Low-level flexible planning for mobile manipulators: A distributed perception approach

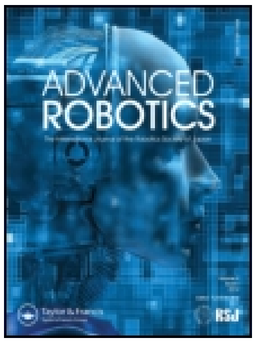
Article *in* Advanced Robotics · October 2014

DOI: 10.1080/01691864.2014.946446

2 authors:

Pietro Falco
ABB
28 PUBLICATIONS 165 CITATIONS
SEE PROFILE

Ciro Natale
Università degli Studi della Campania "Luigi Vanvitelli"
142 PUBLICATIONS 1,910 CITATIONS
SEE PROFILE



Advanced Robotics

ISSN: 0169-1864 (Print) 1568-5535 (Online) Journal homepage: http://www.tandfonline.com/loi/tadr20

Low-level flexible planning for mobile manipulators: a distributed perception approach

Pietro Falco & Ciro Natale

Advanced Robotics, 2014
Vol. 28, No. 21, 1431–1444, http://dx.doi.org/10.1080/01691864.2014.946446

FULL PAPER

Low-level flexible planning for mobile manipulators: a distributed perception approach

Pietro Falco and Ciro Natale*

Dipartimento di Ingegneria Industriale e dell'Informazione, Seconda Università degli Studi di Napoli, 81031 Aversa, Italy

(*Received 19 September 2013; revised 21 February 2014 and 10 June 2014; accepted 9 July 2014*)

The paper proposes a method to improve flexibility of the motion planning process for mobile manipulators. The approach is based on the exploitation of perception data available only from simple proximity sensors distributed on the robot. Such data are used to correct pre-planned motions to cope with uncertainties and dynamic changes of the scene at execution time. The algorithm computes robot motion commands aimed at fulfilling the mission by combining two tasks at the same time, i.e. following the planned end-effector path and avoiding obstacles in the environment, by exploiting robot redundancy as well as handling priorities among tasks. Moreover, a technique to smoothly switch between the tasks is presented. To show the effectiveness of the method, four experimental case studies have been presented consisting in a place task executed by a mobile manipulator in an increasingly cluttered scene.

Keywords: mobile manipulation; flexible motion planning; perception; obstacle avoidance; multiple tasks

1. Introduction

Nowadays, one of the main objectives of roboticists is to take robots out of the factories and let them enter into unstructured environments. In those environments, planning the robot motion completely off-line may likely bring to a failure of the assigned task, since a high degree of uncertainty is present and the environment can dynamically change. To tackle this issue, research in advanced robotics is going towards the concept of flexible planning. A planning method can be considered flexible when it is able to quickly modify a robot motion computed off-line when unexpected changes happen in the environment, such as the presence of unexpected obstacles, humans moving in the workspace, or uncertainties on the position of expected obstacles. Another useful feature of a flexible planning is the possibility to fulfill several tasks with given priorities. To make a planning method flexible, perception capabilities play a key role. Various sensorial data can be taken into account to give the robot system a knowledge of the external world, e.g. visual, proximity, force information, etc. If based on visual perception, the robot acquires global information about the surrounding environment that can be used for motion planning and for obstacle avoidance of mostly slowly moving objects, proximity measurements seem more suitable for both early detection and collision avoidance with faster moving objects and thus for fast reaction to unexpected situations when high-level replanning is not possible because the motion of the robot has to be quickly adapted to the possibly dangerous situation. From the technological point of view, various solutions exist, e.g. sensors based on laser range finders [1] or on optoelectronic [2–4] or on electromagnetic [5] technologies.

In the literature, motion generation for robotic systems has been traditionally addressed by the robotic research community from two different points of view. On one side, the motion planning community has developed global approaches [6] requiring a priori knowledge of the environment and intensive computational time, aimed at finding global and optimal solutions in the configuration space of the robot corresponding to a desired task in a workspace populated by physical objects representing obstacles to be avoided. To the other side, the control community has concentrated on local reactive approaches [7], at computing joint control signals based on sensor information, for an instantaneous, possibly constrained, motion in the task/operational space, but they are subject to local minima resulting into suboptimal motion.

The category of global methods, which do not suffer from local minima, includes the probabilistic road map, a multiple-query stochastic method [8], and the rapidly exploring random tree [9], a single query stochastic method, where the free configuration space is explored only in the interesting parts. However, all these methods assume a static environment, as they are executed off-line on the basis of the a priori information on the environment.

The second category, still in the framework of potential-field methods, comprises the elastic strips [10] and, more recently, the elastic roadmaps [11], which constitute an advance towards obstacle avoidance strategies and reactive planning methods. Differently from other types of motion

*Corresponding author. Email: ciro.natale@unina2.it planning algorithm, in this framework there is no need to explore the entire configuration space, instead it maps proximity information from the environment into the configuration space, using the kinematics of the manipulator. However, the potential fields are applied to a discretized representation of the entire trajectory and not only to a particular configuration of the robot, which, again, increases the computational complexity and still is a local method.

The idea proposed in this work, belonging to the local methods category, is the concept of a low-level flexible planning system able to quickly react to environment changes or uncertainties on the a priori knowledge of the environment by locally adjusting trajectories that can be either planned by the cited global methods or they can even be very simple motion primitives in case of simple scenes. The proposed algorithm, that could be applied to a generic robotic system, is here specialized to a mobile manipulator and it fully exploits all its degrees of freedom (DOF) achieving full coordination between the arm and the mobile base. The approach is based on the Null-Space-Behavioral (NSB) framework [12] with the introduction of task priority. In order to handle the unexpected situations, two tasks with priorities have been defined. The first task consists in computing the robot configurations such that the off-line planned trajectory of the end effector is followed in absence of obstacles. The second task tries, through the NSB approach, to avoid unexpected obstacles and, if possible, to follow the planned end-effector trajectory. An approach to effectively combine the tasks and exchange their priority in a smooth way is also proposed. It is similar to the one presented in [13,14], with the difference that the combination is not in the task space but in the configuration space. This, besides a computational simplification, allows a better separation between planning and control levels, in the sense that the planner is not requested to know the local modifications computed at execution time, but it just needs to know if the task has been accomplished or not, e.g. to command a new task or to re-plan the same task. This feedback from the low-level reactive control layer to the high-level planning layer is a first attempt towards a hybrid integrated approach for a tighter connection between planning and control.

The proposed method uses poor proximity perception and does not require a sensorized environment, i.e. all the required sensors are aboard of the robot. In detail, a discrete number of proximity sensors are installed on a mobile manipulator, not only located on the mobile base but also in some points distributed along the kinematic chain of the robotic arm so as to build a sort of 'safety volume' around the whole robot. This will allow the robot to detect and react to unexpected changes in the scene configuration. Such distributed approach, in fact, defines a safety volume with a geometry which changes with the robot configuration, without requiring neither accurate knowledge of the link geometry except for the position of the sensors with respect to the robot link frames, nor the adoption of geometric primitives that can result in too conservative reactions, like spherical surfaces used in [15]. To the best of our knowledge, it is the first time that a mobile manipulator is equipped with such proximity sensors distributed both on the mobile base and the arm.

The implementation of the idea described above is carried out by introducing the concept of the 'total pseudo-energy' of a set of virtual springs distributed along the robot links. Each spring is attached on one side to each sensor and on the other side to the point of the obstacle closest to the sensor location. In this way, differently from the classical virtual potential fields that are assigned to each obstacle in the working space [7], a sort of 'protective' buffer is placed around each robot link. Another important difference with the classical potential field method is that it is straightforward to define multiples behaviors with different priorities. Such a feature is particularly useful for cluttered and dynamic environments in which complex tasks have to be executed, not only navigation but also manipulation of objects. A further characteristic of the proposed method is that it is robot-centered and not environment-centered like the potential field, in the sense that the pseudo-energy does not depend on the number of obstacles and on the complexity of the environment but only on the distribution of sensors on the robot. Therefore, the complexity of the algorithm does not increase with the complexity of the environment, in contrast to Khatib's approach, where for each obstacle, a contribution to the potential has to be computed. On the other hand, a similarity with the potential field approach is its local validity and possible occurrence of local minima, limitation common to any algorithm exploiting only local perception data.

The proposed control algorithm aims at minimizing the total pseudo-energy by changing the robot configuration trying to fulfill the planned task at the same time by exploiting the redundancy of the base-arm system. The parameters of the algorithm have been chosen in the light of the findings presented in [16] to ensure stability when implemented in discrete-time. A similar approach has been adopted by

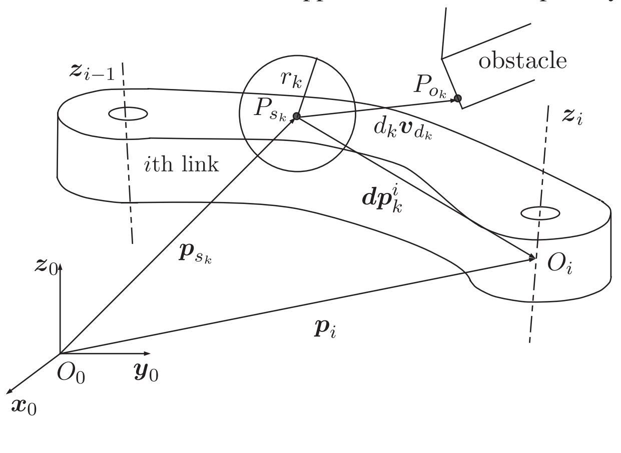


Figure 1. The kth sensor mounted on the ith link and an obstacle.

Table 1. DH table of the KUKA youBot.

Joint	a [mm]	α [rad]	d [mm]	θ [rad]
1	0	$\pi/2$	q_1	0
2	0	$\pi/2$	q_2	$\pi/2$
3	167	0	0	q_3
4	33	$\pi/2$	147	q_4
5	155	0	0	q_5
6	135	0	0	q_6
7	0	$\pi/2$	0	q_7
8	0	0	217.5	q_8

the skeleton algorithm [17], but it can handle only self-collisions since it exploits only proprioceptive sensing.

In order to show the effectiveness of the proposed algorithm in a cluttered and dynamic environment, four experimental case studies have been carried out on a Kuka youBot. The experiments prove how the robot, only with a very simple off-line plan, can fulfill the assigned task in a significantly different scene or in presence of unexpected changes in the scene.

2. The approach

As said, to handle collisions with objects not initially considered by the high-level planner, the robot is considered as 'protected' by N springs with a certain rest length, each one attached to a proximity sensor. When an obstacle 'touches' one or more springs (in the sense that it comes close enough to the corresponding sensors), the total elastic energy associated to these springs increases. To accomplish the task without hitting any object, the robot path in the configuration space is locally corrected so as to minimize the total energy, i.e. the sum of the elastic energies of all the springs touched by the obstacle. When possible, the energy is minimized by keeping the robot end effector along the desired trajectory.

In order to formalize the approach, assume that the kth sensor, located in the point P_{s_k}, measures the distance d_k between the nearest point on the obstacle P_{o_k} and the sensor itself as well as the direction of the minimum distance expressed by the unit vector $\boldsymbol{v}_{d_k}$. The method can be applied to a generic kinematic open chain, therefore the mobile base is assumed to be holonomic, as it is for the youBot [18] used in the experiments presented in this work. Assuming to adopt the Denavit–Hartenberg (DH) convention and that all vectors expressed in frame 0 (the first fixed DH-frame of the chain) have no superscript, with reference to Figure 1, let

- $\boldsymbol{q} = [q_1\ q_2\ \cdots\ q_n]^{\mathrm{T}} \in \mathbb{R}^n$ be the vector of configuration variables;
- $\boldsymbol{e}_4 = [0\ 0\ 0\ 1]^{\mathrm{T}}$;
- $\tilde{\boldsymbol{p}}_{i-1} = \mathbf{A}_1^0(q_1)\cdots\mathbf{A}_{i-1}^{i-2}(q_{i-1})\boldsymbol{e}_4$ be the position vector (homogeneous coordinates) of the origin of the DH-frame fixed to the link i, being $\mathbf{A}_i^{i-1}(q_i)$ the homogeneous transformation matrix expressing the pose of frame i with respect to frame $i-1$;
- $\boldsymbol{z}_{i-1} = \mathbf{R}_1^0(q_1)\cdots\mathbf{R}_{i-1}^{i-2}(q_{i-1})\boldsymbol{z}_0$ be the z axis (along joint i) of the frame fixed to link $i-1$, being $\mathbf{R}_i^{i-1}(q_i)$ the rotation matrix expressing the orientation of frame i with respect to frame $i-1$;
- $\boldsymbol{p}_{o_k}$ be the position vector of the obstacle point at minimum distance from the kth sensor; and
- $\boldsymbol{p}_{s_k}(\boldsymbol{q}) = \boldsymbol{p}_i(\boldsymbol{q}) - \mathbf{R}_i^0(\boldsymbol{q})\boldsymbol{dp}_k^i$ be the position vector of the kth sensor point, being $\boldsymbol{dp}_k^i$ the (constant) position of the kth sensor (assumed mounted on link i) with respect to frame i fixed to link i, being $\mathbf{R}_i^0(q)$ the rotation matrix expressing the orientation of frame i with respect to frame 0.

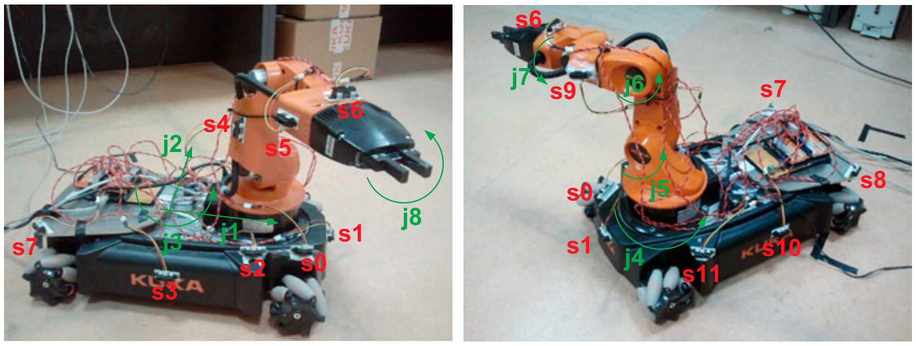


Figure 2. Kuka youBot with the proximity sensors installed and labeled in red; robot joints are labeled in green.

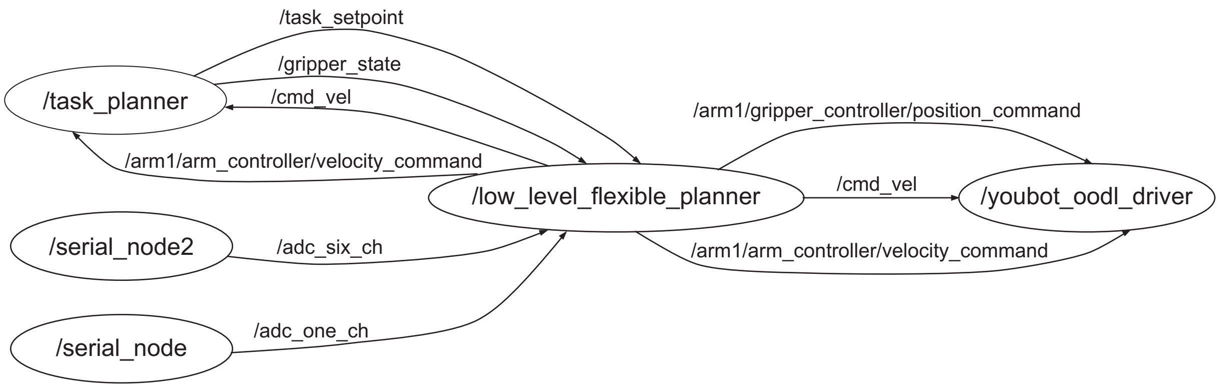

/task_setpoint
/gripper_state
/cmd_vel
/task_planner
/arm1/arm_controller/velocity_command
/arm1/gripper_controller/position_command
/low_level_flexible_planner
/cmd_vel
/youbot_oodl_driver
/serial_node2
/adc_six_ch
/arm1/arm_controller/velocity_command
/serial_node
/adc_one_ch

Figure 3. Graph of the ROS nodes.

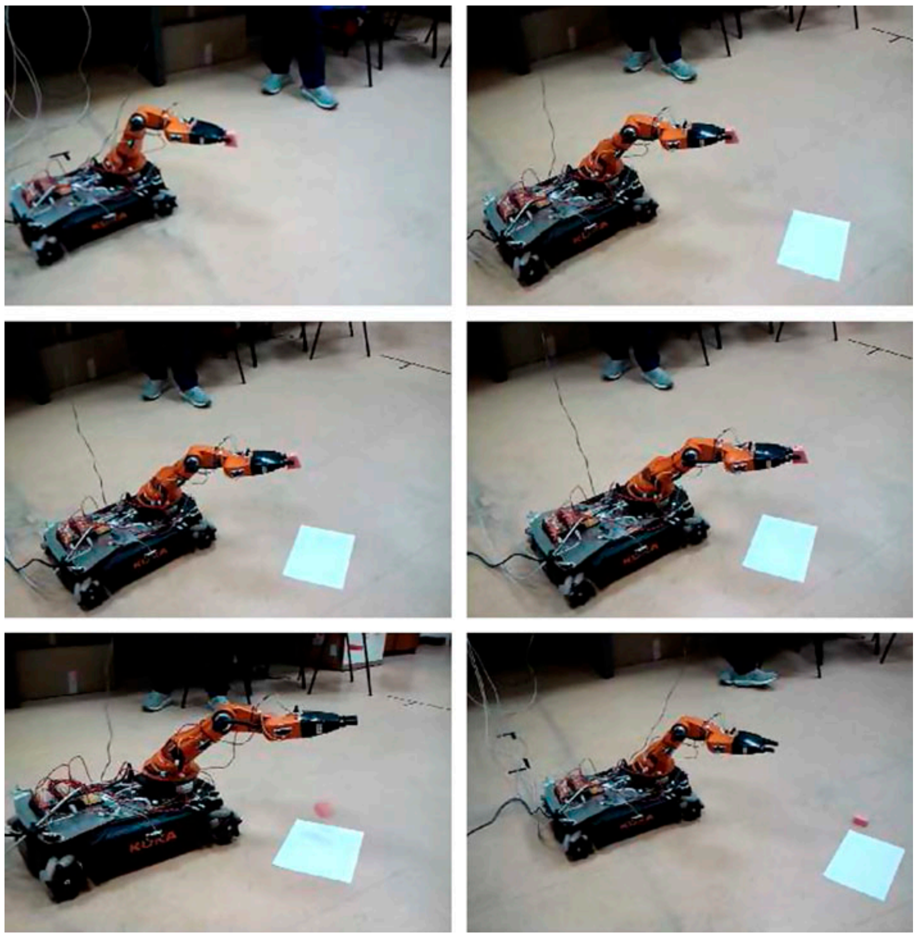


Figure 4. Snapshot sequence relative to the place task without obstacles.

Now, the pseudo-energy of the kth spring is defined as the following continuous function

$$\epsilon_k(\boldsymbol{q}) = \begin{cases} \frac{1}{2}(d_k - r_k)^2, & \text{if } d_k \le r_k \\ 0, & \text{otherwise} \end{cases}, \tag{1}$$

where $d_k(\boldsymbol{q}) = \left\| \boldsymbol{p}_{o_k} - \boldsymbol{p}_{s_k}(\boldsymbol{q}) \right\|$ is the information provided by the kth sensor, together with the direction $\boldsymbol{v}_{d_k}(\boldsymbol{q}) = (\boldsymbol{p}_{o_k} - \boldsymbol{p}_{s_k}(\boldsymbol{q}))/\left\| \boldsymbol{p}_{o_k} - \boldsymbol{p}_{s_k}(\boldsymbol{q}) \right\|$.

Note how the value of the pseudo-energy depends on both the distance between the obstacle and the robot and on the value r_k, defined by the user and interpreted as the rest length of the spring. The total energy of all the springs is simply

$$\sigma(\boldsymbol{q}) = \sum_{k=1}^{N} \epsilon_k(\boldsymbol{q}). \tag{2}$$

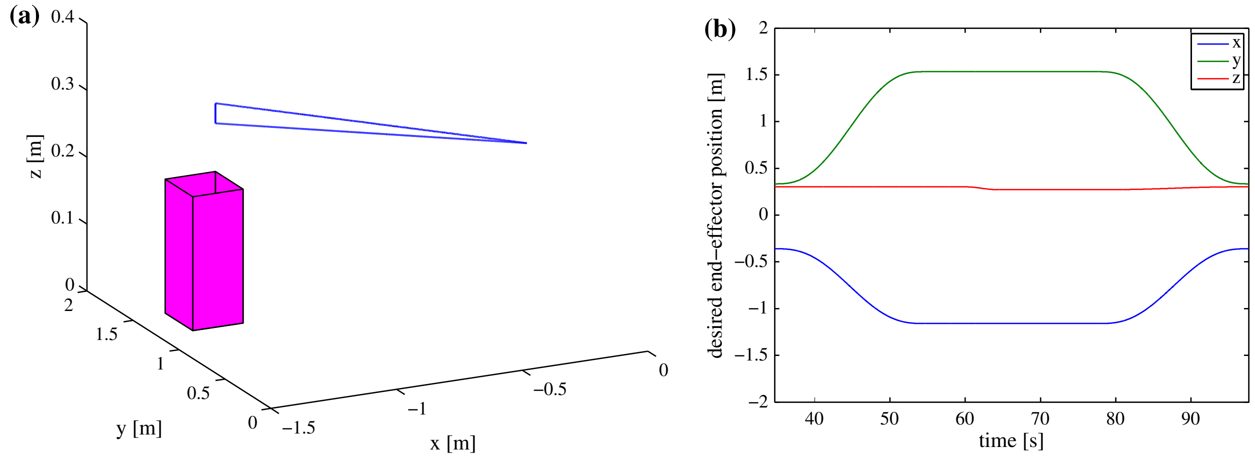


Figure 5. Planned motion of the robot end effector. (a) 3D path and (b) trajectory.

The objective of the algorithm is to follow the given reference trajectory $\boldsymbol{x}_d(t)$ with the robot end effector avoiding collision between any part of the robot with any other object in the environment, i.e. any other part of the robot and any obstacle fixed in unknown locations or subject to unknown motions.

To tackle the problem, a NSB approach is adopted with two tasks. The first one consists in finding a trajectory in the configuration space so that the end-effector trajectory is followed, therefore the first task function is simply the direct kinematics function, i.e.

$$\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q}), \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^r$ is the vector containing the task space variables. The second bahavior consists in finding a trajectory in the configuration space so that the total pseudo-energy is zero, i.e. $\sigma_d = 0$.

The robot control law used to accomplish both these tasks is presented as follows: first, the velocity commands needed to accomplish each task separately are derived, and then the method proposed to combine these commands ensuring continuity of the commanded velocity is presented. According to the NSB approach, each of the two tasks above can be executed by actuating the corresponding velocity in the configuration space, i.e.

$$\dot{\boldsymbol{q}}_g = \mathbf{J}_g^{\dagger}(\boldsymbol{q})(\dot{\boldsymbol{x}}_d + \gamma_g(\boldsymbol{x}_d - \boldsymbol{k}(\boldsymbol{q}))) \tag{4}$$

$$\dot{\boldsymbol{q}}_o = \mathbf{J}_o^{\dagger}(\boldsymbol{q})(\dot{\sigma}_d + \gamma_o(\sigma_d - \sigma(\boldsymbol{q}))) \tag{5}$$

where $\mathbf{J}_g(\boldsymbol{q}) = \partial \boldsymbol{k}(\boldsymbol{q})/\partial \boldsymbol{q}$ is the Jacobian of the robot, $\mathbf{J}_o(\boldsymbol{q}) = \partial\sigma(\boldsymbol{q})/\partial\boldsymbol{q}$ is the gradient of the energy function, and the symbol $\mathbf{X}^{\dagger}$ denotes the Moore–Penrose pseudo-inverse of the matrix $\mathbf{X}$; γ_g and γ_o are gains, which have an influence on the transient behavior of the robot and on the convergence of the closed-loop inverse kinematics (CLIK) algorithms. For a thorough discussion on the selection of these gains when the above algorithms are implemented in discrete time, the reader is referred to [16]. Note that for the task Jacobian $\mathbf{J}_g$, instead of the standard Moore–Penrose pseudo-inverse, the weighted pseudo-inverse $\mathbf{J}_g^{\dagger}$ can be computed so that motion of the arm DOF can be privileged with respect to base motion that can help to save power consumption. It is $\mathbf{J}_g^{\dagger} = \mathbf{W}\mathbf{J}_g^{\mathrm{T}}(\mathbf{J}_g\mathbf{W}\mathbf{J}_g^{\mathrm{T}})^{-1}$, with $\mathbf{W}$ usually selected as a positive definite diagonal matrix with higher entries in the places corresponding to the DOF with a desired higher speed.

The definition of the energy function in (2) and (1) allows the computation of the gradient $\mathbf{J}_o$ in closed form as:

$$\mathbf{J}_o(\boldsymbol{q}) = \sum_{k=1}^{N} \frac{\partial \epsilon_k(\boldsymbol{q})}{\partial \boldsymbol{q}}, \tag{6}$$

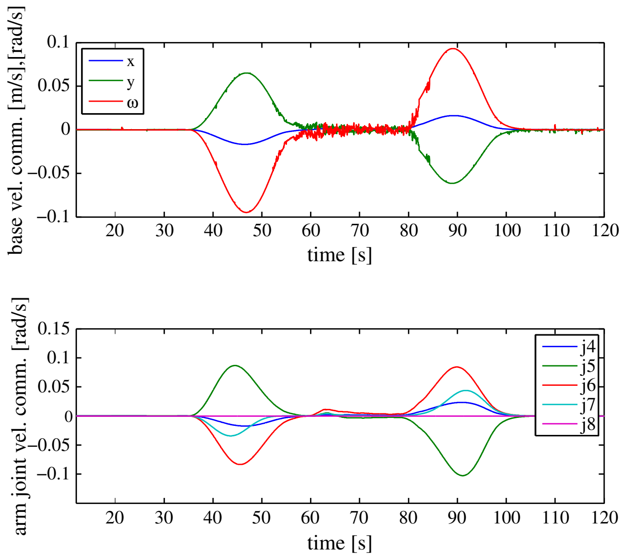


Figure 6. Velocity commands for the base and the arm relative to the place task without obstacles.

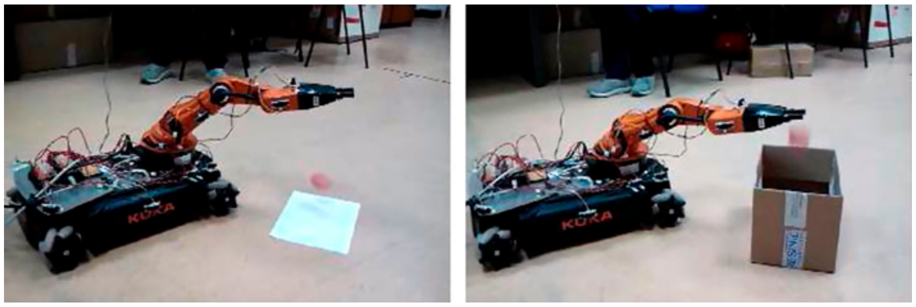
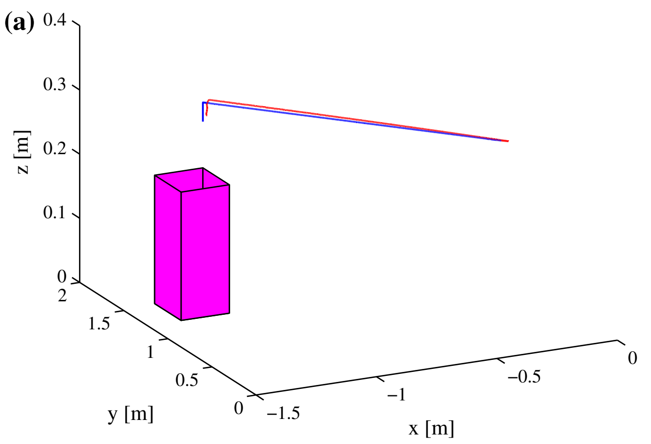


Figure 7. Comparison between arm configuration with and without the box in the first case study.



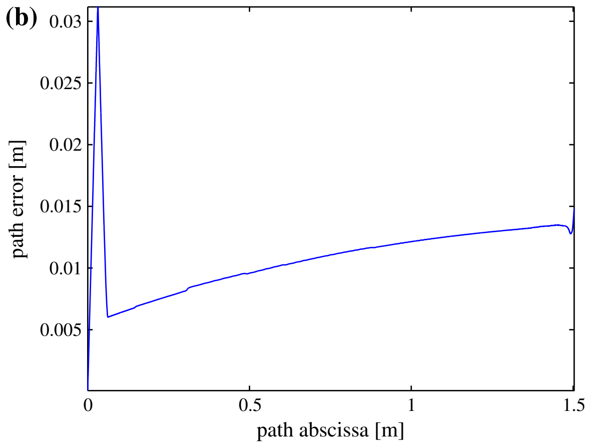


Figure 8. (a) Planned (blue) and executed (red) end-effector path, and (b) Distance between planned and executed path: task without obstacles.

where

$$\frac{\partial \epsilon_k(\boldsymbol{q})}{\partial \boldsymbol{q}} = \begin{cases} -(d_k - r_k)\boldsymbol{v}_{d_k}^{\mathrm{T}} \frac{\partial \boldsymbol{p}_{s_k}(\boldsymbol{q})}{\partial \boldsymbol{q}}, & \text{if } d_k \le r_k \\ 0, \text{ otherwise} \end{cases} \quad (7)$$

A key contribution of this work is the method to combine the two tasks. This is accomplished by the following convex combination

$$\dot{\boldsymbol{q}} = (1-\lambda(d))\dot{\boldsymbol{q}}_g + \lambda(d)\Big(\dot{\boldsymbol{q}}_o + (\mathbf{I} - \mathbf{J}_o^{\dagger}(\boldsymbol{q})\mathbf{J}_o(\boldsymbol{q}))\dot{\boldsymbol{q}}_g\Big), \quad (8)$$

where d measures the distance between the closest obstacle and the sensor closest to this obstacle, i.e.

$$d = \min_{k=1,\dots,N} d_k \quad (9)$$

and $\lambda(d)$ is a weighting function. In classical approaches, $\lambda(d)$ is chosen as a simple switching function, i.e.

$$\lambda(d) = \begin{cases} 1 \; d \le f \\ 0 \; d > f \end{cases}, \quad (10)$$

where f is an activation threshold to be selected. It is well known that this choice can generate a sort of chattering of the computed velocity [12]. To avoid this undesired effect that can generate vibrations of the mechanical structure of the robot, $\lambda(d)$ can be chosen as a smooth sigmoidal function, i.e.

$$\lambda(d) = \frac{1}{\pi} \arctan\left(-K(d-f)\right) + 1/2, \quad (11)$$

or a piece-wise linear function, i.e.

$$\lambda(d) = \begin{cases} 1 & \text{if } \; d \le f - \Delta/2 \\ 0 & \text{if } \; d \ge f + \Delta/2 \\ 1/2 - (d-f)/\Delta & \text{otherwise} \end{cases}. \quad (12)$$

Note that in this way, a sort of 'gray zone' is defined where a priority among the tasks is not 'crisply' defined. Outside of this zone (namely for d large or small enough), the priority is established by the classical null-space projection method, i.e. with $\lambda = 1$, the obstacle avoidance task has a higher priority with respect to the path following task that means the robot tries to accomplish the latter task only by exploiting the redundant DOF. The design parameters $K > 0$ and Δ vary the slope of the function and thus the width of this gray zone.

3. Experimental setup

To test the proposed algorithm, a Kuka youBot mobile manipulator is considered, whose DH table is reported in Table 1. Figure 2 shows a picture of the youBot with the proximity sensors distributed all over the base and the arm. Twelve proximity sensors have been mounted on the robot, in detail 8 Sharp GP2Y0A21YK have been mounted on the mobile base since they have a range of about 80 cm, while 4 Sharp GP2D120XJ00F with 30 cm range have been mounted on the arm. The position of each sensor, expressed in the frame fixed to the link the sensor is attached to, is

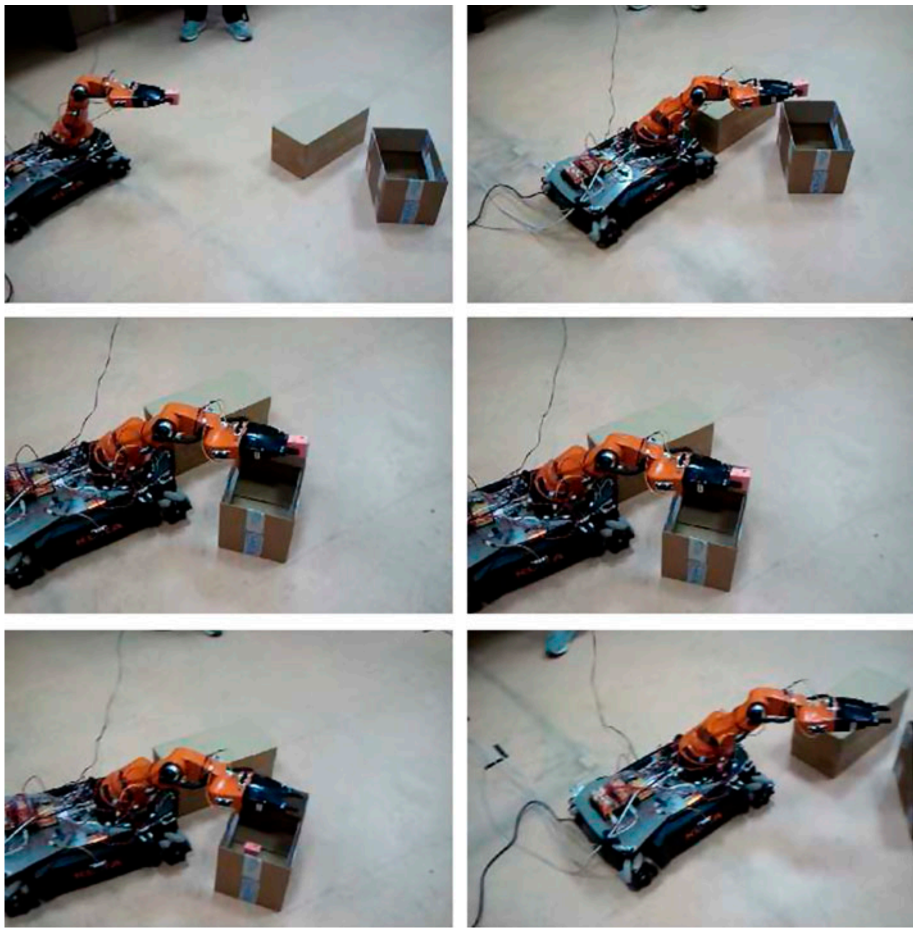


Figure 9. Snapshot sequence relative to the place task without enabling the flexible planning algorithm.

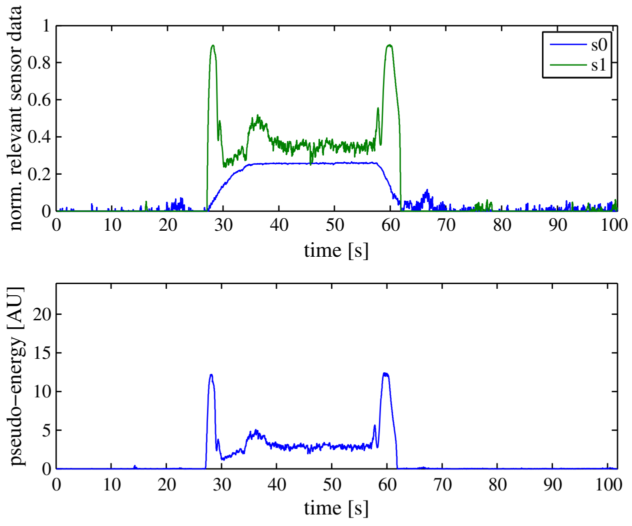


Figure 10. Normalized output, expressed in arbitrary units (AU) of the relevant proximity sensors (top) and pseudo-energy (bottom) in presence of an obstacle and without enabling the flexible planning algorithm.

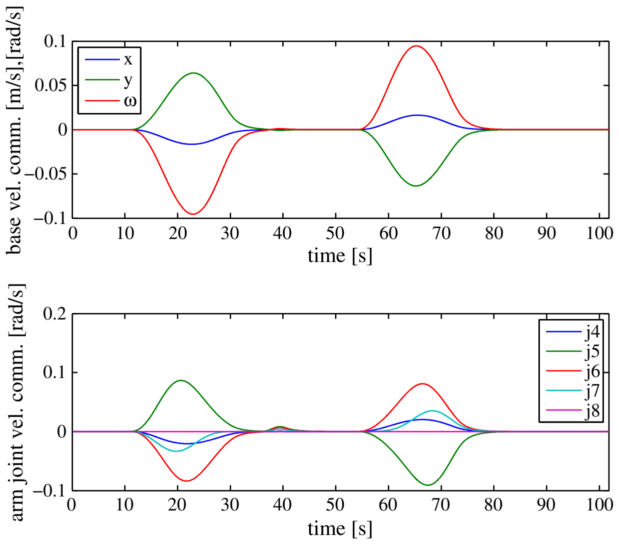


Figure 11. Base and arm command velocities during the task in presence of an obstacle and without enabling the flexible planning algorithm.

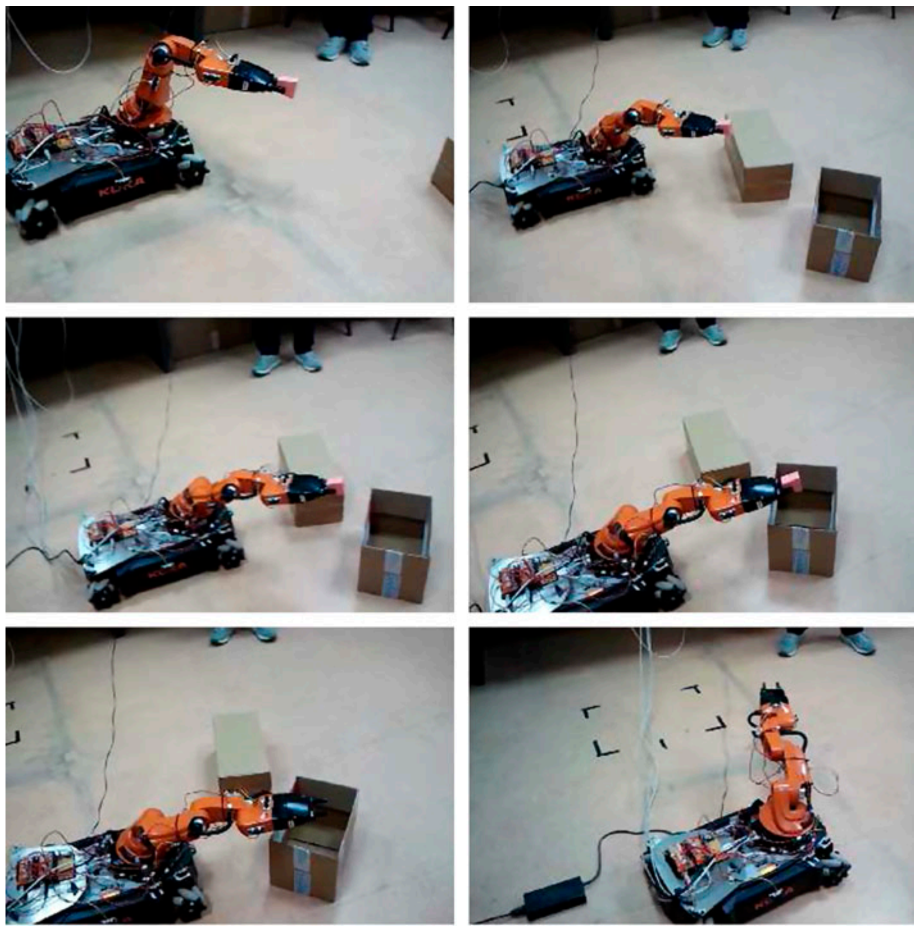


Figure 12. Snapshot sequence relative to the place task in the presence of an obstacle and with the flexible planning algorithm enabled.

reported in the matrix

$$\begin{aligned}\mathbf{P} &= \left[\, \boldsymbol{p}_{s1}^{l_1} \;\dots\; \boldsymbol{p}_{s12}^{l_{12}} \,\right] \\ &= \begin{pmatrix} 16 & 23 & 10 & -4 & -7 & 0 & 4 & -29 & -29 & 0 & -8 & 12 \\ -11 & 0 & -16 & -16 & -2 & 6 & 0 & -13 & 13 & 0 & 16 & 20 \\ 0 & 0 & 4 & 0 & 8 & 4 & 7 & 5 & 5 & -8 & 0 & 4 \end{pmatrix}\end{aligned} \tag{13}$$

where the i-th entry l_i of the vector $\boldsymbol{l} = [\, l_1 \;\dots\; l_{12} \,]^{\mathrm{T}}$ is the index of the link on which the i-th sensor is mounted and the 12 sensors are mounted such that $\boldsymbol{l} = [3\ 3\ 3\ 3\ 5\ 7\ 7\ 3\ 3\ 6\ 3\ 3]^{\mathrm{T}}$ (note that the mobile base is link 3, while the arm links go from 4 to 8). Similarly, the unit vector of each sensor optical axis expressed in the same frame is reported in the matrix

$$\begin{aligned}\mathbf{V} &= \left[\, \boldsymbol{v}_{d1}^{l_1} \;\dots\; \boldsymbol{v}_{d12}^{l_{12}} \,\right] \\ &= \begin{pmatrix} \frac{\sqrt{2}}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 1 & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 0 & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{3}}{2} & -1 & 0 & 0.98 & 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 & 0.2 & 0 & 0 & 0 & -1 & 0 & 0 \end{pmatrix}\end{aligned} \tag{14}$$

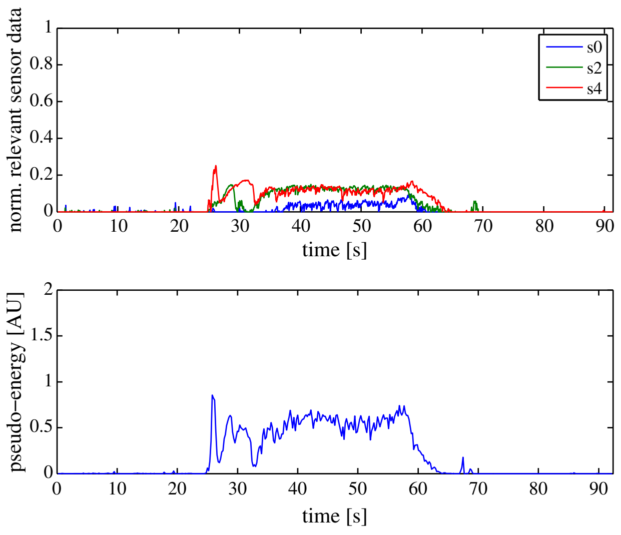


Figure 13. Normalized output of the relevant proximity sensors (top) and pseudo-energy (bottom) during the place task with obstacle.

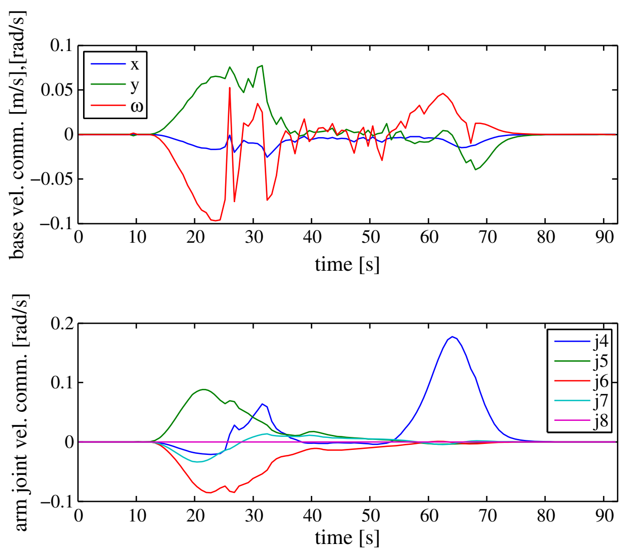


Figure 14. Base and arm command velocities during the place task in presence of an obstacle.

Note that the positions and directions of the sensors have been selected to cover the widest volume around the robot. Moreover, the sensors on the arm were located such that no sensor detects any robot link as an obstacle in the initial configuration selected to carry out the tasks described in the next section. A more systematic way to optimally select the locations of such sensors on a fixed-base industrial robot has been recently presented in [19].

The sensors have been interfaced with the robot control PC through a serial communication channel by connecting the sensor analog outputs to the A/D channels of two microcontroller boards ARDUINO Mega 2560. On each microcontroller, a firmware runs that sends, via a serial channel, sensor data to a ROS node on the control PC. This node then publishes ROS messages containing the sensor data. To the best of our knowledge, no similar robot setup or study with all the features of the proposed method, e.g. multi-task switching and distributed perception, exists to which a direct comparison can be actually performed.

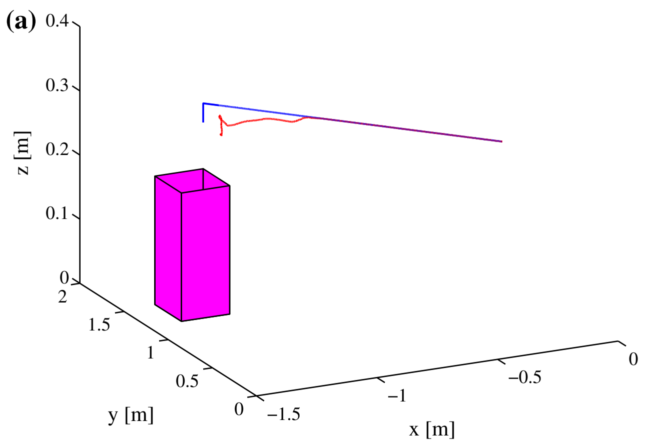
4. The software architecture

The experiments presented in Section 5 have been performed by exploiting the ROS programming framework. The software architecture is sketched in Figure 3 as a ROS graph and consists of different ROS nodes:

- *serial node*. It handles the communication with the first ARDUINO board. The messages sent by the node are measurements from the proximity sensors at a rate of 309 Hz.
- *serial node 2*. It handles the communication with the second ARDUINO board.
- *task planner*. This node generates the desired trajectories in the task space and the desired position of the gripper. This node publishes two kinds of messages: the desired gripper state (open or closed) as an aperiodic message and the planned end-effector position at a rate of 100 Hz.
- *low-level flexible planner*. It is the core of the proposed approach, since it implements the algorithm described in Section 2. As inputs, this node reads the proximity sensor measurements and the planned end-effector position. As outputs, it publishes messages with arm and base velocity commands at a rate of 100 Hz.
- *youbot-oddl*. This node is the ROS wrapper of the youBot driver. It is in charge of actuating the velocity commands for the base and the arm.

5. Experimental results

In order to show the performance of the developed algorithm, a task and four case studies have been selected. The



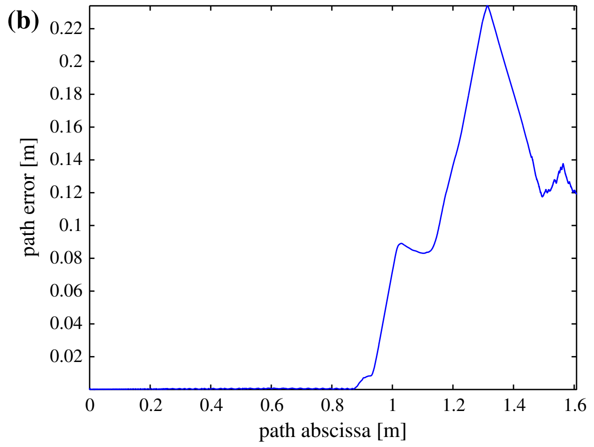


Figure 15. (a) Planned (blue) and executed (red) end-effector path and (b) Distance between planned and executed path: task with obstacle.

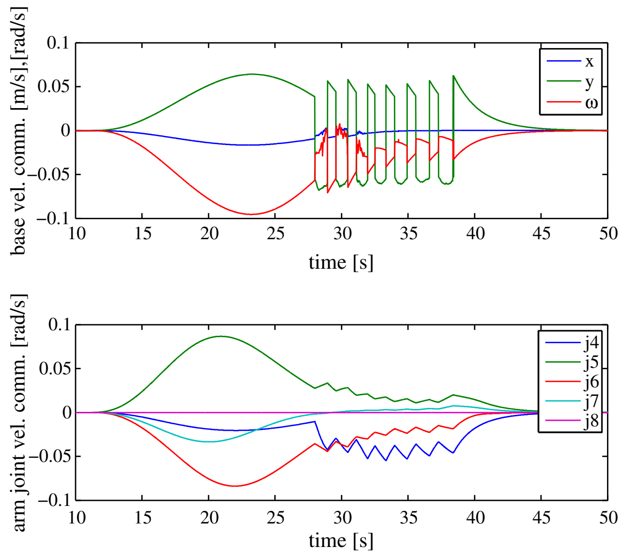


Figure 16. Base and arm command velocities during the place task with the crisp switching function.

task consists of placing an object into a box and it is composed by the following steps:

- the robot opens the gripper,
- a human gives an object to the robot,
- the robot closes the gripper,
- the robot brings the object into the box, and
- the robot end effector moves back to the initial position.

All the experiments have been executed with the following values of the parameters: $T = 10\,\text{ms}$, $\gamma_g = 0.5/T$, $\gamma_o = 0.1/T$, $f = r_k\,\forall k$, *and* $\mathbf{W} = \text{diag}\{0.08, 0.08, 0.1, 1, 1, 1, 1, 1\}$. Note that T is the sampling time and it affects algorithm stability. The CLIK gains have thus been selected according to the findings in [16].

The first case study, shown in Figure 4, consists of the task execution in the nominal scene configuration, that is with the workspace completely empty. It is important to stress, in fact, that the off-line planning does not consider the presence of the box in which the robot has to put the object, which can be an obstacle itself, thus, considering the $30 \times 30\,\text{cm}$ size of the box aperture, the place task has been considered accomplished if the final end-effector position is within the target area, i.e. 15 cm around the center of the box. The off-line planned path and trajectory are depicted in Figure 5(a) and (b), respectively. Figure 6 shows the velocity commands which the flexible planner sends to the robot base and to the robot arm.

In this case, it is evident that the commanded trajectories are simply the velocities in the configuration space such that the end effector follows the off-line planned trajectories in the Cartesian space. The task has then been executed in presence of the box (without any other obstacle), the flexible planner computes a correction of the off-line planned trajectory that brings the arm in a more stretched configuration (see Figure 7). This allows the base to stay at a safe distance from the box and, at the same time, it allows the end effector to bring the object in the target area. To better assess the effects of the flexible planning algorithm, the executed end-effector path has been qualitatively compared with the planned path as reported in Figure 8(a) and quantitatively in terms of the distance between the two paths as function of the arc length of the executed path as reported in Figure 8(b). Specifically, this distance, called path error, has been computed as

$$e(s) = \min_{\hat{s}} \|\boldsymbol{x}_d(\hat{s}) - \boldsymbol{x}(s)\|, \qquad (15)$$

where s and $\hat{s}$ are the arc lengths of the executed and planned paths, respectively. It is evident that the error is very low and due only to the tracking error of the low-level velocity control loop of the robot. Note that the paths are reported only from the starting point to the goal point omitting the backward motion.

In the second case study (Figures 9–11), an unexpected obstacle is placed between the box and the robot, but the flexible planning module is not enabled, that is the robot has to execute it without any modification. In more detail, the off-line planned path and trajectory are the same as in the previous case study (see Figure 5(a) and (b)), while in Figure 10, the outputs of the most relevant sensors and the pseudo-energy are plotted. It is interesting to note that in the range 30–60 s, the normalized outputs of two sensors get close to 1 (the maximum value) and the pseudo-energy dramatically increases. This indicates that the robot has collided with the obstacle as it is evident from the intermediate snapshots of Figure 9. As it can be seen in Figure 11, the velocity commands are practically identical to those computed in the absence of obstacles, except for the absence of high-frequency noise. This can be easily explained since in this case study, the proximity sensors are disabled, thus the flexible planning algorithm computes no correction to the velocity commands aimed at minimizing the pseudo-energy.

In the third case study, shown in Figure 12, the obstacle is still present but the flexible planning algorithm is enabled. Note how in Figure 13, both the pseudo-energy and the sensor outputs assume significantly lower values than in the previous scenario (that means the obstacle is at a greater distance). This is due to the trajectory corrections, computed by the proposed algorithm, applied through the velocity commands shown in Figure 14. From the snapshot sequence reported in Figure 12, it is evident that no collision occurs. The last snapshot shows the robot configuration at the end of the task, that is with the end effector in the same position as at the task start. Interestingly enough, this configuration is totally different from the initial one, as the

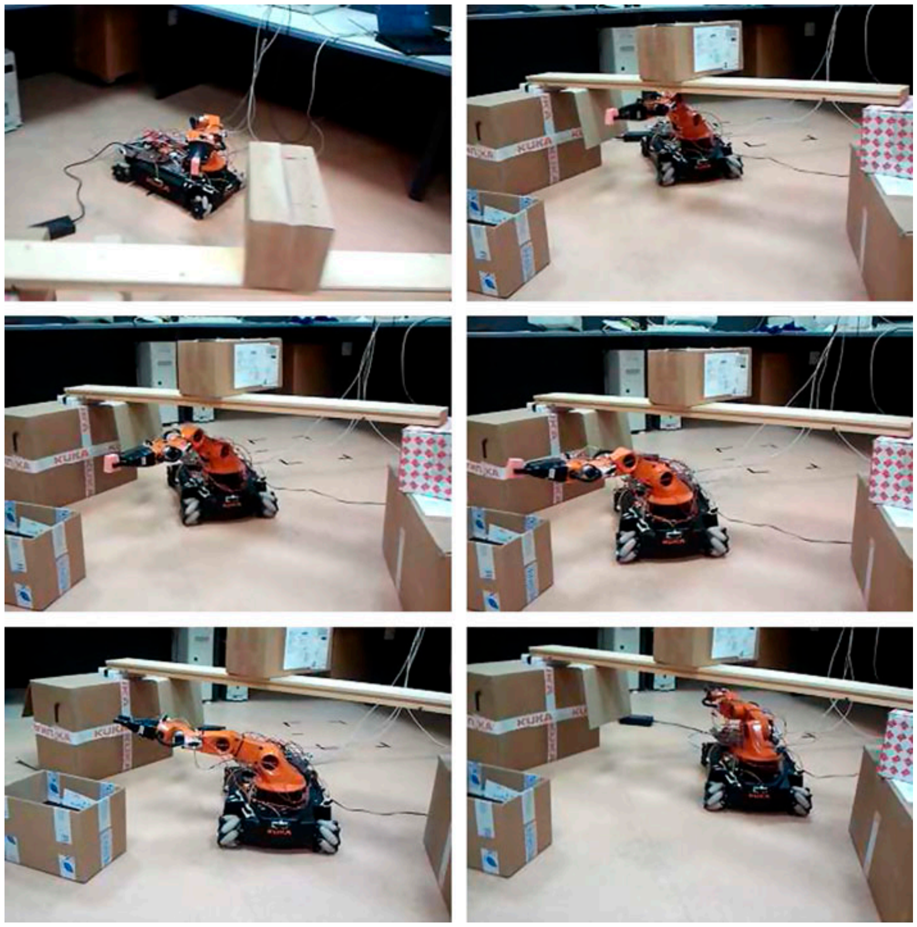


Figure 17. Snapshot sequence relative to the place task in a cluttered scene.

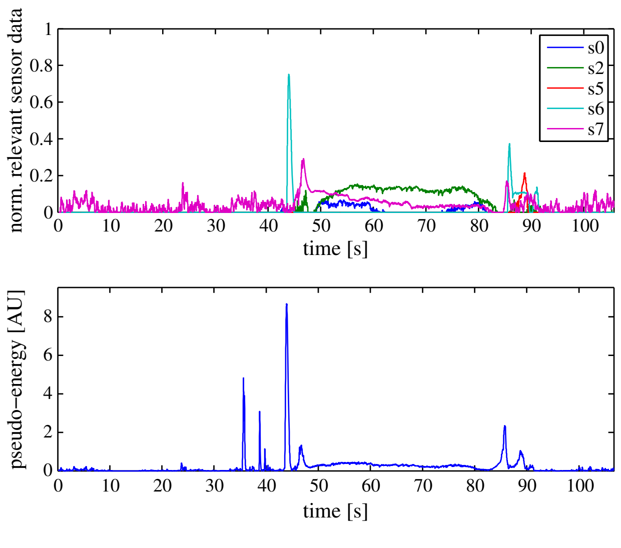


Figure 18. Normalized output of the relevant proximity sensors (top) and pseudo-energy (bottom) during the place task executed in the cluttered environment depicted in Figure 17.

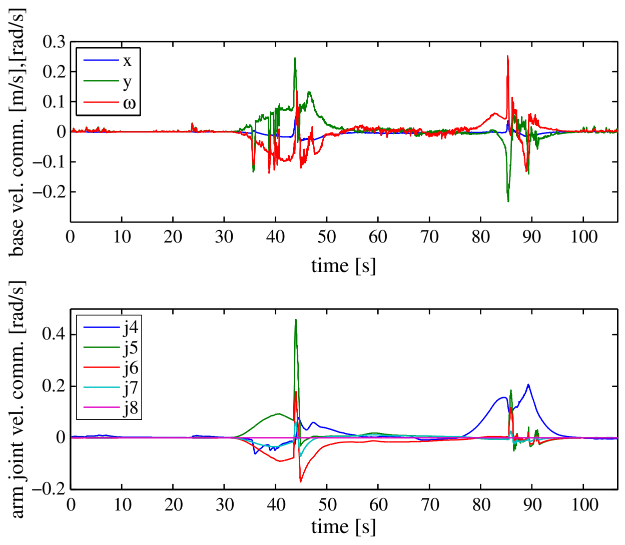


Figure 19. Base and arm command velocities during the place task in presence of a cluttered environment.

redundancy resolution method base on the Jacobian pseudo-inverse is not cyclic. Again, the planned and executed paths are compared in the same way of the first case study as reported in Figure 15(a) and (b). In this case, due to the

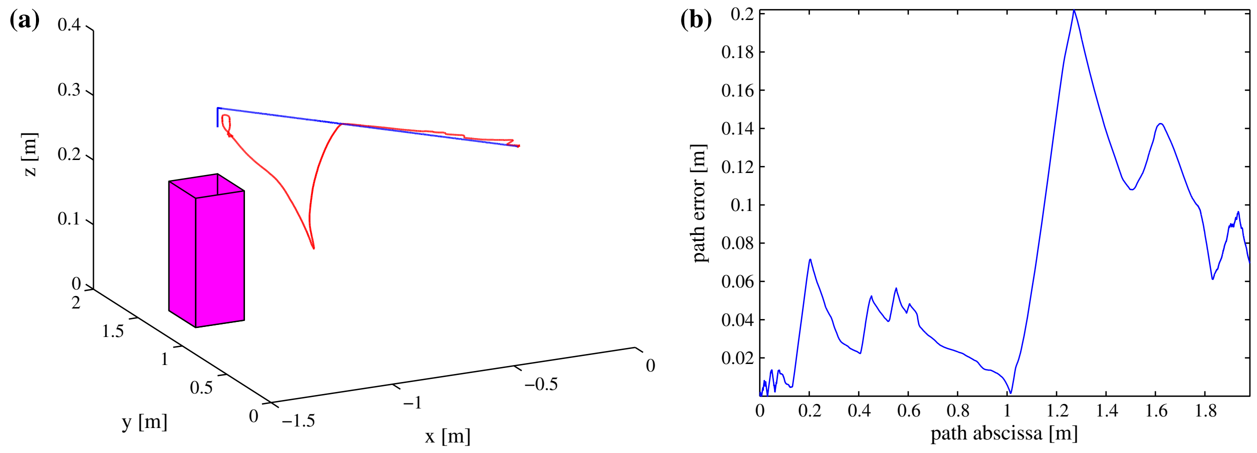


Figure 20. (a) Planned (blue) and executed (red) end-effector path and (b) Distance between planned and executed path: task in a cluttered environment.

obstacle in the proximity of the planned path, a path error up to 20 cm arises since the obstacle avoidance task has a greater priority with respect to the end-effector path tracking (see Equation (8) with $\lambda \to 1$, i.e. near an obstacle). This means that the redundant DOF are not enough to accomplish both tasks. At the end of the path, the error decreases below the target area dimension (15 cm), thus the task is successfully accomplished. However, note that the final error is due not only to the tracking error of the low-level control but also to the presence of the obstacle close to the target area and the robot is enforced to stay at a safe distance (note how the pseudo-energy in Figure 13 does not vanish in the time interval corresponding to the place phase). To show how the regularized switching function used to exchange priority among tasks is useful to avoid chattering of the velocity commands, the same task has been carried out using the switching function in (10). The results reported in Figure 16 confirm that a discontinuous switching function produces a chattering behavior of the robot.

As a last case study, the task has been executed with the robot acting in a more cluttered scene, which contains complex obstacles that obstruct the planned path not only from lateral directions but also from above. Figure 17 shows the usual snapshot sequence of the task execution. In Figure 18, it can be seen that there are more significant sensors than in the other cases. This is due to the presence of more obstacles in the workspace. Note how the algorithm is able to compute corrections to the planned motion in every direction so that no obstacle is hit. In particular, owing to the proximity sensors on the arm, it gets down to keep a safe distance from the horizontal beam. This is evident in Figure 18-(top) by considering the peak observed by sensor $s6$ (mounted on the arm) at about 44–46 s and by considering the energy peak in the same time interval in Figure 18-(bottom). Moreover, Figure 19 shows how the pseudo-energy peak mentioned above yields an arm velocity peak which gets the arm down. Finally, concerning the evaluation of the path error, it is evident from Figure 20(a) how the end effector has to get down to pass below the horizontal obstacle. Note also that the final error in Figure 20(b) is lower than in the previous case, since there is no obstacle very close to the target area and the error is mainly caused by the tracking error of the low-level velocity controller.

By summarizing, the presented experiments show that, using only a very simple high-level, off-line plan, the proposed algorithm allows the robot to successfully execute the planned task even in presence of unexpected obstacles and changing of the scene considered in the planning phase. Powerful off-line planning tools, e.g. Kineo CAM [20] or probabilistic methods cited in the introduction, can be invoked every time the low-level flexible planner gets stuck in a local minimum.

6. Conclusions

The paper reports both theoretical and experimental results of a new algorithm for flexible planning of mobile manipulation tasks based on a distributed perception approach. The idea is aimed at preventing the use of complex perception data such as video or depth maps of the scene, where the robot has to execute the task. Only simple proximity information is exploited to facilitate planning and execution of a mobile manipulation task in a cluttered environment. The flexibility of the approach relies on the ability of the algorithm to modify in real-time a predefined task by adapting it to the actual scene configuration on the basis of proximity information provided by simple sensors attached to both the base and the arm of the mobile manipulator, so as to create a sort of protective buffer around the whole robot. Based on the virtual deformations of such a buffer caused by proximity of objects to the robot, the planned trajectories of the mobile manipulator are changed in real-time to avoid unforeseen obstacles, obstacles with uncertain positions or even moving in the scene. The strategy does not require any complex 3D model of the scene nor of the robot, thus it has a low computational burden that allows it to easily work in real-time on simple processors. The low complexity can be ascribed to the poor perception data and the adoption of a multi-task Jacobian-based algorithm to compute the configuration variables of the robot. The limitation of the method is basically its local validity, i.e. the robot is not guaranteed to fulfill the complete task in any scene configuration, as it is shown in the attached video. Future research work will thus be devoted to overcome this limitation by a more strict interaction with higher level, global planning modules, which are complementary to the proposed algorithm. In fact, the algorithm can currently be integrated within a more complex architecture, comprising a high-level planner that can provide intermediate goals to be reached by the proposed low-level flexible planner. The method is experimentally tested on a mobile manipulator with eight degrees of freedom executing a place task in a complex scene shown in the attached video.

Funding

This work was supported partially by the European Commission under Grant Agreement n. 287513 (SAPHARI project) and partially by MIUR under the research project ROCOCO' (PRIN 2009).

Supplemental data

Supplemental data for this article can be accessed at http://dx.doi.org/10.1080/01691864.2014.946446.

Notes on contributors

Pietro Falco was born in Naples on May 3rd, 1984. He received the 'Laurea Triennale' and the 'Laurea Magistrale' both cum Laude in 2006 and 2008, respectively. In 2012, he received the Research Doctorate degree in Electronic Engineering and currently he holds a post-doc position at Second University of Naples. From December 2010 to July 2011 Pietro Falco was a visiting scholar at Karlsruher Institute of Technology (KIT) in Karlsruhe, Germany in collaboration with Dillmann's research group. His research interests include mobile manipulation, sensor fusion, multimodal perception, and observation of human manipulation. He has been principally working within DEXMART, ECHORD, and SAPHARI EU projects.

Ciro Natale received the Laurea degree and the Research Doctorate degree in Electronic Engineering from the University of Naples in 1995 and 2000, respectively. From 2000 to 2004, he has been a research associate at the Second University of Naples, where he currently holds the position of an associate professor of Robotics. From November 1998 to April 1999, he was a visiting scholar at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. His research interests on robotics include modeling and control of industrial manipulators, force and visual control and cooperative robots, and sensor fusion and multimodal perception. In the aeronautics application sector, his research activities are focused on modeling and control of flexible structures, active noise and vibration control, and identification and control of smart actuators. He has published more than 90 journals and conference papers as well as authored and co-authored monographs on robotics and control of flexible structures published by Springer. He currently serves as an associate editor of the IEEE Transactions on Control Systems Technology.

References

[1] Thrun S, Burgard W, Fox D. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In: IEEE International Conference on Robotics and Automation; San Francisco, CA; 2000 Apr. p. 321–328.

[2] Cirillo A, Cirillo P, De Maria G, Natale C, Pirozzi S. A proximity/contact-force sensor for Human safety in industrial robot environment. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics; Wollongong, Australia; 2013 Jul 9–12. p. 1272–1277.

[3] Automatic Control Laboratory at the Second University of Napoli Experiment of integrated contact/proximity sensing. Demonstration video. [Internet] Available from: www.dii.unina2.it/acl/video/ABB_video.wmv.

[4] Terada K, Suzuki Y, Hasegawa H, Sone S, Ming A, Ishikawa M, Shimojo M. Development of omni-directional and fast-responsive net-structure proximity sensor. In: IEEE/RSJ International Conference on Intelligent Robots and Systems; San Francisco, CA; 2011 Sept 25–30. p. 1954–1961.

[5] Conte G, Scaradozzi D, Rosettani M. An application of E-field sensors in industrial robotics. In: The 17th Mediterranean Conference on Control and Automation; Thessaloniki, Greece; 2009 Jun 24–26. p. 760–765.

[6] Latombe JC. Robot motion planning. Boston (MA): Kluwer; 1991.

[7] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Rob. Res. 1986;5:90–98.

[8] Kavraki LE, Svestka P, Latombe J-C, Overmars MH. Probabilistics roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Rob. Autom. 1996;12:566–580.

[9] LaValle SM, Kuffner JJ. Rapidly-exploring random trees: progress and prospects. In: Donald BR, Lynch K, Rus D, editors. New directions in algorithmic and computational robotics. Wellesley (MA): AK Peters; 2001. p. 293–308.

[10] Brock O, Khatib O. Elastic strips: a framework for motion generation in Human enviroments. Int. J. Rob. Res. 2002;21:1031–1052.

[11] Yang Y, Brock O. Elastic roadmaps – motion generation for autonomous mobile manipulation. Auton. Robots. 2010;28:113–130.

[12] Antonelli G, Arrichiello F, Chiaverini S. The null-space-based behavioral control for autonomous robotic systems. Intell. Serv. Rob. 2007;1:27–39.

[13] Lee J, Mansard N, Park J. Intermediate desired value approach for task transition of robots in kinematic control. IEEE Trans. Rob. 2012;28:1260–1277.

[14] Lee J, Mansard N, Park J. Intermediate desired value approach for continuous transition among multiple tasks of robots. In: IEEE International Conference on Robotics and Automation; Shanghai, China; 2011 May 9–13. p. 1276–1282.

[15] Yoshida E, Kanehiro F. Reactive humanoid motion planning for reaching tasks. Workshop on robot motion planning: online, reactive, and in real-time. In: IEEE/RSJ International Conference on Intelligent Robots and Systems; Vilamoura, Portugal; 2012 Oct 7–12.

[16] Falco P, Natale C. On the stability of closed-loop inverse kinematics algorithms for redundant robots. IEEE Trans. Rob. 2011;27:780–784.

[17] De Santis A, Albu-Schäffer A, Ott C, Siciliano B, Hirzinger G. The skeleton algorithm for self-collision avoidance of a humanoid manipulator. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics; Zurich (CH); 2007 Sept 4–7. p. 1–6.

[18] Bischoff R, Huggenberger U, Prassler E. KUKA youBot – a mobile manipulator for research and education. In: IEEE International Conference on Robotics and Automation; Shanghai, China; 2011 May 9–13. p. 1–4.

[19] Ceriani NM, Buizza Avanzini G, Zanchettin AM, Bascetta L, Rocco P. Optimal placement of spots in distributed proximity sensors for safe human-robot interaction. In: IEEE International Conference on Robotics and Automation; Karlsruhe, Germany; 2013. p. 5838–5843.

[20] Laumond JP. Kineo CAM: a success story of motion planning algorithms. IEEE Rob. Autom. Mag. 2006;13: 90–93.