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. This grasp change can be realized by first placing the object in the shared workspace of the two arms then picking it with the other arm and achieving the task. However, a more efficient way to change the grasp is to plan a dual-hand grasp of the object. Moreover, such two-hand regrasping strategy results in a more human-like behavior and does not require computing stable positions of the object in the shared workspace.

Among the earlier work on this topic, [1] proposed a practical planner for PUMA 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. The DLR's Justin robot equipped with two Schunk Anthropomorphic 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 manipulated 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 [3] 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 planing queries and determine the grasp exchange position (Section V-B).

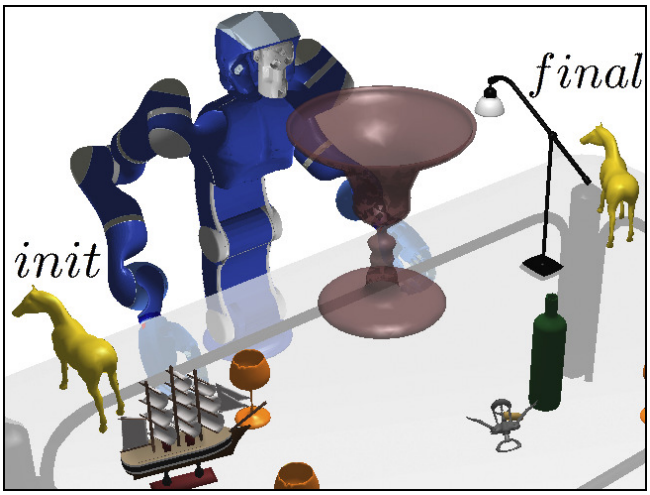

Fig. 1. Example of pick-and-place problem that needs regrasping in constrained environment. The robot has to move the horse statuette from its initial to its final configuration.

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 [4], [5], with corresponding stability criteria [5], [6]. More recent works integrate considerations on finger and/or arm kinematics in order to favor directions when searching for the set of possible grasps, based on a primitive shape decomposition of the object (see [7], [8], [9]). Other work give more focus on finger or arm inverse kinematics issues ([10], [11]). Work in [11] investigates how to find

grasp configurations in cluttered environments, for a given robot base position in the object range. 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 gives more focus on path planning for the robot base (or body) and arm [12]. 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 a complex shaped object.

B. Motion Planning

Sampling-based planners are able today to solve complex problems in high-dimensional spaces. In particular, the PRM approach introduced in [13] and further developed in many other works (see [14], [15] for a survey) has been shown to perform well for a broad class of problems, even if its performance degrades in the presence of narrow passages. A number of variants and extensions have been proposed to alleviate this problem, e.g. biasing sampling around obstacles [16], [17], [18] or towards the medial axis [19], using free-space dilatation [20], [21], visibility-based filtering [22] exploiting search space information [23], or delaying collision detection [24], [25]. Our planner is based on recent work [3], 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 planning and grasp planning was first touched in early work on automatic robot programming systems (e.g. the Handey system [26]). The manipulation planning approach in [27] provided a unified framework allowing to better tackle the interdependency issues between both planning levels. This framework was further developed in several other works yielding to practical manipulation planners (e.g. [28], [29], [30], [31]). It was extended in [1] to multi-arm manipulation with a practical planner relying on several simplifications, but able to deal with complex and realistic problems. More recently, the *BiSpace*, algorithm [12] 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 RRTs tree [32], that explore the hand workspace, while another RRT is grown from the robot start configuration, that explores the robot configuration space.

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 robot equipped with two arms, with a hand mounted on each arm, of an object and of a set of static obstacles referred as the environment. The problem inputs are the initial and final (goal) configurations of the robot (q_{robot}^{init} and q_{robot}^{final}) and of the object (q_{object}^{init} and q_{object}^{final}). q_{object}^{init} and q_{object}^{final} correspond to stable placements of the object on a support (*e.g.* a table). q_{object}^{init} is such that the object is only reachable with one of the arm. Let us refer to it as arm 1. q_{object}^{final} 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_{object}^{exch} for the object, q_{robot}^{exch} for the whole robot (base or torso plus arms). The robot will grasp the object in q_{object}^{init} with configuration q_{robot}^{grasp} and will place it in q_{object}^{final} with configuration q_{robot}^{place}.

The method proposed in this paper for planning pick-and-place tasks with regrasping involves two different topics. First one is the grasp planner (see Section IV). Picking and regrasping the object, needs a grasp planner able to find hands configurations that are stable and collision-free. The grasps have also to satisfy the robot kinematic constraints. Section IV-A explains how the hands configurations are computed and ordered given some criterion whereas Section IV-B shows how the double grasps are filtered and ordered. The Second one is the path planning (see Section V) that will produce collision-free motion of the robot to achieve the pick-and-place task. For this, an offline roadmap is computed (see Section V-A) to speed up the online planning phase detailed in Section V-B.

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 (referred later as single grasps) and two-handed grasps (referred later as 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 even small 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 (SAHand, 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, to find an exchange double grasp even in cluttered environment for an object with a complex shape.

A. Single Grasp Planning

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

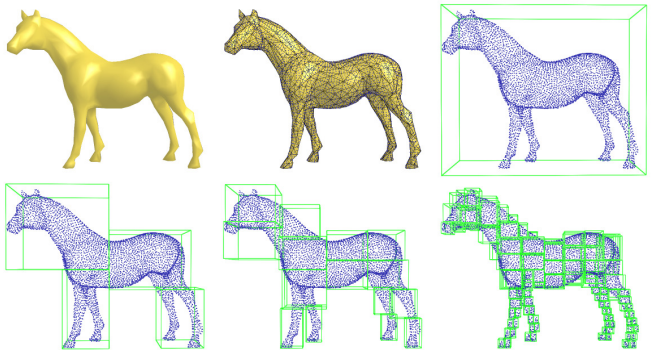

Fig. 2. The object mesh is uniformly sampled with a point set (top images). The point set is then partitioned using a kd-tree (bottom images).

- A relative pose of the hand reference frame (*e.g.* palm or wrist frame) with respect to the object frame. We refer to this frame as *grasp frame*.
- A set of contact points (a point on the object surface and the ID of the finger realizing the contact).
- A configuration of the hand (a set of finger joint parameters).

As our main concern is motion planning, it is not possible to rely on the computation of an only grasp or to compute grasps according to 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.
- Compute a quality score for each grasp.

1) Grasp frame sampling: For manipulation planning, it is important to avoid biasing the possible approach of the hand when we compute the grasp. Therefore, we choose to sample the possible grasp frames uniformly. This is done by the mean of a grid. We have chosen, for our hand, 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 number of positions and the number of orientations, each couple 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 like the one in [33]. 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 surface of contact 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 specified hand pose (relative to the object), a finger can establish a contact on the object surface and, if it is the case, 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 radius. 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 surface kd-tree. As spheres are invariant in rotation, they are interesting to build an approximation of the finger workspace. Starting from a grid approximation of the finger workspace (Fig. 3), we build incrementally a set of spheres fitting inside the workspace. First, points of the grid are marked as being boundary points (on the workspace envelope) or inner points (strictly inside the workspace volume). For each inner point, the smallest distance to the boundary points is computed, referred as $dmin$. The inner point having the biggest $dmin$ is the center of the first sphere S_1, of radius $dmin$. For all the inner points that are not inside S_1, a new $dmin$ is computed, that is the minimum of the old $dmin$ and the minimal distance to S_1. The point that has the biggest $dmin$ is the center of the second sphere S_2, of radius $dmin$. This process is repeated until we have reached the maximal desired sphere number or the last computed sphere has a radius less than a specified threshold. We keep the ordering of the construction so that the sphere hierarchy starts from the biggest ones, corresponding to workspace parts that are the farthest to the finger joint bounds. These bounds were first slightly reduced (Fig. 3) in order to eliminate configuration where the fingers are close to completely stretched.

Once we have both the kd-tree and the sphere hierarchy, it is very fast and easy 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 singularities due to the joint bounds. For a given grasp frame, the grasp is computed finger by finger, that means that, 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 the 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)

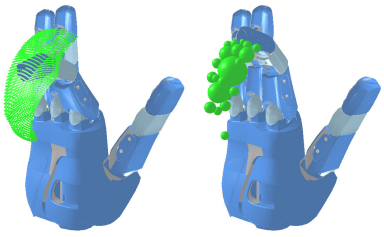

Fig. 3. The finger workspace discretized with a grid (forefinger workspace, left image). The grid is converted to a volumetric approximation as a set of spheres (right image).

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, that explains why at least three contacts are required. From the contact positions and normals, we compute a stability score. It is based on a force closure test and stability criterion [34]. All the grasps that do not verify force-closure are discarded. We also compute and add a second score that is the distance to the mass center of the object. 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 [35]. A tradeoff is often chosen with a score that is a weighted sum of several measures. We chose to combine the previous stability criterion with two other criteria: A finger force ellipsoid major axis score and a contact curvature score. The idea behind the first one is that it is preferable to favor contact such that the contact normal is in a direction close to the direction of the major axis of the force ellipsoid, corresponding to the better force transmission ratio. The curvature score is used to favor contacts where the mean curvature of the object surface is low. In real situation, it will reduce the impact of a misplaced contact as the contact normal will be susceptible to smaller change in a low curvature area than in a high curvature one.

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 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 (*e.g.* a table) 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 stability score for the double grasp.
- A robot configuration score is computed for sg_1 and sg_2. It is 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.

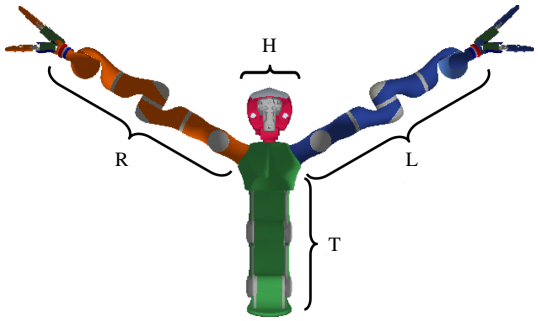

Fig. 4. The decomposition of the humanoid system into elementary parts. The two arms and the head are the "independent" parts and the torso is the "common" one.

V. Path planning

A. Offline roadmap

Computing an offline roadmap is suitable to speed up the pick-and-place task resolution. This roadmap is computed using specially designed one for multi-arm systems [3]. 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 [3] 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 Vis-PRM [22] or PDR [36] 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 [37] for the specific case of multiple

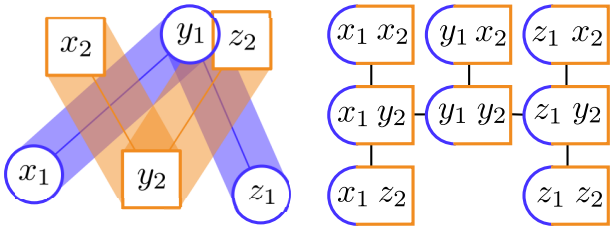

Fig. 5. On the left, an representation of elementary roadmaps computed for the presented system (circle and square). The generated *Super Graph* on the right.

car-like robots. Merging two nodes from $\mathcal{R}_r$ and $\mathcal{R}_l$ respectively creates $\mathcal{SG}$ node. The $\mathcal{SG}$ nodes are connected via a $\mathcal{SG}$ 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 independents 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_{robot}^{init} to q_{robot}^{grasp})
- Carry it to the exchange position (from q_{robot}^{grasp} to q_{robot}^{exch})
- Place it (from q_{robot}^{exch} to q_{robot}^{place})
- Goto rest configuration (from q_{robot}^{place} to q_{robot}^{final})

with q_{robot}^{init} and q_{robot}^{final} given as input and the three other configurations (q_{robot}^{grasp}, q_{robot}^{exch} and q_{robot}^{place}) are generated by the planner.

Once the grasp configurations have been generated (see 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 configuration space of the robot changes. The configuration space 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 [24], [25], [38].

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 [32], [39]. 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_{object}^{init} and q_{object}^{final} provided as input. However, the exchange configuration q_{object}^{exch} of the object is unknown. The object position is determined by minimizing wrist motions to perform the task. Fig. 6 shows the elementary distances to be minimized. The distance minimization is done on a 3D grid. The object exchange position is the collision-free grid node that minimizes $(ig + ge + ef + ie + ep + pf)$. Once a position

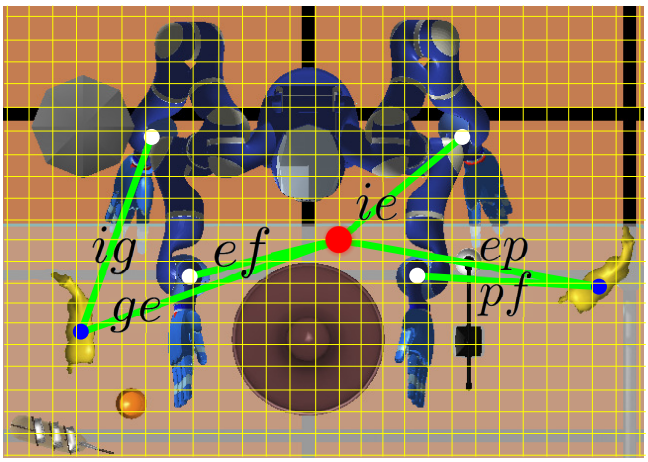

Fig. 6. Top view of a pick-and-place task showing the elementary distances to be minimized over a 3D grid in order compute the double grasp position (the center red circle)

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. After, 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_{object}^{exch} 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 that directions is far from the object robot-base axis. The free arm score is added to favor "natural" robot configurations. This score is composed by 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 plan composed by the robot torso and the shoulders.

VI. Results

To evaluate the performance of the planner, we propose to plan a pick-and-place task with the robot Justin [40] equipped with two SAHands, depicted on Fig. 1. Justin is a humanoid torso composed of two 7 DoF DLR-Lightweight-Robot-III arms mounted on a 3 DoF torso and a head with a 2 DoF neck. The SAHands hands are composed by four 3 DoF fingers and a movable thumb base. Disregarding the neck joints, which are considered to be fixed in our experiments, Justin involves 43 DoF. 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).

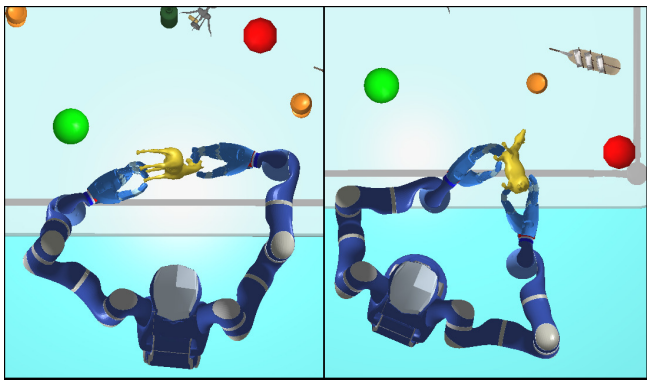

Fig. 7. Example of computed exchange positions for different initial (red) and final (green) object poses, taking into account the cost to minimize and obstacle collision avoidance.

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 constraint the robot motions and the exchange configuration. As the single grasp generation is a deterministic and workspace independent operation, same single grasps lists (one for each hand) are used for 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 and for each test, 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 (shown as red and green spheres). As it can be seen, 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 obtained for the presented pick-and-place planner detailed following the task decomposition presented in Section V-B (Grasp, Carry, Place and go to Rest). 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 [22] algorithm, 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 [32] instead of the presented, one needs 35 seconds to solve the same pick-and-place problem. Theses two last results does 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 are more important than for the two other planning phases. This is due to the biggest change of the robot configuration space, introduced by grasping the object than just adding it as 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

[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

TABLE I

NUMERICAL RESULTS

Problem		Online Planning	Path validation	Total
Grasp	n_{nodes}	65	3	68
	T (sec)	2.7	0.3	3
Carry	n_{nodes}	53	18	71
	T (sec)	2.6	1.5	4.1
Place	n_{nodes}	32	12	44
	T (sec)	1.2	1.1	2.3
Rest	n_{nodes}	49	3	51
	T (sec)	1.9	0.3	2.1

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.

VIII. ACKNOWLEDGMENTS

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.

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] 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.

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

[5] C. Ferrari and J. Canny, "Planning optimal grasps," *IEEE Conf. on Rob. & Autom.*, pp. 2290–2295 vol.3, May 1992.

[6] A. Bicchi, "On the closure properties of robotic grasping," *Int. J. Rob. Res.*, vol. 14, no. 4, pp. 319–334, 1995.

[7] 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.

[8] C. Goldfeder and et al., "Grasp planning via decomposition trees," *IEEE Conf. on Rob. & Autom.*, pp. 4679–4684, April 2007.

[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] 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.

[11] D. Berenson and et al., "Grasp planning in complex scenes," in *IEEE-RAS Int. Conf. on Humanoid Robots*, December 2007.

[12] 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.

[13] 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.

[14] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge: MIT Press, 2005.

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

[16] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Robotics: The Algorithmic Perspective (WAFR)*, P. Agarwal, L. E. Kavraki, and M. Mason, Eds., 1998.

[17] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *IEEE Int. Conf. Robot. & Autom.*, 1999.

[18] D. Hsu, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *IEEE Int. Conf. Robot. & Autom.*, 2003.

[19] S. Wilmarth, N. M. Amato, and P. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *IEEE Int. Conf. Robot. & Autom.*, 1999.

[20] D. Hsu, H. Cheng, and J.-C. Latombe, "Multi-level free-space dilation for sampling narrow passages in PRM planning," in *IEEE Int. Conf. Robot. & Autom.*, 2006.

[21] M. Saha, J.-C. Latombe, Y.-C. Chang, and F. Prinz, "Finding narrow passages with probabilistic roadmaps: The small-step retraction method," *Auton. Robots*, vol. 19, no. 3, pp. 301–319, 2005.

[22] 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.

[23] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Robotics: Science and Systems*, 2005.

[24] R. Bohlin and L. Kavraki, "Path planning using lazy prm," in *IEEE Int. Conf. Robot. & Autom.*, vol. 1, 2000, pp. 521 –528 vol.1.

[25] G. Sanchez and J.-C. Latombe, "On delaying collision checking in prm planning: Application to multi-robot coordination," *Int. J. of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

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

[27] R. Alami, T. Siméon, and J.-P. Laumond, "A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps," in *Int. sym. on Robotics research*. Cambridge, MA, USA: MIT Press, 1990, pp. 453–463.

[28] J. Ahuactzin, K. Gupta, and E. Mazer, *Manipulation planning for redundant robots: a practical approach*, K. Gupta and A. D. P. (Eds), Eds. J. Wiley, 1998.

[29] C. L. Nielsen and L. E. Kavraki, "A two level fuzzy prm for manipulation planning," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2000.

[30] 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.

[31] M. Stilman, J. Schmburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *IEEE Int. Conf. Robot. & Autom.*, 2007.

[32] 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.

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

[34] 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.

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

[36] 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.

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

[38] L. Jaillet and T. Simeon, "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.

[39] J. Cortés, L. Jaillet, and T. Siméon, "Disassembly path planning for complex articulated objects," *IEEE Trans. on Robotics*, vol. 24, no. 2, pp. 475–481, April 2008.

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