Physical-consistent behavior embodied in B-spline curves for robot path planning

Davide Chiaravalli * **Federico Califano** * **Luigi Biagiotti** **
Daniele De Gregorio * **Claudio Melchiorri** *

* *DEI, Department of Electrical, Electronic and Information Engineering - University of Bologna (e-mail: {davide.chiaravalli2},{federico.califano2},{daniele.degregorio3}, {claudio.melchiorri}@unibo.it).*
** *DIEF, Department of Engineering "Enzo Ferrari"- University of Modena and Reggio Emilia (e-mail: luigi.biagiotti@unimore.it)*

Abstract: In this paper, a path planning algorithm for smooth obstacle avoidance of mobile robots is presented and discussed. The idea is based on B-splines, which are trajectories defined by a linear combination of basis functions weighted by constants called control points. Since the overall curve depends locally only on the position of a limited set of control points, B-splines are ideal tools for generating online reference trajectories able to react to local changes of the environment (e.g. mobile or spawning obstacles). The via-points of the reference B-spline are treated as dynamic agents with customized dynamic properties. These agents interact with each other and with the environment in such a way to generate the correct reference for the B-spline generator and thus for the robot. The high flexibility and the low computational burden of the proposed algorithm allow to develop applications in partially unknown or dynamically changing environments since a local modification of the trajectory does not require an entire re-computation. Experimental results performed with a KUKA Youboot mobile robot within a ROS based set-up are reported.

Keywords: Trajectory planning, Mobile robots, Splines, Smoothing filters.

1. INTRODUCTION

The motion problem, consisting in steering a robotic system from a given initial configuration to a desired final configuration while avoiding obstacles, has been coped in a number of different ways. However, it is worth noticing that two main philosophies exist: on the one side, the motion problem is considered as a planning problem, aiming at computing a collision-free path between two configurations, which is solved before the motion starts. On the other hand the motion problem is often solved according to a reactive paradigm in which, while the robot progresses towards the desired goal, the sequence of motion controls is modified in order to avoid the obstacles detected by the sensors. This approach is clearly performed at execution time. The former category includes sample-based planners Siciliano et al. (2010) or combinatorial roadmaps Yahja et al. (2000), Kucuk (2016) able to find the optimal geometric path, if any exists, guaranteeing the feasibility of the motion at the cost of a high computational burden due to online validation. In the latter group, artificial potential fields represent the most common approach since they allow to achieve a fast and reactive response to a dynamically changing environment Arkin (1989), Khatib (1986), Slack (1993); however, since the motion of the robot is determined by descending the gradient of the potential field generated by the target and the obstacles, it is possible that the robot stacks into a local minimum Koren and Borenstein (1991).

Attempts of mixing the two motion control philosophies are also present in the literature. In Sgorbissa and Zaccaria (2012) a reactive component of the algorithm can modify the pre-computed path online in order to take care of dynamic changes of the environment. Masone et al. (2012) introduces a telemanipulation-based technique that allows a human operator to interact with the preplanned trajectory by modifying online some geometric properties of a desired cyclic path. The proposed work exploits the same idea since it aims at finding a bridge between motion planning and reactive behaviour. The key point is represented by the use of spline curves in the so-called B-form, whose geometric elements, i.e. the control points, are defined as virtual agents able to interact with each other and with the environment, populated by obstacles modelled as potential fields, in a simple and efficient manner that preserves stability and avoids local minima stacking situations. Note that the initial position of the control points/agents can be determined offline with a standard planning algorithm, based for instance on optimization procedures for obstacle avoidance Sprunk et al. (2012), Kolter and Ng (2009), Stoican et al. (2016).

The paper is organized as follows. In Sec. 2 the properties of B-spline curves are explained. Then in Sec. 3 the multiagent framework and the hypothesis considered are illustrated. In Sec. 4 the algorithm is described and its advantages are analysed. In Sec. 5 the experimental set-up description and the results are reported.

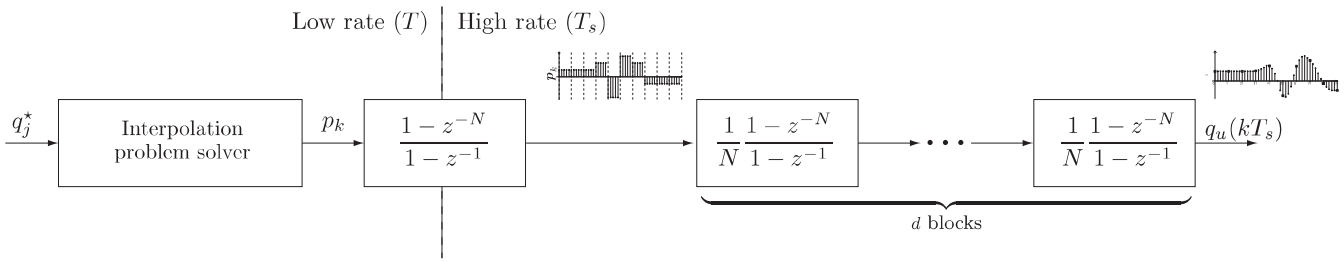


Fig. 1. Structure of the discrete-time filter for B-spline trajectories planning.

2. B-SPLINE CURVES

The basic idea of this paper is to modify the trajectory tracked by the robot in runtime by acting on the parameters defining it. To this end the use of spline trajectories in the so called B-form is the ideal solution. B-spline curve are defined as

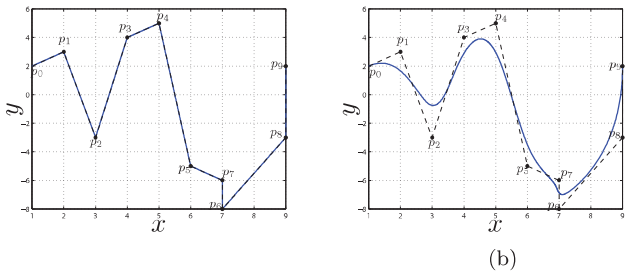
$$q(t) = \sum_{k=0}^{n-1} p_k \, B_k^d(t), \qquad t_0 \le t \le t_{n-1} \tag{1}$$

where the parameters p_k are control points that roughly define the geometrical shape of the trajectory, while $B_k^d(t)$ are the so-called B-spline basis functions of degree d, which determine how the curve is followed since they contain the temporal information. The smoothness of the curve depends on the order d of the basis functions $B_k^d(t)$ which are scalar functions of class C^{d-1} defined over a vector of time-instants $[t_0, t_1, \ldots, t_{n-2}, t_{n-1}]$, known as knots vector. A noteworthy property of basis functions is that $B_k^d(t)$ is zero everywhere except in the time interval $[t_k, t_{k+d+1}]$As a consequence, the k-th control point p_k influences the B-spline trajectory only in this temporal interval. This means that a variation on a point causes only a local modification of the whole curve, see Fig. 3. As already mentioned, a B-spline of order $d > 1$ does not cross the control points p_k and in order to define a trajectory passing though a given set of n points $q_i^\star$ it is necessary to impose some interpolation conditions at the desired time-instants t_i, i.e.

$$q(t_i) = q_i^\star. \tag{2}$$

These conditions lead to a linear system whose solution provides the value of the control points p_k, $k = 0, \ldots, n-1$. For more details refer to Biagiotti and Melchiorri (2008).

All the properties above mentioned remain valid if uniform B-splines are considered, i.e. B-splines characterized by an equally-spaced by T distribution of the time knots



(b)

Fig. 2. 2D B-spline curves: linear (a) and cubic (b).

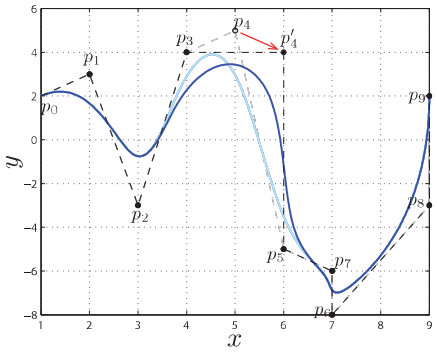


Fig. 3. Local modification of a B-spline curve due to the change of position of a control point.

$$q_u(t) = \sum_{k=0}^{n-1} p_k \, B^d(t - kT), \qquad 0 \le t \le (n-1)T. \tag{3}$$

In Biagiotti and Melchiorri (2010) it is shown that in this case the trajectory $q_u(t)$ can be efficiently generated by means of a chain of d dynamic filters defined as

$$M(s) = \frac{1 - e^{-sT}}{Ts} \tag{4}$$

and by a zero-order hold $H_0(s)$ fed by a train of impulses of amplitude p_k. If the B-spline function must be evaluated at discrete-time instants, it is possible to directly define a filter in the discrete-time domain by discretizing the continuous-time filters (4). In this way, the trajectory generator of Fig. 1 is obtained, where $N = T/T_s$ (see Biagiotti and Melchiorri (2010) for details). Obviously the scheme of Fig. 1 produces scalar B-splines. However, for generating vectorial B-splines, that is splines defined in a multi-dimensional space, it is sufficient to consider a chain of filters for each component of the control points, which in this case are vectors.

Note that the use of the digital filter for B-spline generation offers the following advantages (see Biagiotti and Melchiorri (2013)):

- a new value of the curve is provided at each sample time of the control loop;
- the continuity of the output curve is always guaranteed even if the control points position is modified;
- the computational burden is extremely low and the implementation process rather simple, since d FIR filters, whose input is kept constant to the value of the control points for a duration of T seconds, are sufficient for computing the trajectory.

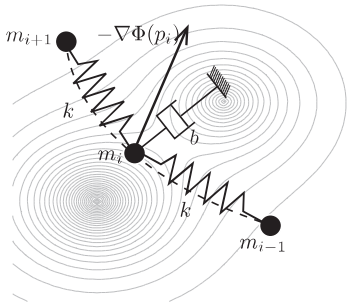


Fig. 4. A detail of all the forces acting on agent i.

3. MULTIAGENT FRAMEWORK AND ENVIRONMENT CHARACTERIZATION

The aim of the proposed algorithm is to control the motion of a mobile robot by providing a reference trajectory which is aware and reactive to the environment and its changes. The peculiarities of B-spline curves mentioned in section 2 make them a very efficient tool for achieving the required objective. These trajectories are in fact suitable set point signals for the control layer of the robot because of their smoothness (depending on the degree d of the spline) and their property of global curvature minimization. These aspects avoid automatically abrupt variations in the computed path and thus harsh torque profiles at a joint level. Since a position change of the control points causes only a local modification of the overall trajectory, each of them can be treated as a single dynamic agent interacting with the environment and the other agents, leading to a multiagent framework. The steady state position of the agents will univocally define the trajectory to be routed. In the following a 2-D environment (xy plane) within a planar motion contest is considered but extension to the 3-D case is straightforward.

3.1 Characterization of the Agents

An ordered set of n virtual agents representing the viapoints of the B-spline to be interpolated are initially defined. They are modelled as point-like masses with mass m_i, $i = 1, .., n$. Each agent i is recursively placed along the line starting at the beginning of the path and directed toward the target at a fixed distance δ_r from the previous agent. Agent i is connected to its neighbors (agents $i-1$ and $i+1$) through a virtual linear spring with stiffness k and initially in resting condition (with rest length δ_r). The virtual points are considered to be immersed in a viscous environment characterized by the viscosity coefficient b. Obstacles are modelled to generate a repulsive potential that will be described later in detail.
As shown in Fig.4, a dynamic multivariable system is thus introduced in which agent i is affected at any time by four different forces:

- an elastic force generated by the spring connecting it to the following agent $i+1$;
- an elastic force generated by the spring connecting it to the previous agent $i-1$;
- a viscous friction force proportional to its velocity;
- an external force provided by the gradient of the overall potential field generated by the obstacles.

All the agents dynamically react and modify their position in the environment according to the resulting external force in each time instant. The considered system is a passive mechanical system only composed by energy storing elements (masses and springs) and dissipative elements (dampers) and thus stability and convergence to steady state position values of the agents are assured. In particular, once the transients are expired, the action of the springs forces all the agents to be almost equally spaced with each other, guaranteeing the uniformity of robot motion provided by the B-spline generator that receive the agents' positions as input.

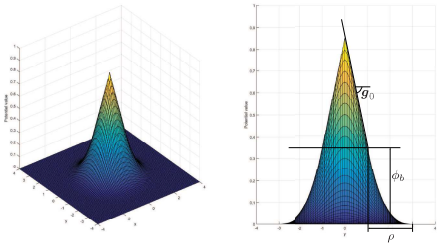


Fig. 5. Potential field described in (6).

3.2 Environment Description

In this study the obstacles characterizing the environment are assumed to be convex and separated from each other. This choice allows to focus on the analysis of the advantages of the proposed algorithm whereas more complex environments characterized by randomly shaped obstacles will be considered in future works. Each obstacle is modelled by a repulsive potential field. The resulting potential Φ in a generic point $p = (p_x, p_y)$ is given by the sum over the 2-D plane of the potential generated by the l obstacles present in the environment:

$$\Phi(p) = \sum_{k=0}^{l} \Phi_k(p) \tag{5}$$

If we assume for simplicity that the shape of obstacles is extended to a circle circumscribing the simplex, the value of the radial potential generated by obstacle k, with radius r centered in $o = (o_x, o_y)$, is given by

$$\Phi_k(p) = \begin{cases} g_0(|p-o| - r) + \phi_b, & \text{if } |p-o| \leq r \\ f(p), & \text{if } |p-o| \leq r+\rho \\ 0, & \text{if } |p-o| > r+\rho \end{cases} \tag{6}$$

$$f(p) = \phi_b + g_0(|p-o| - r) - \frac{3\phi_b + 2g_0\rho}{\rho^2}(|p-o| - r)^2 + \\ + \frac{2\phi_b + g_0\rho}{\rho^3}(|p-o| - r)^3 \tag{7}$$

where g_0 is the constant slope in the interior region of the potential, ϕ_b is the potential value at the obstacle's border and ρ represents the safety distance from the border, that is chosen greater than half of the robot's length. $f(p)$ has been chosen as a third degree polynomial function that smoothly joins the linear part of the potential (see Fig. 5). Note that these parameters are independent with respect to the considered obstacle. In this way each obstacle presents the same repulsive field regardless of its radius in the region adjacent to its border. The modulus of the gradient presents a constant value g_0 within the obstacle preventing the application of possibly very large forces in

case one of the virtual agents spawns near to the obstacle's center and gradually reduces to zero at a radial distance ρ from the border of the obstacle. Eventually each agent i has the following dynamic constraint

$$m_i\ddot{p}_i + b\dot{p}_i + F_e^i - F_e^{i+1} = -\nabla\Phi(p_i) \tag{8}$$

where

$$F_e^i = k(|p_i - p_{i-1}| - \delta_r)vers(p_i - p_{i-1})$$

with $p_i = (x_i, y_i)$ the current position of the agent, $\nabla\Phi(p_i)$ the gradient of the potential and $vers(p) = \frac{p}{|p|}$.

4. ALGORITHM DESCRIPTION

According to the environment characteristics and the task to be computed, two different versions of the algorithm have been considered. The initial study has been carried out in a fully known environment characterized by static obstacles. In this kind of scenario a *global planner*, evaluating the whole trajectory toward the final goal is defined. In case the environment is only partially known (e.g. through proprioceptive sensors equipped on a mobile robot) and the environment itself is dynamically changing, a *local planner*, producing runtime a trajectory with adaptive properties is considered.

4.1 Global planner

The global planner task consists in simply reaching a goal position $G = (G_x, G_y)$ starting from a start position $S = (S_x, S_y)$ in a completely know environment. The first step takes into account the initialization and the definition of the n virtual agents, which are equidistantly placed in an ordered way along the line connecting S to G. The number of agents is clearly provided by

$$n = \left[\frac{|G - S|}{\delta_r} - 1\right] \tag{9}$$

where the $[\,]$ operator returns the closest integer. Each agent is characterized by the same mass m. Agent 1 and n are in this way placed at a distance δ_r respectively from S and G, that can be seen as two additional static agents with infinite mass. Once the initialization procedure is terminated the agents start to react to the environment and the dynamic motion starts according to (8). When the agents converge to their steady state values, their positions are again interpreted as via-points and the corresponding control points are provided to the B-spline generator that dispenses the geometric trajectory that smoothly interpolates the agents as shown in Fig.6. It is important to choose $\delta_r < \min_{k\in O} r_k$ where O is the set of obstacles indexed by k and r_k is the radius of obstacle k. This is necessary to ensure that any pair of adjacent agents are not initially located at opposite sides of any obstacle, leading to a trajectory colliding with the obstacle itself. One of the advantages of this method is the overcoming of

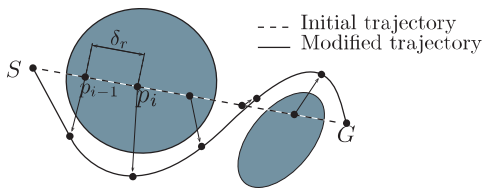


Fig. 6. Working principle of the global planner.

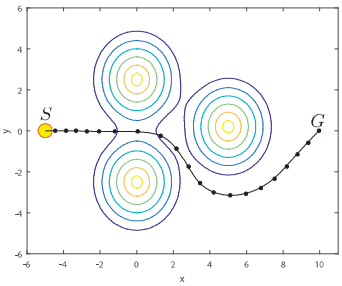


Fig. 7. The global planner avoids potential traps.

potential traps: even if it may happen that some agents will stack in a stationary point descending the potential, at the interpolation stage the resulting trajectory will smoothly join the start with the goal without taking care of the local minima (see Fig. 7)

4.2 Local adaptive planner

Let's consider now a dynamically changing environment and let's suppose that the robot has a large, possibly unknown, distance to cover. This is the case, for example, when the stating point S is known but we have information only of the heading direction in which the goal G lies. In this new scenario the previous global planner can not be considered a satisfactory solution for two main reasons:

- an uniform distribution of a great amount of agents between S and G would be both very expensive from a computational point of view and an unjustified expedient since eventually the robot travels along a trajectory which is locally defined only by the few agents that are close to it.
- an obstacle with variable position could end up stretching the trajectory indefinitely.

In these circumstances it is possible to considered an improved version of the algorithm, called the local adaptive planner. The main difference lies in the role of the agents. With respect to the global planner, where n virtual agents were placed equidistantly from S to G, now the agents will cover only a limited portion of space starting from the current robot position and will be updated as described later. In this perspective the agents gain a deeper meaning that can be abstracted as follows: since the first agents will define locally the trajectory to be computed they have to convoy the robot during his motion, while the last ones are aimed to accomplish an exploration task since they have to react responsively to the upcoming environment. In particular the last agent on the line, from now on called the "explorer", will always be considered the current target of the robot in the global planner algorithm sense. It is placed at a defined fraction of the line (depending on the number of agents n) connecting the starting point and the goal and the initialization stage is completed by placing the remaining agents, equally spaced by δ_r, between S and the explorer.
The masses of the agents are set as monotonically decreasing:

$$m_i > m_j, \quad \forall i < j, \quad i, j = 1, ..., n$$

In this way the first part of the resulting trajectory will present less fluctuations since the agents are subjected to slower dynamics and the smoothness of the motion of the robot is not compromised. The last agents gain opposite

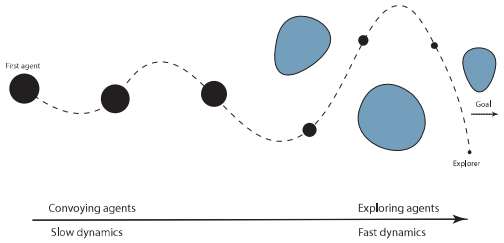


Fig. 8. A graphical interpretation of the roles of the agents.

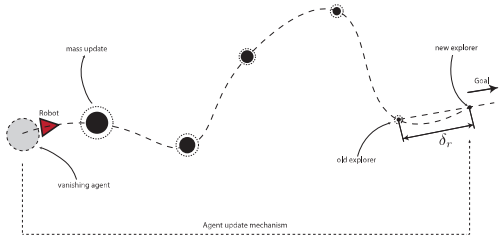


Fig. 9. A graphical visualization of the update mechanism.

characteristics and in particular the explorer, which is clearly set to have the lowest mass value, makes the last part of the trajectory very reactive to the presence of obstacles and very responsive to the changes in the environment. Even if the last part of the B-spline presents fast variations (e.g. caused by a spawned obstacle) the current motion of the robot is not affected by them while the adaptive part of the trajectory is taking care of the said variations. In Fig. 8 a graphical interpretation of the dynamic characterization of the agents in the local adaptive planner is presented. The number of agents represents a tradeoff between the computational burden of the algorithm and the adaptation capability of the resulting trajectory: as n increases, the amount of computation required increases, while at the same time the conceptual independence between the quality of the agents (from convoying ones to exploring ones) as described above is accentuated. In order to reach the goal G a discrete periodic reconfiguration of the position of the agents is needed: once the first agent is reached it vanishes and a new agent, that will be the new explorer, is placed toward G at a distance δ_r from the old one or toward the next trajectory point in the case the initialization is performed using a sample-base planner. All the masses of the agents are scaled according to the new configuration (see Fig. 9). The reconfiguration stops when the distance of the robot from the goal is small enough to use the global planner to end the trajectory. On the contrary of what happens in the global planner case, the motion of the robot does not wait for the steady state configuration achievement of all the agents but only (with a prescribed level of tolerance) for the assessment of the convoying ones. During the reconfigurations the robot keeps moving relying on the slow dynamics of the agents close to it provided by a sufficiently high number of agents and by the scaled mass values.
The velocity can be controlled by the online modification of the mutual distances between agents by acting on the rest length of the virtual springs δ_r. In particular by enlarging the rest length value the velocity increases and viceversa.

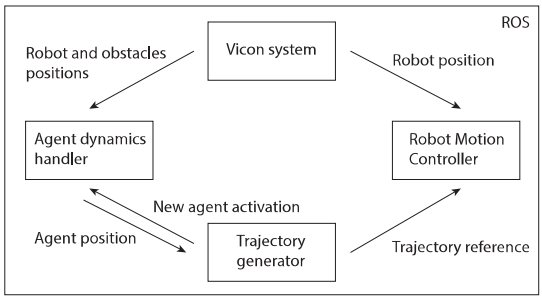


Fig. 10. Overall conceptual scheme of the trajectory planner in ROS.

5. EXPERIMENTS

The experimental tests have been carried out in a known environment using a Kuka Youbot that has to avoid obstacles to reach a goal in a 2D navigation plane. All the positions of the obstacles and the robot have been continuously tracked using a "Vicon" network of cameras. The overall software infrastructure has been built using the Robot Operating System (ROS). The node network is shown in 10. The structure can be seen as a dual rate system with an inner sampling period of T_s (corresponding to the frequency of the nodes) and an outer one of T (set to be an integer multiple of T_s) that provides the frequency of the update mechanism.

5.1 Experimental Results

The arena inside which the mobile robot is moving, defined by a 3×3 m^2 area, is characterized by four obstacles of different shapes and dimensions. In Fig.11 five snapshots of the total motion of the robot during the experiment are presented. The real image is augmented with the path reconstruction in Rviz. In this manner it is possible to see how the agents react to the potential fields generated by the obstacles, visualized as a dark cylinder within the shape of the obstacle itself. The light region on the adjacent area up to a distance ρ from the border of the obstacle represents the polynomial part of the potential as described in Sec. 3. According to the youBot dimensions and the environment's characteristics the parameters shown in the Table 1

In the sequence of figures it is possible to appreciate how the exploring agents interact with the potential field generated by the obstacles and modify their position accordingly, producing a smooth trajectory that circumnavigates the area around the obstacles. The convoying agents providing the instantaneous references for the motion are unaffected by the modification of the trajectory caused by the explorers. As result an overall trajectory characterized by a constant velocity along the defined path is generated. In the tests performed over fifty different scenarios, characterized by obstacles randomly generated but whose potential fields are not interacting with each other, the

Table 1.

Variable	Value	Unit	Variable	Value	Unit
δ_r	0.1	[m]	ϕ_b	0.35	[Nm]
k	1	[N/m]	g_0	-1	[N]
n	15		ρ	0.25	[m]

Fig. 11. Snapshots of robot motion and Rviz interface.

success rate has been of 100%. Moreover the generation of the initial trajectory by means of a sample-based planner has allowed to extend the result to environments in which the interaction between the potential fields is considered. The analysis of the generated trajectory showed that, by taking into account the parameters values defined in 1, the agents become stable in 1.25 seconds, on average. This result confirms the reactiveness of the algorithm with respect to dynamical changes in the environment.

6. CONCLUSIONS

In this paper a novel path planning algorithm based on modification of B-spline curves is proposed. The via-points of the B-spline curve have been treated as reactive agents and have been modelled in a physical consistent way in order to generate collision free paths. The experimental results have shown a successful behaviour of the proposed algorithm which presents low computational burden, may provide adaptability to changing environments and can lead to many different studies. In particular very promising applications could be considered in the fields of human safety in human-robot interaction scenarios and of industrial warehousing handled by multi-robot networks. The proposed approach is amenable to be used in cooperation with other algorithms. For example it could act after a sample based planner has already computed an initial guess of the trajectory offline and adjust the precomputed path smoothly and responsively with respect to variations in the environment. In the future the extension of the local planner for unknown dynamical environment reconstructed by on-board sensors will be carried out with particular attention on non-convex and large shaped obstacles. Furthermore telemanipulation-oriented works considering the control of the exploring agents by means of an haptic device, relying on the global assessment of the underlying trajectory, will be explored.

REFERENCES

Arkin, R. (1989). Motor schema based mobile robot navigation. *The International Journal of Robotics Research*, 8, 92–112.

Biagiotti, L. and Melchiorri, C. (2008). *Trajectory Planning for Automatic Machines and Robots*. Springer, Heidelberg, Germany.

Biagiotti, L. and Melchiorri, C. (2013). Online trajectory planning and filtering for robotic applications via b-spline smoothing filters. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 5668–5673.

Biagiotti, L. and Melchiorri, C. (2010). B-spline based filters for multi-point trajectories planning. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 3065–3070.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5, 90–98.

Kolter, J.Z. and Ng, A.Y. (2009). Task-space trajectories via cubic spline optimization. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 1675–1682.

Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation (ICRA), 1991 IEEE International Conference on*, 1398–1404.

Kucuk, S. (2016). Maximal dexterous trajectory generation and cubic spline optimization for fully planar parallel manipulators. *Computers & Electrical Engineering*.

Masone, C., Franchi, A., Bulthoff, H., and Giordano, P.R. (2012). Interactive planning of persistent trajectories for human-assisted navigation of mobile robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal.

Sgorbissa, A. and Zaccaria, R. (2012). Planning and obstacle avoidance in mobile robotics. *Robotics and Autonomous Systems*, 60, 628–638.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010). *Robotics: modelling, planning and control*. Springer Science & Business Media.

Slack, M. (1993). Navigation templates: mediating qualitative guidance and quantitative control in mobile robots. *IEEE Transaction on Systems, Man and Cybernetics*, 23, 452–466.

Sprunk, C., Lau, B., and Burgard, W. (2012). Improved non-linear spline fitting for teaching trajectories to mobile robots. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2068–2073.

Stoican, F., Ivanusca, V., Prodan, I., and Popescu, D. (2016). Obstacle avoidance via b-spline parametrization of flat trajectories. In *24th Mediterranean Conference on Control and Automation (MED)*. Athens, Greece.

Yahja, A., Singh, S., and Stentz, A. (2000). An efficient on-line path planner for outdoor mobile robots. *Robotics and Autonomous systems*, 32(2), 129–143.