Planning Pick-and-Place tasks with two-hand regrasping

Jean-Philippe Saut[1,2], Mokhtar Gharbi[1,2], Juan Cortés[1,2], Daniel Sidobre[1,2], Thierry Siméon[1,2]
{jpsaut, mgharbi, jcortes, daniel, nic}@laas.fr
[1]CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[2]Université de Toulouse; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France

***Abstract*— This paper proposes a planning framework to deal with the problem of computing the motion of a robot with dual arm/hand, during an object pick-and-place task. We consider the situation where the start and goal configurations of the object constrain the robot to grasp the object with one hand, to give it to the other hand, before placing it in its final configuration. To realize such a task, the proposed framework treats the grasp computation, for one or two multi-fingered hands, of an arbitrarily-shaped object, the exchange configuration and finally the motion of the robot arms and body. In order to improve the planner performance, a context-independent grasp list is computed offline for each hand and for the given object as well as computed offline roadmap that will be adapted according to the environment composition. Simulation results show the planner performance on a complex scenario.**

I. Introduction

While humanoid torso robots offer better manipulation capacities compared to single arm/hand robots, they also introduce new issues. For instance, in the case of a two-arm robot, if the object start and goal configurations are not within the workspace of the same arm, it is necessary to change the grasping hand to achieve the task. An efficient and elegant way to change the grasp is to plan a dual-hand grasp of the object.

Among the earlier work on this topic, [1] proposed a practical planner for arms manipulating an object using a predefined set of grasps. More recent work addressed a similar problem for the case of dual-arm manipulation with regrasping for a humanoid robot [2].

This paper describes a practical framework to resolve the pick-and-place problems with a humanoid robot in scenarios such as the one illustrated in Fig. 1. DLR's robot Justin [3] equipped with two multi-fingered hands has to pick the horse statuette from its right side and place it at its left. The pick-and-place problem is then defined by the initial and final configurations of the robot and of the object. The idea developed in this work is to use several offline computed data structures such as grasp list and robot roadmap to reduce the online planning time. First, a scored grasp list is generated for each hand, using the method presented in Section IV-A. Also, a roadmap is preprocessed using the coordination roadmap approach [4] described in Section V-A, that does not account for the manipulated object. To resolve a specific pick-and-place task (online planning), a list of collision-free grasp configurations are generated and sorted given some criterion as described in Section V-C. These two lists are used to filter the large number of double grasp candidates. A scored double grasp list is then produced (Section IV-B). The top-ranked double grasps are used to define the path planning queries and determine the grasp exchange position (Section V-B).

This work has been supported by the European Community's Seventh Framework Program FP7/2007-2013 "DEXMART" under grant agreement no. 216239. This work was also supported by the ANR project ASSIST.

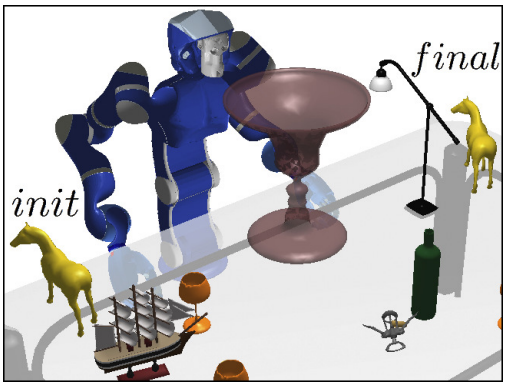

Fig. 1. Example of pick-and-place problem that needs regrasping in constrained environment. DLR's robot Justin [3] has to move the horse statuette from its initial to its final configuration.

II. Related Work

This section briefly presents most closely related work on grasp planning, motion planning and manipulation planning that addresses the interdependency between the two first planning stages.

A. Grasp Planning

Early work on grasp planning does not account for finger nor arm kinematics and is often referred as contact-level techniques *e.g.* [5]. More recent works give more focus on finger or arm inverse kinematics issues [6], [7] or on smart selection of the possible hand approaches [8], [9]. Work in [7] investigates how to find grasp configurations in cluttered environments. From different object approaches, a set of stable grasps is first computed and a *grasp scoring* function is used to evaluate the grasps that are more likely to succeed the inverse kinematics and collision tests. Other recent works pay more attention on path planning for the robot base and arm [10]. Those last methods are clearly the most complete and generic as they deal with the complete pick-regrasping-and-place task. However, unlike works focusing on grasp planning, they consider simple objects or assume a given set of grasps.

Our planner includes a generic grasp planner for multi-fingered hands and determines a set of single grasps that can be used to find an exchange double grasp, even in a cluttered environments and for an object with a complex shape.

B. Motion Planning

Sampling-based planners are able today to solve complex problems in high-dimensional spaces. In particular, the probalistic methods like PRM, introduced in [11], and RRT [12] have been shown to perform well for a broad class of problems, even if their performance degrades in the presence of narrow passages. Many variants and extensions have been proposed to alleviate this problem (see [13] for a survey). Our planner is based on recent work [4], that proposes a roadmap coordination approach for multi-arm systems.

C. Manipulation Planning

One of the challenging issues of manipulation planning is to integrate the additional difficulty of planning the grasping and re-grasping operations to the path planning problem. This interdependency between path and grasp planning was first touched in early work on automatic robot programming systems (e.g. [14]). The manipulation planning approach in [15] provided a unified framework allowing to better tackle the interdependency issues between both planning levels. More recently, the *BiSpace*, algorithm [10] was proposed to plan how to go and grasp an object. The idea is to first compute a set of grasp configurations for the hand alone. Once one or more collision-free configurations for the hand are found, they become the start nodes of several RRT trees [12], that explore the hand workspace, while another RRT is grown from the robot start configuration, that explores the robot configuration space ($\mathcal{CS}$).

Our work also considers a particular instance of manipulation planning problem and focuses on the combination of efficient grasp and motion synthesis techniques to solve a pick-and-place task requiring two-hand regrasping.

III. Problem Formulation and Approach

The studied system is composed of a dual-arm robot, with a hand mounted on each arm, an object and a set of static obstacles. The problem inputs are the initial and final (goal) configurations of the robot (q_r^i and q_r^f) and of the object (q_o^i and q_o^f). q_o^i and q_o^f correspond to stable placements of the object on a support. q_o^i is such that the object is only reachable with one of the arm, referred as Arm_1. q_o^f is such that the object is only reachable with the other arm, referred as Arm_2. The robot will thus have to exchange the object between its two hands. The exchange configuration is q_o^e for the object, q_r^e for the whole robot (torso plus arms). The robot will grasp the object in q_o^i with configuration q_r^g and will place it in q_o^f with configuration q_r^p.

The method proposed in this paper for planning pick-and-place tasks relies on two sub-task planners: The Grasp planner presented in next section and the path planner presented in Section V.

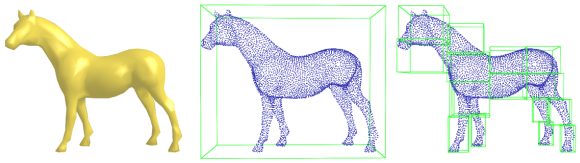

Fig. 2. The object mesh is uniformly sampled with a point set and then partitioned using a kd-tree.

IV. Grasp Planning

Grasp planning basically consists in finding a configuration for the hand(s) or end effector(s) that will allow to pick up the object. In the present context, we are interested in two kinds of grasps: One-handed grasps (or *single grasps*) and two-handed grasps (or *double grasps*). We consider only precision grasps *i.e.* contacts are made with fingertips only. This allows to reason with point contact only, that is the most common case in literature. It also gives more grasping possibilities as smaller parts can be grasped, at the cost of a weaker stability compared to power grasps. We have implemented our algorithm for the Schunk Anthropomorphic hand (SAH hand, depicted on Fig. 3). Our method is not specific to this hand but we will use it for illustration purpose.

The first stage of our grasp planner consists in building a grasp list to capture the variety of the possible grasps.

A. Single Grasp Planning

A single grasp is defined for a specific hand type and for a specific object. It is defined by:

- A *grasp frame*, *i.e.* a relative pose of the hand reference frame (*e.g.* palm or wrist frame) wrt. the object frame.
- A set of contact points (a point on the object surface and the ID of the finger realizing the contact).
- A hand configuration (finger joint parameters).

As our main concern is motion planning, it is not possible to rely on the computation of an only grasp or on a heuristic that could introduce a bias on the choice of the grasp. It is preferable to compute a grasp list that aims to reflect the best the variety of all possible grasps of the object. Our algorithm applies the following steps that will be detailed further:

- Build a set of grasp frame samples.
- Compute a list of grasps from the set of grasp frames.
- Perform a stability filter step and compute a quality score for each grasp.

1) Grasp frame sampling: To avoid biasing the possible approach of the hand when we compute the grasp, we choose to uniformly sample the possible grasp frames, by the mean of a grid. We have chosen a grasp frame that is centered on the intersection of the finger workspaces so that it is roughly centered where the contacts may occur. We set as an input the numbers of positions and orientations, each pair position-orientation defining a frame. The positions are uniformly sampled in the object axis-aligned bounding box with a step computed to fit the desired number of position samples. The orientations are computed with an incremental grid as in [16]. For each grasp frame, a set of grasps will be computed.

2) Grasp list computation: As the proposed grasp planning method does not restrict the possible hand poses or contact surface on the object, it requires a lot of computation. Therefore, we have to introduce some data structures to reduce the computation times. Except for collision test, the most expensive computation is the finger inverse kinematics. One has to be able to know the fastest possible if, for a given hand pose (relative to the object), a finger can establish a contact on the object surface and, if it can, where. The contacts can only occur in the intersection of the finger workspace and the object surface. For each finger, it is consequently crucial to find this intersection or at least an approximation. We propose to approximate the object surface with a point set. The set is obtained by a uniform sampling of the object surface. The sampling step magnitude is chosen from the fingertip radii. A kd-tree is built upon the point set in order to have a hierarchical space partition of the points (Fig. 2).

We then need to find the intersection of each finger workspace with the object kd-tree. As spheres are invariant in rotation, they are interesting to build an approximation of the finger workspace. Starting from grid approximations of the finger workspace's interior volume W and envelope E (Fig. 3), we build incrementally a set of spheres fitting inside the workspace using Algorithm 1. The interior volume grid is obtained by sampling the three joint angles over their respective ranges. The envelope grid is obtained by blocking one of the joint angle to its limit values (lower then upper) and sampling the two others. By construction, the sphere hierarchy starts from the biggest ones, corresponding to workspace parts that are the farthest to the finger joint bounds.

Algorithm 1: Finger workspace approximation

input : W= a set of points strictly inside the finger workspace ; E= a set of points on the envelope of the finger workspace ; k_{max}= the desired maximal number of spheres ; r_{min}= the desired minimal sphere radius ;
output : S= a set of spheres S_k
$S = \emptyset$; $k = 1$;
while $k < k_{max}$ **do**
  foreach $p \in W$ **do**
    $d(p) = \min_{p_i \in (E \cup S)} (\|p - p_i\|)$;
  $p_{best} = \{p \in W : d(p) = \max_{p_i \in W}(d(p_i))\}$;
  $S_k = sphere(center = p_{best}, radius = d(p_{best}))$;
  $S = S \cup S_k$;
  $W = W - \{p \in W : p \subset S_k\}$;
  $k = k + 1$;
  if $d(p_{best}) < r_{min}$ **then**
    break ;
return S ;

Once we have both the kd-tree and the sphere hierarchy, it is very fast to determine the intersection of the two sets and so the contact points. The intersection is tested from the biggest to the smallest sphere, guarantying that the "best" parts of the workspace will be tested first, *i.e.* the one farthest to workspace singularities, due to the joint bounds. For a given grasp frame, the grasp is computed finger by finger, *i.e.*, if we have the contact and configurations of the fingers 1 to $i-1$, we search a contact point for finger i and test collision only with fingers 1 to i as the other finger configurations are not yet known. We start from the thumb as no stable grasp can be obtained without it. If a finger can not establish a contact, it is left in a "rest" (stretched) configuration. if we have three contacts or more, we can proceed to the stability test. Note that, at this stage, we have a collision-free grasp *i.e.* no collision between the hand and the object and do not yet consider collision with the environment or the robot arms or body.

3) Stability filter and quality score: The stability test is based on a point contact with friction model. From the contact positions and normals, we compute a force-closure stability score [17]. All the grasps that do not verify force-closure are discarded. The stability score is not sufficient to discriminate good grasps so we build a more general quality score. Several aspects can be taken into account to compute a grasp quality measure [18]. A tradeoff is often chosen with a score that is a weighted sum of several measures. We used the following scores: the above force-closure score, the distance from the contact centroid to the mass center of the object, the sum of the angles between each contact normal and the force ellipsoid major axis of the associate finger, the sum of the object's surface inverse curvature at the contact points. The curvatures are normalized over all the values on the sampled object's surface. The weights were set manually.We found that $(1.0, 1.0, 1.0, 6.0\ (curvature\ score))$ offers a good trade-off even though it might not always be the case. Combining different quality scores remains however a hard problem, beyond the scope of this paper.

B. Double Grasp Planning

A double grasp is a grasp involving both hands. It is computed from two single grasp lists L_1 and L_2, obtained for each hand. Each single grasp pair sg_1 and sg_2, belonging to L_1 and L_2 respectively, is tested. All colliding pairs are rejected. To avoid an excessive number of pairs to test, we first filter L_1 and L_2 to remove all the grasps that lead to a collision with the environment for the given initial and final

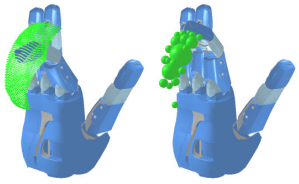

Fig. 3. The finger workspace discretized with a grid (forefinger workspace, left image). The grid is converted to a volumetric approximation as a set of spheres (right image).

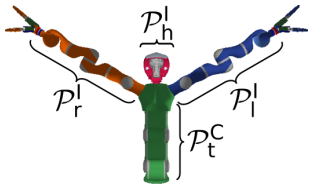

Fig. 4. The decomposition of the humanoid system into elementary parts. The two arms and the head are the "independent" parts and the torso is the "common" one.

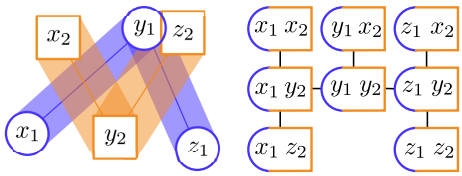

Fig. 5. On the left, a representation of elementary roadmaps computed for the presented system (circle and square). The generated *Super Graph* on the right.

object poses. For instance, all the grasps that take the object from "below" will be removed as they lead to a collision between the object support and the hand. For each double grasp, a score is then computed based on two scores: the quality of each single grasp and a robot configuration score.

- The minimum of sg_1 and sg_2 quality is used as a the first score for the double grasp.
- A robot configuration score is computed for sg_1 and sg_2, based on how "natural" is the way to grasp the object in its start and goal configuration using sg_1 and sg_2. For the double grasp, we take the minimum of the robot configuration scores of sg_1 and sg_2.

After normalizing these two scores separately for all the computed double grasps, we sum them for each double grasp to obtain its score.

V. Path Planning

A. Offline roadmap

Computing an offline roadmap is suitable to speed up the task resolution. This roadmap is computed using specially designed one for multi-arm systems [4]. During this generation, only self-collisions and collisions against static objects are considered while ignoring the object to move. The multi-arm systems roadmap composition algorithm [4] is a suitable method to efficiently plan multi-arm systems motions in constrained workspaces. It is based on the decomposition of the system into kinematically independent parts, which are treated as individual robots in a multi-robot roadmap composition approach. Fig. 4 illustrates the different parts of Justin. Each arm is independent from the other. If the value of an arm joint is modified, the change does not affect the position of any other part in the system. However, a change in one of the torso joints modifies the pose of the arms and the head. In general case, a part is said to be *independent*, if the change of its configuration does not affect the pose of other system parts. Thus, this system involves three independent parts $\mathcal{P}^I$: the right and the left arms ($\mathcal{P}^I_r$ and $\mathcal{P}^I_l$ respectively), and the head ($\mathcal{P}^I_h$); and a common one: the torso ($\mathcal{P}^C_t$). Given the relatively low mobility of the head, it can be considered together with the torso in order to simplify the system decomposition.

This decomposition permits to split the roadmap construction into two stages. The first stage is to compute two collision-free roadmaps $\mathcal{R}_r$ and $\mathcal{R}_l$ for the two sub-systems composed by the torso and the right and left arms respectively. Such roadmaps construction considers self-collisions of the sub-system and collisions with the obstacles in the workspace. Any PRM-like method can be used to generate these roadmaps. However, the use of methods generating compact roadmaps such as [19] or [20] is preferable in order to limit the size of the composite roadmap, which is defined as the Cartesian product of all the sub-system roadmaps, and whose size may become huge if standard PRM methods are used.

The constructed roadmaps are then merged into a composite one, called *Super Graph* ($\mathcal{SG}$), extending the idea initially proposed in [21] for the case of multiple car-like robots. Merging two nodes from $\mathcal{R}_r$ and $\mathcal{R}_l$ creates $\mathcal{SG}$ node. The $\mathcal{SG}$ nodes are connected via $\mathcal{SG}$ edges. Fig. 5 illustrates the principle of the $\mathcal{SG}$ construction. Creation of nodes and edges are explained below.

1) Super Graph Nodes: The $\mathcal{SG}$ nodes are created by the composition of elementary nodes x_r and x_l, in $\mathcal{R}_r$ and $\mathcal{R}_l$ respectively. Due to change of common configuration parameters in x_r and x_l, merging consists of creating two $\mathcal{SG}$ nodes, obtained by fusing configurations of the independent parts. Each $\mathcal{SG}$ node is only partially checked for self-collision, since nodes of the elementary roadmaps are collision-free with the environment. Each independent part configuration added up to an elementary node has been checked against the other independent parts, the common parts, and the workspace obstacles. Only the collision-free nodes are kept in $\mathcal{SG}$.

2) Super Graph Edges: Once a node X is created and inserted into $\mathcal{SG}$, its connection to the other $\mathcal{SG}$ nodes Y is computed. In order to preserve the efficiency of the roadmap construction, a filter, based on the information given by the elementary nodes, only considers connections between two $\mathcal{SG}$ nodes X and Y if their composing nodes, x_r and y_r, and x_l and y_l are connected in $\mathcal{R}_r$ and $\mathcal{R}_l$ respectively. Another possible strategy for saving computing time is to construct a roadmap tree instead of a graph. In this case, connection tests (using a local planner) are only performed between $\mathcal{SG}$ nodes belonging to different connected component of $\mathcal{SG}$. Like for the $\mathcal{SG}$ nodes, validating $\mathcal{SG}$ edges only requires to test collisions between pairs of parts and with workspace obstacles that have not been checked when computing the edges of the elementary roadmaps. A $\mathcal{SG}$ edge is added to the $\mathcal{SG}$ if it is collision-free.

B. Online planning

We explain below how robot motions are computed online in order to realize the pick, regrasp and place task, given the precomputed single / double grasps and roadmap. The task can be decomposed into four consecutive steps:

- Grasp the object (from q_r^i to q_r^g)
- Carry it to the exchange position (from q_r^g to q_r^e)
- Place it (from q_r^e to q_r^p)
- Goto rest configuration (from q_r^p to q_r^f)

with q_r^i and q_r^f given as input and the three other configurations (q_r^g, q_r^e and q_r^p) are generated by the planner.

Once the grasp configurations have been generated (Section V-C), the path planner executes the four previously presented queries sequentially. To obtain collision-free paths, the previously computed roadmap has to take into account the manipulated object. In fact, by adding an object in the robot working space, the collision-free $\mathcal{CS}$ of the robot changes. The $\mathcal{CS}$ also changes when the robot carries the object. Revalidating online the entire roadmap would be too costly. We use instead a lazy node and edge revalidation as in [22].

First the query is executed in the precomputed roadmap disregarding the collisions of the existing nodes and edges in the graph. Once a path is found, each local path composing it is checked for collision against the manipulated object that is not considered in the preprocessed roadmap. If a collision is detected for some local paths, replanning strategy is performed as follows. First, the collision-free configurations bounding the collision portion are determined. An alternative path between the disconnected configurations is then searched in the precomputed graph. If no solution is found, local reconnections are computed using RRT-like planners [12]. The path is iteratively modified until all its local paths are collision-free. Our implementation uses the same data structure for storing PRM roadmaps and RRT diffusion trees. Then, it is easy to enrich a graph computed using PRMs with RRTs to efficiently reconnect the disconnected components in the graph.

C. Grasp Configuration

The grasp and place configurations of the robot are simply derived from the object initial and final placement q_o^i and q_o^f provided as input. However, the exchange configuration q_o^e of the object is unknown. The object position is determined by minimizing wrist motions to perform the task. The minimization is done on a 3D grid. The object exchange position is the collision-free grid node that minimizes the length sum of the edges depicted on Fig. 6, in 3D space. Once a position

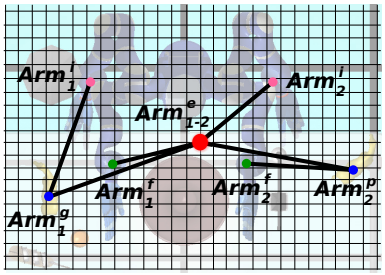

Fig. 6. Top view of a pick-and-place task showing the elementary distances to be minimized over a 3D grid in order to compute the double grasp position (the center red circle). Superscripts corresponds to the different key-configurations of the arms ([*i=initial, f=final, g=grasping, e=exchange, p=placement*]-configuration).

is selected, the object is tested against the static obstacles in predetermined orientations to ensure a collision-free object exchange. Unlike object exchange position, the orientation is determined by the selected double grasp directions and the object position wrt. the robot torso.

The grasp, place and exchange configurations are generated in nearly the same way. All robot joints are sampled except the arm(s) grasping the object (one arm for grasp and place configurations and both for exchange configuration), that are computed using the inverse kinematics characterized by the grasps and the object position. Then, a collision test is performed on the generated configuration. For exchange configuration, the object configuration is sampled following a Gaussian distribution centered on the theoretical best q_o^e previously computed.

The robot grasp configuration score used in the double grasp scoring formula, takes into account grasping and free-arm configurations. The grasping arm score is determined with cosine of the angle between the selected grasp and the object robot-base directions. This will give a bad score for grasps whose direction is far from the object-(robot base) axis. The free-arm score is added to favor "natural" robot configurations. This score is composed of the joint distance between the sampled arm configuration and a user defined rest configuration of the arm. It is also composed by the height difference between the sampled and rest configurations, and the distance between the arm wrist and the plane composed by the robot's torso and shoulders.

VI. Results

To evaluate the performance of the planner, we propose to plan a pick-and-place task with Justin [3], equipped with two SAH hands (Fig. 1). Justin is composed of a 3-DoFs torso, two 7-DoFs DLR-Lightweight-Robot-III arms, and a 2-DoFs head. The SAH hands are composed of four 3-DoFs fingers plus a movable thumb base. Disregarding the neck joints, which are considered to be fixed in our experiments, Justin involves 43-DoFs. The object to work with is a highly non-convex body with several parts (a horse statuette, whose 3D model (widely used in CG community) has been simplified to 672 vertices and 1334 triangles).

In this task, Justin has to pick the object at his right and place it behind the desk lamp at his left. The legged lamp and the vase constrain the robot motions and the exchange configuration. As the single grasp generation is

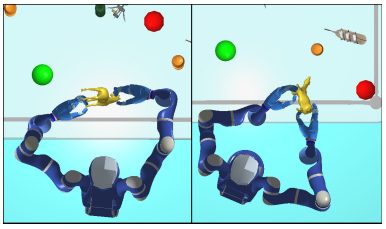

Fig. 7. Example of computed exchange positions for different initial (red) and final (green) object poses, taking into account the cost to minimize and obstacle collision avoidance.

TABLE I
NUMERICAL RESULTS

Problem	Grasp		Carry		Place		Rest	
	n	T(s)	n	T(s)	n	T(s)	n	T(s)
Online Planning	65	2.7	53	2.6	32	1.2	49	1.9
Path Validation	3	0.3	18	1.5	12	1.1	3	0.3
Total	68	3	71	4.1	44	2.3	51	2.1

a deterministic and workspace independent operation, the same single grasps lists (one for each hand) are used for all tests. Right grasp list contains 17 valid grasps and 22 for left list, each one computed in about 1 minute[1]. In the presented scenario, given the single grasp lists, the planner generates 4 (right) and 7 (left) collision-free single grasps configurations in 4.2 seconds, and 4 double-grasp configurations in 2.6 seconds. Fig. 7 shows two examples of computed double grasps, used for regrasping. They are computed for the same object but for different initial and final object poses. The computed exchange configuration brings the object near the robot torso, leading to a motion that looks more "natural" than a simple straight line linking initial and final poses.

Table I reports the numerical results (n= node number, T= computation time) obtained for the presented pick-and-place planner, following the task decomposition presented in Section V-B. The offline roadmap is computed in about 5 minutes and contains near 1700 nodes. The nodes are produced by merging two elementary roadmaps generated using Vis-PRM algorithm [19], each one containing 50 nodes. Using this offline roadmap, our planner solves the entire task in 11.6 seconds. Comparatively, by putting a single query planner [12], one needs 35 seconds to solve the same problem. These two last results do not include the time needed to compute single and double grasps configurations.

Table I also shows that the time consumed in path validation for Carry and Place phases is more important than for the two other planning phases. This is due to the bigger change of the robot $\mathcal{CS}$, induced by considering the object as a robot's body instead of just a static obstacle. However, the time needed to plan (without revalidation) Grasp and Rest phases are higher because of the highly constrained grasp and place configurations.

VII. CONCLUSION AND FUTURE WORK

We have presented a planner that can automatically compute the motion of a dual-arm/hand robot during a pick-and-place task requiring an object exchange between the hands. The planner computes as well all the necessary intermediate configurations. The integration of several offline computed and reusable data structures such as grasp lists and arm roadmaps, allows the planner to significantly reduce its computation times compared to the use of simple single-query techniques. Simulation results show the efficiency of the planner for solving a difficult manipulation task involving a humanoid robot equipped with two redundant arms and two multi-fingered hand, a complex-shaped object and a cluttered environment.

In some situations, no solution may be found by the planner because the initial pose of the object constrains too much the choice of the single grasp used to pick the object up, that in turn constrains the choice of the exchange double grasp, constraining in turn the choice of the single grasp used to place the object in its final configuration. To treat such a case, one or more intermediate placements are mandatory. A more complex version of our planner could try to integrate such notions.

[1]All numerical results in the paper have been averaged over 20 runs of the planner. Computing time corresponds to a Dual-Core AMD Opteron processor 2222 at 3.0 GHz

REFERENCES

[1] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," *IEEE Conf. on Rob & Autom.*, vol. 2, pp. 945–952, 1994.

[2] N. Vahrenkamp and et al., "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, October 2009.

[3] C. Ott and et al., "A humanoid two-arm system for dexterous manipulation," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2006.

[4] M. Gharbi, J. Cortés, and T. Siméon, "Roadmap composition for multi-arm systems path planning," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Oct. 2009, pp. 2471–2476.

[5] V.-D. Nguyen, "Constructing force-closure grasps," *IEEE Conf. on Rob. & Autom.*, vol. 3, pp. 1368–1373, Apr 1986.

[6] Z. Xue, J. Marius Zoellner, and R. Dillmann, "Grasp planning: Find the contact points," *IEEE Int. Conf. on Robotics and Biomimetics*, pp. 835–840, Dec. 2007.

[7] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, "Grasp planning in complex scenes," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids07)*, December 2007.

[8] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," *IEEE Conf. on Rob. & Autom.*, vol. 2, pp. 1824–1829 vol.2, Sept. 2003.

[9] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum volume bounding box decomposition for shape approximation in robot grasping," *IEEE Conf. on Rob. & Autom.*, pp. 1628–1633, May 2008.

[10] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, "Bispace planning: Concurrent multi-space exploration," in *Robotics: Science and Systems*, vol. IV, Zurich, Switzerland, June 2008.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Rob. & Autom.*, vol. 12(4), pp. 566–580, 1996.

[12] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Proceedings of Workshop on the Algorithmic Foundations of Robotics*, 2000.

[13] S. M. LaValle, *Planning Algorithms*. New York: Cambridge University Press, 2006.

[14] T. Lozano-Pérez, J. L. Jones, P. A. O'Donnell, and E. Mazer, *Handey: a robot task planner*. Cambridge, MA, USA: MIT Press, 1992.

[15] T. Siméon, J. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The Int. J. of Robotics Research*, vol. 23, iss. 7-8, pp. 729–746, August 2004.

[16] A. Yershova and S. LaValle, "Deterministic sampling methods for spheres and so(3)," in *IEEE Int. Conf. Robot. & Autom.*, 2004.

[17] B. Bounab, D. Sidobre, and A. Zaatri, "Central axis approach for computing n-finger force-closure grasps," *IEEE Conf. on Rob. & Autom.*, pp. 1169–1174, May 2008.

[18] R. Suarez, M. Roa, and J. Cornella, "Grasp quality measures," in *Universitat Politecnica de Catalunya (UPC), Technical Report*, 2006.

[19] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics Journal*, vol. 14(6), pp. 477–494, 2000.

[20] L. Jaillet and T. Siméon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *Int. J. of Robotics Research*, vol. 27, no. 11-12, pp. 1175–1188, 2008.

[21] P. Svestka, "Robot motion planning using probabilistic roadmaps," Ph.D. dissertation, Universiteit Utrecht, 1997.

[22] L. Jaillet and T. Siméon, "A prm-based motion planner for dynamically changing environments," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, vol. 2, Sept. 2004, pp. 1606 – 1611 vol.2.