (i.e. $2^8$ Hz or $T = 3.90625$ ms). Finally, a global planner can interact with the trajectory generator by setting the sequence of cartesian via-points, together with related via-point commutation distance, through serial communication.

In its current form, the implementation of the trajectory tracking control loop exploits the feedback obtained from wheel odometry estimates and their numerical differentiation. Even if this estimate may be enhanced using inertial sensors, the control performance can be fairly evaluated on the basis of the norm of the tracking error w.r.t. odometry. This norm, measured during experiments on the
reference trajectory of Fig. 5, is always lower than 11.5 mm, with an average value of 3.9 mm. Fig. 7 shows the error measurements in the two distinct coordinates. These good tracking performances are better than those described by Oriolo et al. (2002) and Lau et al. (2009), which used robots with comparable dynamic performances and similar control methods, but much more powerful computational platforms.

Fig. 7. Tracking error in the cartesian space

In particular, Fig. 7 shows that even during initial transient, in which the robot accelerates from zero to maximum linear velocity, tracking is accurate thanks to the feedforward action, the compensation of robot dynamics by means of driving force $\tau_d$ and steering torque $\tau_s$ control inputs and the smoothness of generated trajectory.

Full source code of the firmware for the motion control board used in the experiments can be downloaded from http://sact-unife.googlecode.com.

6. CONCLUSION AND FUTURE WORK

The paper has described an approach to trajectory generation for mobile robots based on the theory of nonlinear smoothing filters. The trajectories obtained from the filter have continuous curvature and are inherently compatible with kinematic and dynamic constraints of a classical unicycle-like robot. Accurate trajectory tracking can be achieved with a control system based on I/O linearization with dynamic state feedback. A global path planner can interact with the trajectory generator by simply setting sequences of fixed via-points or reference points moving along non-smooth and obstacle-free paths.

In future works we aim to develop a complete navigation system including a global planner and the trajectory generator proposed in this paper. Moreover, we plan to extend this navigation system for a three-dimensional space and, for example, aerial vehicle applications (Bertozzi et al. (2009)). Finally, further experiments on embedded control boards will demonstrate the suitability of the approach for low-cost robotic applications.

REFERENCES


