Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Motion Planning for Mobile Manipulators along Given End-effector Paths

Giuseppe Oriolo Christian Mongillo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Eudossiana 18, 00184 Roma, Italy
oriolo@dis.uniroma1.it, chrismongillo@infinito.it

***Abstract*— We consider the problem of planning collision-free motions for a mobile manipulator whose end-effector must travel along a given path. Algorithmic solutions are devised by adapting a technique developed for fixed-base redundant robots. In particular, we exploit the natural partition of generalized coordinates between the manipulator and the mobile base, whose nonholonomy is accounted for at the planning stage. The approach is based on the randomized generation of configurations that are compatible with the end-effector path constraint. The performance of the proposed algorithms is illustrated by several planning experiments.**

***Index Terms*— Mobile manipulators, probabilistic motion planning.**

I. INTRODUCTION

Mobile manipulators combine the two archetypes of robotic systems, i.e., articulated arms and mobile platforms: hence, they exhibit the dexterity and grasping capability of the former and the sensor-based mobility of the latter. Several prototypes of mobile manipulators already exist; see [1-2] for some examples. However, many research aspects still need to be addressed in order to fully exploit the potential of these mechanisms. In fact, the arm and the mobile platform are often treated as distinct entities, neglecting their kinematic and dynamic interaction, whereas their most effective use is expected to rely on the coordinated use of locomotion and manipulation functions [3].

Planning collision-free motions under task constraints is a typical problem where whole-system coordination is crucial. In many applications, the mobile manipulator is required to move the end-effector along a given path in order to realize the task specified by a higher-level module (e.g., for inspection missions with an in-hand camera, or in pick and place operations). A lower-level planner is then in charge of generating joint paths that realize the desired end-effector motion while guaranteeing that the robot avoids collisions with obstacles or with itself. We call this problem Motion Planning along End-effector Paths (MPEP).

Since mobile manipulators are kinematically redundant with respect to end-effector tasks, the MPEP problem can be attacked as an optimal redundancy resolution problem, with the additional difficulty that the mobile platform is often subject to nonholonomic constraints; one possibility is therefore to adapt kinematic [4] or optimal [5] control schemes. However, none of the above solutions is satisfactory when the objective is obstacle avoidance. For example, the optimal control formulation of the MPEP problem for mobile manipulators proposed in [6] leads to a nonlinear TPBVP whose solution can only be sought numerically, without any guarantee of success.

The objective of this paper is to present a family of probabilistic planners for solving the MPEP problem in mobile manipulators by extending our previous work dealing with the same problem in fixed-base redundant manipulators [7]. We exploit the natural partition of generalized coordinates between the manipulator and the mobile platform, and take into account the presence of nonholonomic constraints at the planning stage. All the planners rely on the same mechanism for generating random configurations that are compatible with the end-effector constraint.

The paper is organized as follows. In the next section, we give a precise formulation of the MPEP problem for mobile manipulators and clarify what we consider to be a solution. The procedure for generating random configurations is then presented, and the various proposed planners are described. Results for problems of increasing complexity are finally presented to illustrate the performance of the algorithms.

II. MPEP PROBLEM FOR MOBILE MANIPULATORS

In this section, we generalize our formulation of the MPEP problem [7] so as to apply to the mobile manipulator case. With respect to fixed-base manipulators, the essential features of mobile manipulators are the natural partition of generalized coordinates (mobile platform/manipulator) and the nonholonomy due to the rolling wheels.

Consider a mobile manipulator whose task is to move the end-effector along a given path in a workspace populated by obstacles. The direct kinematics is expressed as

$$p = f(q) = f\begin{pmatrix} q^p \\ q^m \end{pmatrix}, \tag{1}$$

where $p \in I\!R^M$ is the end-effector *pose* (position and/or orientation) and $q \in I\!R^N$ is the system configuration[1], consisting of the platform configuration $q^p \in I\!R^{N_p}$ and the manipulator configuration $q^m \in I\!R^{N_m}$, with $N_p + N_m = N$. While the manipulator subsystem is holonomic (i.e., arbitrary motions are possible for the manipulator configuration q^m), the motion of the platform is generated as

$$\dot{q}^p = G(q^p)u, \tag{2}$$

[1]We consider euclidean spaces for simplicity, but our developments apply to the case in which p, q^p and q^m are defined over manifolds.

where $u \in I\!R^P$ are pseudovelocities (typically, linear and angular platform velocities), while the columns of $G(q^p)$ span the null space of the nonholonomic constraint matrix.

An end-effector path $p(\sigma)$ is assigned, with $\sigma \in [0,1]$ the path parameter. For the problem to be well-posed, we assume that $\forall\,\sigma \in [0,1]$, $p(\sigma) \in \mathcal{T}$, where $\mathcal{T} \subset I\!R^M$ is the *dextrous task space*, defined as the set of end-effector poses that can be realized by ∞^{N-M} configurations[2].

Assume that the mobile manipulator is kinematically redundant with respect to the given task, i.e., $N > M$. Then, the MPEP problem is to find a configuration path $q(\sigma) = (q^p(\sigma), q^m(\sigma))$ such that:

1) $p(\sigma) = f(q(\sigma))$, $\forall \sigma \in [0,1]$;
2) the robot does not collide with obstacles or itself;
3) the path is feasible w.r.t. the kinematic constraints that may exist (e.g., manipulator joint limits);
4) the path is feasible w.r.t. the nonholonomic constraints, i.e., $q^p(\sigma)$ is a solution of eq. (2).

Depending on the application, an initial joint configuration $q(0)$ such that $p(0) = f(q(0))$ may or not be assigned. The first version of the problem is more constrained (and thus possibly easier to solve) than the second.

We seek a solution to the MPEP problem in the form of a sequence of configurations:

$$\{q(\sigma_0), q(\sigma_1), \ldots, q(\sigma_{s-1}), q(\sigma_s)\}, \quad \sigma_0 = 0, \sigma_s = 1,$$

with the σ_i's equispaced and $p(\sigma_i) = f(q(\sigma_i))$. The integer s is called *path sampling*. A continuous path will be derived from this sequence by joining successive configurations by a *local planner*; this may use simple linear interpolation[3] for q^m, while the mobile platform nonholonomy must be taken into account when joining successive values of q^p.

III. Generation of Random Configurations

The algorithms we have developed for solving the MPEP problem share the same basic tool, i.e., a procedure which performs random sampling of self-motion manifolds. The mechanism proposed in [7] for generating N-dimensional random configurations compatible with the M-dimensional task constraint is based on a partition of q into M *base* and $N - M$ *redundant* variables; the value of the latter is first randomly generated, and the value of base variables is then computed by inverse kinematics in such a way that the resulting configuration places the end-effector at a certain point of its assigned path[4]. In this section, we show how this basic strategy can be adapted to mobile manipulators.

Assume for illustration that the platform is a unicycle, described by coordinates x, y, θ (position and orientation) and controlled by pseudovelocities v, ω (linear and angular velocity), while the manipulator is a spatial three-dof arm with rotational joints (Fig. 1). Hence, we have $q^p \in I\!R^3$, $q^m \in I\!R^3$, and $q \in I\!R^6$. The end-effector task is specified at the position level, i.e., $p \in I\!R^3$. The extension to the case where orientation is also specified is straightforward (provided that the manipulator has sufficient dof's).

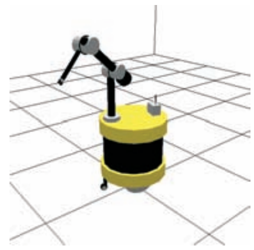


Fig. 1. The mobile manipulator considered in this paper

As the configuration q is naturally split between platform variables q^p and manipulator variables q^m, it is reasonable to choose the base/redundant partition accordingly. Since in our case $M = 3$ and $N = 6$, we must select three variables as redundant, and generate their values randomly. Throughout the paper, we use the *platform* variables for this purpose. Other possibilities (e.g., using the manipulator variables or a mixed set) are not discussed here.

With the assumptions of Sect. II, each pose $p(\sigma_i) \in \mathcal{T}$ along the given end-effector path can be realized by $\infty^{N-M} = \infty^3$ configurations of the mobile manipulator, which represent the so-called *self-motion manifold*[5]. Assume that the configuration q_i^p of the platform is randomly chosen. For each value of $p_i = p(\sigma_i)$, $i = 0, \ldots, s$, there exist a finite number (up to 4, in our case) of manipulator configurations $q_i^m = q^m(p_i, q_i^p)$ such that $p_i = f(q_i^p, q_i^m)$, computed by inverting the kinematic map (1) with $q^p = q_i^p$. Depending on the value of q_i^p, it may happen that no value of q_i^m is compatible with the end-effector pose p_i.

According to the above strategy, the procedure generating a random sample of the self-motion manifold corresponding to p_i is described in pseudocode as follows:

RAND_CONF($p_i, q_{\rm bias}$)
  $q_i^p \leftarrow$ RAND_PLATFORM($q_{\rm bias}^p$)
  $q_i^m \leftarrow$ INV_KIN($p_i, q_i^p, q_{\rm bias}^m$)
  if INV_KIN_FAIL
    Return RAND_CONF_FAIL
  else Return $q_i \leftarrow (q_i^m, q_i^p)$

RAND_PLATFORM($q_{\rm bias}^p$)
  $u_i \leftarrow$ RAND_INP($q_{\rm bias}$)
  $q_i^p \leftarrow$ MOVE($q_{\rm bias}^p, u_i$)
  Return q_i^p

The effect of the optional argument $q_{\rm bias}$, which appears in both procedures, is to bias the distribution characterizing the randomly generated samples. When $q_{\rm bias}$ is present, RAND_CONF returns (if successful) a configuration q_i such that (*i*) there is a simple feasible path connecting $q_{\rm bias}^p$ to q_i^p, (*ii*) $p_i = f(q_i)$, and (*iii*) $\|q_i^m - q_{\rm bias}^m\|_\infty < d$, where d is a maximum allowed joint displacement. The first of these properties is guaranteed by RAND_PLATFORM, which generates a random pseudovelocity vector u_i and then computes (by forward integration of eq. (2)) the platform

[2] $\mathcal{T}$ does not contain its boundary, which includes the unavoidable singularities realized by a single configuration.

[3] This will lead to an path error between successive poses, whose entity can however be reduced at will by increasing the sampling s.

[4] This sampling mechanism is similar to the one used in [8] for guaranteeing the closure constraint.

[5] To be precise, the inverse image of any point $p \in \mathcal{T}$ is in general a finite number of disjoint manifolds.

configuration q_i^p reached from $q_{\rm bias}^p$ by applying u_i over a sampling interval. The last two properties are enforced by INV_KIN, which takes as inputs p_i and q_i^p and seeks an inverse solution q_i^m for the manipulator joints which also satisfies the displacement constraint. If no such solution exists (either because no inverse solution exists for the chosen q_i^p or because all solutions violate the displacement constraint) the boolean INV_KIN_FAIL becomes *true*.

If $q_{\rm bias}$ is absent, RAND_PLATFORM simply generates a random configuration q_i^p; hence, the self-motion manifold sample computed by RAND_CONF is not biased by any configuration. Finally, when RAND_CONF is invoked with no argument, a completely random configuration (not belonging to any self-motion manifold) is produced.

The reason for biasing the random generation of q_i with $q_{\rm bias}$ is that all the algorithms to be presented work in an incremental fashion, trying to build a connectivity roadmap from the initial end-effector pose. When a sample q_i has been randomly generated on the self-motion manifold corresponding to p_i, it is used as $q_{\rm bias}$ for the next self-motion manifold in order to guarantee that (*i*) q_i^p and q_{i+1}^p are connected by a feasible path, and (*ii*) q_{i+1}^m will be sufficiently close to q_i^m. The latter fact will ensure that the end-effector constraint violation between q_i and q_{i+1} due to the use of linear interpolation for q^m is reduced.

We now give more details about two important issues.

A. The Compatible Platform Region

Given an end-effector pose p_i, it would be inefficient to generate the random sample q_i^p (i.e., the mobile platform configuration) in the whole configuration space. The condition for the existence of an inverse kinematic solution q_i^m for the manipulator is that q_i^p 'places' the manipulator workspace so as to contain p_i. We could also say that q_i^p must 'belong' to the self-motion manifold of p_i.

For simplicity, we shall use a coarse estimation of this manifold, based on the geometric argument in Fig. 2. With the manipulator in full extension (elbow singularity), a circle is drawn with the center at p_i and passing through the platform center. Such circle (shown in gray) identifies the platform positions that we consider to be *compatible* with the end-effector constraint. Therefore, randomly generated q_i^p's placing the platform outside this region are discarded.

Note that, even if q_i^p is such that the platform position lies in the circle, there is no guarantee that an inverse kinematic solution q_i^m exists. This will depend on the orientation component of q_i^p as well as on the vertical component of p_i. However, the number of failures of INV_KIN using the above approximation of the self-motion manifold was found to be very low and therefore acceptable[6].

If RAND_PLATFORM is called without a $q_{\rm bias}$, the position part of q_i^p is directly generated inside the compatible region. Finally, if RAND_PLATFORM is called without any argument (i.e., without a particular end-effector pose p_i), the compatible region is assumed to be the union of all the circles centered on the given end-effector path.

[6] A more accurate computation is possible following the ideas in [9].

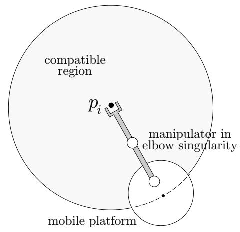


Fig. 2. Construction of the compatible platform region

B. Generation of Random Pseudovelocities

In practice, the pseudovelocities of the platform will be bounded. Let $\mathcal{U} = \mathcal{V} \times \Omega = [v_{\rm min}, v_{\rm max}] \times [\omega_{\rm min}, \omega_{\rm max}]$ be the set of admissible pseudovelocities.

The procedure RAND_INP for generating random pseudovelocity inputs is crucial, as it is responsible for the platform motion and, ultimately, for the effectiveness of the solution space exploration. In the remainder of this section, we discuss the different strategies developed to such end.

1) Completely random pseudovelocities: v, ω are generated according to a uniform probability distribution in $\mathcal{U}$.

2) Constant-energy pseudovelocities: v is generated according to a uniform probability distribution in $\mathcal{V}$, while ω is computed from the following equation:

$$v^2 + c\,\omega^2 = \gamma^2,$$

where $c > 0$ is a weighting factor and γ^2 is the desired input energy level.

3) Optimal pseudovelocities: v and ω are chosen among a set $\mathcal{C}$ of candidate pseudovelocities so as to optimize a certain criterion. In particular, $\mathcal{C}$ consists of four pseudovelocity vectors, randomly chosen within the subregions

$$\begin{aligned}
\mathcal{U}_1 &= [0, v_{\rm max}] \times [0, \omega_{\rm max}] && \text{(a forward-right motion)}\\
\mathcal{U}_2 &= [0, v_{\rm max}] \times [\omega_{\rm min}, 0] && \text{(a forward-left motion)}\\
\mathcal{U}_3 &= [v_{\rm min}, 0] \times [\omega_{\rm min}, 0] && \text{(a backward-right motion)}\\
\mathcal{U}_4 &= [v_{\rm min}, 0] \times [0, \omega_{\rm max}] && \text{(a backward-left motion)}
\end{aligned}$$

These four vectors may be generated as either completely random or constant-energy within their domain. We propose two different choices for the optimality criterion: the first (to be minimized) is $I_{\rm dist} = \|q_{\rm des} - q_{\rm new}\|$, i.e., the (weighted) euclidean distance between the new configuration $q_{\rm new}$ (obtained by moving the platform from $q_{\rm bias}^p$ with the candidate pseudovelocity and then computing the manipulator posture through inverse kinematics) and a desired configuration $q_{\rm des}$ (whose role will be clarified later, see Sect. IV-B). The second criterion $I_{\rm comp}$ (to be maximized) is the *task compatibility* [10] of $q_{\rm new}$, which quantifies the motion capability of the mobile manipulator along the end-effector path starting from the posture $q_{\rm new}$.

The different effects of the above strategies will be illustrated in the planning experiments of Sect. V.

IV. Planning Algorithms

Having discussed the procedure for generating random samples of a desired self-motion manifold, we now present the algorithms developed for the solution of the MPEP problem for mobile manipulators. All of them make use of the collision checking procedure NO_COLL. When invoked with a single argument q_i, it performs a collision check (including self-collisions) and returns *true* if q_i is safe. When invoked with two arguments (q_i, q_j), it performs a collision check on both configurations as well as on the path joining them (by sampling it at a sufficiently high rate). In particular, a feasible path (produced under the action of the constant pseudovelocity inputs selected by RAND_INP) is used for for the mobile platform, while the manipulator path is obtained by linear interpolation.

A. Greedy Planner

The core of the first algorithm is the STEP function which, given two generic poses p_i, p_k $(0 \leq i < k \leq s)$ belonging to the end-effector sequence and a configuration q_i on the self-motion manifold of p_i, builds a (sub)sequence of configurations $\{q_i, \ldots, q_k\}$ connecting p_i to p_k and such that collisions are avoided along the path. If successful, STEP returns the sequence in the variable PATH.

```
STEP(i, p_i, q_i, k)
 for j = i to k − 1 do
   l ← 0; Succ ← 0;
   while l < MAX_SHOTS and !Succ do
      q_{j+1} ← RAND_CONF(p_{j+1}, q_j);
      if !RAND_CONF_FAIL and NO_COLL(q_j, q_{j+1})
         Succ ← 1; ADD_TO_PATH(q_{j+1});
      l ← l + 1;
   if l = MAX_SHOTS
      Return STEP_FAIL
   else
      j ← j + 1;
 Return PATH
```

The parameter MAX_SHOTS represents the upper bound to the number of calls to RAND_CONF(p_{j+1},q_j) for each end-effector pose p_j. If RAND_CONF succeeds in finding a configuration q_{j+1} realizing p_{j+1}, sufficiently close to the bias configuration q_j, and such that the path between q_j^p and q_{j+1}^p is feasible, the whole path between q_j and q_{j+1} is verified to be collision-free; in this case, q_{j+1} is added to the current sequence through the ADD_TO_PATH function. If the maximum number of trials of RAND_CONF is exceeded, the procedure returns STEP_FAIL.

A direct approach to the solution is to devise a greedy algorithm based on iterated calls to the STEP function with p_0, p_s as subsequence extrema and random q_0.

```
GREEDY algorithm
   j ← 0;
   while j < MAX_ITER and STEP_FAIL do
     q_0 ← RAND_CONF(p_0);
     STEP(0, p_0, q_0, s);
     j ← j + 1;
   if !STEP_FAIL
     Return PATH
   else
     Return FAILURE
```

Given the initial pose p_0, RAND_CONF(p_0) generates an initial configuration q_0 as described in the previous section. STEP is then invoked to search for a sequence of configurations guaranteeing feasible collision-free motion while the end-effector moves from p_0 to p_s. In case of success, the path found by STEP is returned. If STEP fails and MAX_ITER has not been exceeded, a new q_0 is generated and STEP starts a new search from q_0.

GREEDY implements a depth-first search, as for any initial configuration q_0 a sequence of random configurations (one for each self-motion manifold, and each biased by the previous one) is generated, and discarded if STEP does not reach the last self-motion manifold. Experiments have shown that this planner is effective in easy problems (see Sect. V), essentially due to the end-effector path constraint, which greatly reduces the admissible internal motions of the robot once a q_0 has been chosen. Still, the only possible way to backtrack for this planner is to generate a new q_0, and this may prove inefficient in complex problems.

B. RRT-Like Planner

To overcome the limitations of the depth-first algorithm GREEDY, one may try to generate *multiple* random samples for each self-motion manifold and to connect configurations on successive manifolds by local paths. As in [7], this exploratory behavior is achieved by the RRT_LIKE algorithm, which adapts the notion of RRT (Rapidly-exploring Random Tree, [11]) to mobile manipulators.

Our algorithm tries to expand a tree τ rooted at q_0, a random sample of the p_0 self-motion manifold, until the self-motion manifold of p_s is reached. If the expansion fails a certain number of times, a different q_0 is generated and another tree is built, until the maximum number of iterations is exceeded. If a tree connecting p_0 to p_s is found, a path is extracted by graph search techniques.

```
RRT_LIKE algorithm
   j ← 0;
   while p_new != p_s and j < MAX_ITER do
     q_0 ← RAND_CONF(p_0);
     CREATE(τ, q_0);
     i ← 0;
     repeat
       p_new ← EXTEND_LIKE(τ);
       i ← i + 1;
     until p_new = p_s or i = MAX_EXT
     j ← j + 1;
   if p_new = p_s
     Return τ
   else
     Return FAILURE
```

```
EXTEND_LIKE(τ)
   q_rand ← RAND_CONF;
   (q_near, k) ← NEAR_NODE(q_rand, τ);
   q_new ← RAND_CONF(p_{k+1}, q_near);
   if !INV_KIN_FAIL and NO_COLL(q_near, q_new)
      ADD_NODE(τ, q_new);
      ADD_EDGE(τ, q_near, q_new);
      Return p_{k+1}
   else
      Return NULL
```

First, RAND_CONF is called with no arguments to find a random $q_{\mathrm{rand}} = (q^p_{\mathrm{rand}}, q^m_{\mathrm{rand}})$, and NEAR_NODE

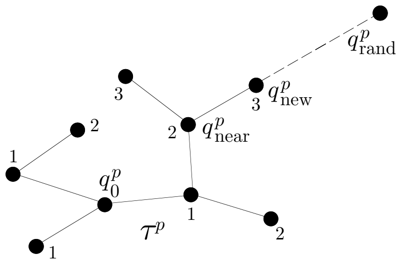


Fig. 3. The tree expansion in the subspace of platform configurations. The integer k associated to each node identifies the end-effector pose p_k of which the node is a preimage. Note that the path between adjacent configurations, represented here as a segment for simplicity, is by construction feasible w.r.t. the nonholonomic constraints.

identifies q_{near}, the node of τ closest[7] to q_{rand} with respect to the platform variables, and returns the index k of the end-effector pose p_k to which q_{near} is associated. Then, RAND_CONF computes q_{new} by generating first a random input (using one of the strategies described in Sect. III-B), then the corresponding q^p_{new} starting from q^p_{near}, and finally (if successful) q^m_{new} by inverse kinematics on the manifold associated to p_{k+1}. The path joining q_{near} to q_{new} is now checked for collision; if the result is negative, τ is expanded and p_{i+1} is returned. Figure 3 shows the expansion of the tree τ in the subspace of platform configurations.

The role of q_{rand} in guiding the expansion of RRT_LIKE is only to identify q_{near}, the closest node to q_{rand} w.r.t. the platform variables; the direction of expansion from q_{near} is then determined by the choice of the pseudovelocity input, and in general does not depend on q_{rand}. This is quite different from what happens in the classical RRT algorithm, where the direction of expansion is chosen to be the line joining q_{rand} with q_{near}. To retain this strategy, which is indeed essential for the effectiveness of RRT (namely, for driving the expansion toward wide Voronoi regions), one can adopt the optimal pseudovelocity generation outlined in Sect.III-B, using the I_{dist} criterion with $q_{\mathrm{des}} = q_{\mathrm{rand}}$.

C. Variations on RRT_LIKE

The expansion of τ toward randomly selected directions gives to RRT_LIKE an exploratory attitude which, for the MPEP problem, could prove inefficient due to the strong constraint represented by the end-effector path. However, it is possible to modify the RRT_LIKE planner by alternating depth-first searches with expansion steps. This can be done by invoking the STEP function right after the EXTEND_LIKE operation has been executed; the arguments passed to STEP are p_l, p_s, where p_l is the closest pose to p_s reached so far by the algorithm. This modified RRT-based planner, which tries at the same time to explore the portion of configuration space consistent with the end-effector path constraint and to approach the goal self-motion manifold through a greedy search, is called RRT_GREEDY.

The exploratory attitude of RRT_LIKE may also be a drawback when the start and goal self-motion manifolds are very distant. Inspired by [12], one possible solution is to expand two trees, respectively rooted at the start and at the goal self-motion manifolds. The trees will 'meet' on some intermediate manifold, but different nodes will be generated; it is then necessary to compute a self-motion connecting the two nodes by an RRT-based search restricted to the manifold. This planner is called RRT_BIDIR.

V. Planning Experiments

We now present some MPEP experiments for the mobile manipulator of Fig. 1. The algorithms were implemented in C on a 1 Ghz PC and integrated in the software platform Move3D, dedicated to motion planning and developed at LAAS-CNRS, France[8].

The first two experiments aim at highlighting the effects of the different pseudovelocity generation procedures of Sect.III-B: to this end, the mobile manipulator must move its end-effector along a given rectilinear path in the *absence* of obstacles, and only the RRT_LIKE planner is used.

For the first experiment, Fig. 4 compares the solution tree (more precisely, its projection on the x, y plane) obtained by completely random pseudovelocity inputs (left) and constant-energy optimal inputs with criterion I_{dist} (right). The left tree is uselessly erratic (given that no obstacles are present) if compared with the right one, which extends mostly along the direction of the task trajectory. The table reports a performance comparison (averaged over 20 trials) between these two and other methods, in terms of time needed to find a solution, failures of INV_KIN, and nodes in the tree. The superiority of constant-energy methods is confirmed by the smaller number of kinematic inversion failures and by the reduced tree size. As for the choice of the performance criteria, the minimization of I_{dist} appears to be more effective than the maximization of I_{comp}. I_{mix} denotes a weighted criterion that combines the two.

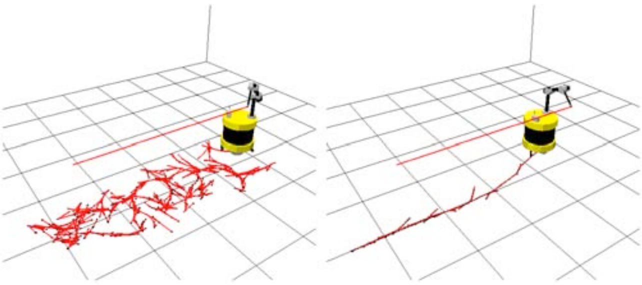
The second experiment is similar to the first, but a different initial configuration has been assigned. In particular, the arm is completely stretched and the mobile platform orientation is orthogonal to the end-effector path; hence, q_0 has a low task compatibility with respect to the given end-effector trajectory. Figure 5 shows the results obtained by RRT_LIKE using constant-energy optimal inputs with the performance criteria I_{dist} (left) and I_{comp} (right). In this case, the use of I_{comp} allows the robot to recover and maintain a higher compatibility value (note the absence of kinematic inversion failures), ultimately resulting in a smoother motion and in a smaller computation time.

Experiments of the second group take place in environments with obstacles, and aim at comparing the performance of the various planners. In these trials, pseudovelocity inputs are generated by optimizing the mixed performance criterion I_{mix} over a set of four constant-energy candidates in $\mathcal{U}_1, \ldots, \mathcal{U}_4$ (see Sect. III-B). As before, the planners' performances are averaged over 20 trials.

The third experiment scene is quite simple. The robot must move its end-effector along a polynomial path through

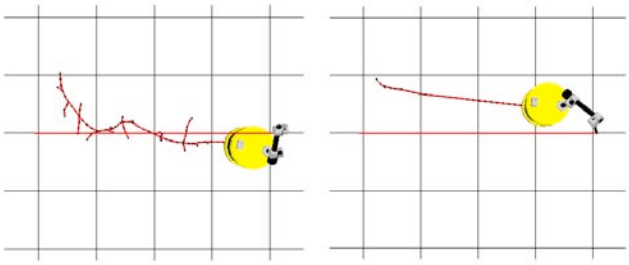
[7]Here, we use a weighted euclidean metric for simplicity, but a more appropriate nonholonomic metric can be adopted.

[8]Move3D is at the origin of the product KineoWorks currently marketed by the company Kineo CAM (www.kineocam.com).



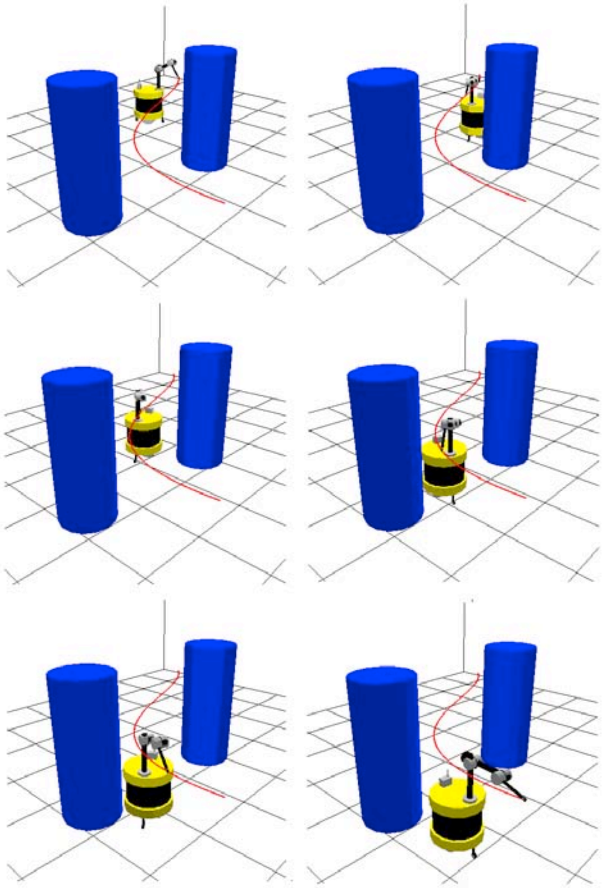
Pseudovelocity generation	time (s)	# kin fail	# nodes
completely random	2.16	41	165
constant energy	0.50	17	127
optimal (comp. rand., I_{dist})	2.00	59	158
optimal (const. ene., I_{dist})	0.33	4	96
optimal (const. ene., I_{comp})	1.00	10	189
optimal (const. ene., I_{mix})	0.60	21	115

Fig. 4. First experiment: The tree built on the x, y plane by RRT_LIKE using completely random inputs (left) and constant-energy optimal inputs with criterion I_{dist} (right). Also reported is a comparison extended to other input generation methods.



Criterion	time (s)	# kin fail	# nodes
I_{dist}	0.90	145	78
I_{comp}	0.64	0	55

Fig. 5. Second experiment: The tree built on the x, y plane by RRT_LIKE using constant-energy optimal inputs with criteria I_{dist} (left) and I_{comp} (right), with a comparison between the two methods



Planner	time (s)	# cc	# nodes
GREEDY	1.60	163	61
RRT_LIKE	3.48	187	141
RRT_GREEDY	2.08	208	119
RRT_BIDIR	2.26	152	172

Fig. 6. Third experiment: Solution obtained with GREEDY (left to right, and top to bottom) and comparison of planners' performance

two columns. Figure 6 contains some frames from the solution path computed by GREEDY and a table summarizing the performance of the planners. In this case, the GREEDY planner performs best under all aspects due to its depth-first strategy, which is invariably more effective in easy planning problems. Also RRT_BIDIR achieves a good result thanks to the large space available for reconfiguration.

Figure 7 shows the scene of the fourth experiment; the end-effector must follow a rectilinear path which is dangerously close to an obstacle. The solution shown was planned by RRT_GREEDY; note how the robot stretches the arm to move under the obstacle. RRT-based algorithms perform best in this case thanks to their exploratory attitude, with the exception of RRT_BIDIR, which is penalized by the reduced space for reconfiguration in the contact manifold.

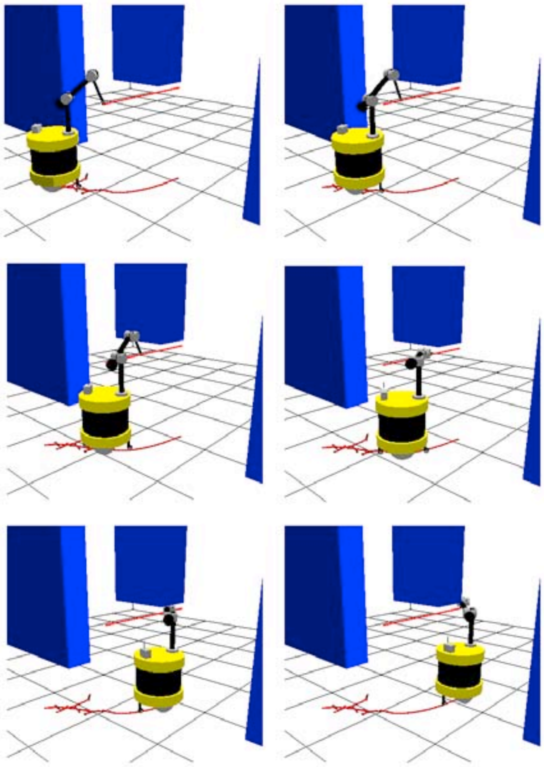
The fifth planning problem is very difficult: to complete its task, the mobile manipulator must first cross a narrow passage and then carefully move its arm so as to drive the end-effector between the sandwich-shaped obstacle. While GREEDY failed to produce a solution within the allotted time, RRT-based planners performed quite well.

A number of experiments have confirmed the above indications. Essentially, GREEDY is very effective when dealing with simple queries, while RRT_LIKE and RRT_GREEDY perform much better when the difficulty of the problem increases. The bidirectional strategy of RRT_BIDIR is convenient when there is a large space available for reconfiguration, as in the case of Fig. 9.

VI. CONCLUSIONS

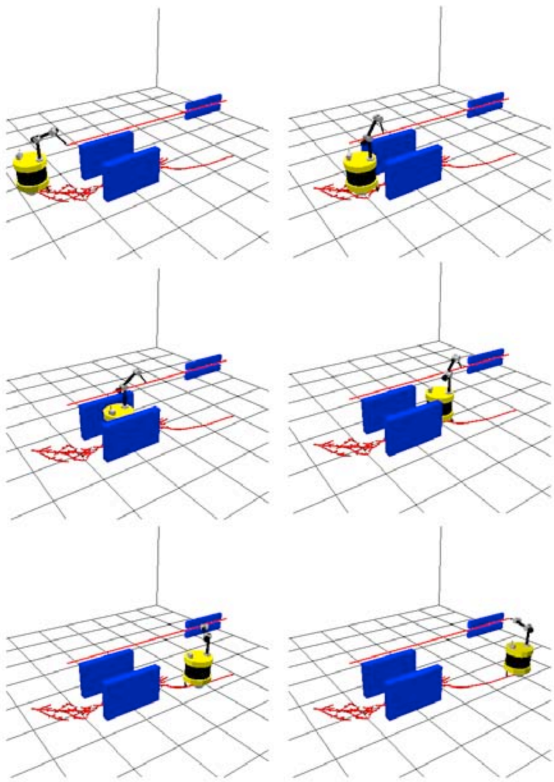
Single-query probabilistic planners have been presented for the problem of generating collision-free motions for a nonholonomic mobile manipulator moving along a given end-effector path. Experiments show that simple instances of the problem can be solved more efficiently by a greedy approach, whereas the breadth-first search of RRT-based planners is needed to deal with more complex cases.

Among the issues deserving further attention, we mention *(i)* proving probabilistic completeness along the lines of [8,11], *(ii)* complexity analysis, and *(iii)* the choice of performance criteria for pseudovelocity generation, with the ideas in [13] as possible inspiration. An extension of the proposed methods to sensor-based exploration can also be envisaged following the approach in [14].



Planner	time (s)	# CC	# nodes
GREEDY	12.6	478	40
RRT_LIKE	6.3	218	115
RRT_GREEDY	6	278	76
RRT_BIDIR	33	1230	511

Fig. 7. Fourth experiment: Solution obtained with RRT_GREEDY and comparison of planners' performance



Planner	time (s)	# cc	# nodes
GREEDY	-	-	-
RRT_LIKE	33	1768	560
RRT_GREEDY	47	5439	229
RRT_BIDIR	230	15769	1322

Fig. 8. Fifth experiment: Solution obtained with RRT_LIKE and comparison of planners' performance

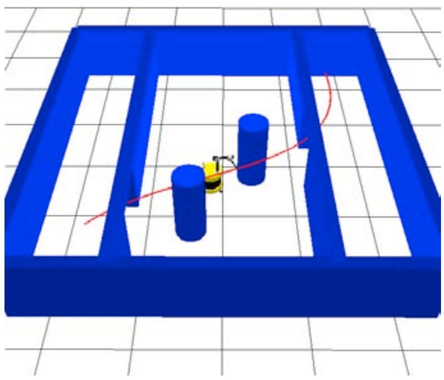


Fig. 9. A problem for which RRT_BIDIR is more efficient than other planners

REFERENCES

[1] D. Apostolopoulus, M. Wagner, and W. Whittaker, "Technology and field demonstration results in the robotic search for antarctic meteorites," *Field and Service Robotics Conf.*, 1999.

[2] M. Nechyba and Y. Xu, "Human robot coordination in space: (sm)ˆ 2 for new space station structure," *IEEE Robotics and Automation Mag.*, vol. 2, no. 4, pp. 4–14, 1995.

[3] Y. Yamamoto and X. Yun, "Coordinating locomotion and manipulation of a mobile manipulator," *IEEE Trans. on Automatic Control*, vol. 39, pp. 1326–1332, 1994.

[4] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *J. of Intelligent and Robotic Systems*, vol. 3, pp. 201–212, 1990.

[5] D. P. Martin, J. Baillieul, and J.M. Hollerbach, "Resolution of kinematic redundancy using optimization techniques," *IEEE Trans. on Robotics and Automation*, vol. 5, pp. 529–533, 1989.

[6] A. Mohri, S. Furuno, and M. Yamamoto, "Trajectory planning of mobile manipulator with end-effector's specified path," *2001 IEEE Int. Conf. on Intelligent Robots and Systems*, vol. 4, pp. 2264–2269, 2001.

[7] G. Oriolo, M. Ottavi, and M. Vendittelli, "Probabilistic motion planning for redundant robots along given end-effector paths," *2002 IEEE Int. Conf. on Intelligent Robots and Systems*, vol. 2, pp. 1657–1662, 2002.

[8] L. Han and N. Amato, "A kinematic-based probabilistic roadmap method for closed chain systems," *4th Int. Work. on Algorithmic Foundations of Robotics*, pp. 233–246, 2000.

[9] J. Cortes, T. Simèon, and J. P. Laumond, "A random loop generator for planning the motions of closed kinematic chains using prm methods," *2002 IEEE Int. Conf. on Robotics and Automation*, pp. 2141–2146, 2002.

[10] S. L Chiu, "Task compatibility of manipulator postures," *The Int. J. of Robotics Research*, vol. 7, pp. 13–21, 1988.

[11] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Technical Report No. 98-11, Computer Science Dept., Iowa State University.*, 1998.

[12] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," *2000 IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 995–1001, 2000.

[13] P. Leven and S. Hutchinson, "Using manipulability to bias sampling during the construction of probabilistic roadmaps," *IEEE Trans. on Robotics and Automation*, vol. 19, pp. 1020–1026, 2003.

[14] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso, "The srt method: Randomized strategies for exploration," *2004 IEEE Int. Conf. on Robotics and Automation*, pp. 4688–4694, 2004.