On the Problem of Moving Objects with Autonomous Robots: a Unifying High-Level Planning Approach

Hamal Marino[1,2], Mirko Ferrati[1], Alessandro Settimi[1,2], Carlos Rosales[1], and Marco Gabiccini[1,2,3]

***Abstract*—Moving objects with autonomous robots is a wide topic that includes single-arm pick-and-place tasks, object regrasping, object passing between two or more arms in the air or using support surfaces such as tables and similar. Each task has been extensively studied and many planning solutions are already present in the literature.**
In this paper we present a planning scheme which, based on the use of pre-defined elementary manipulation skills, aims to unify solutions which are usually obtained by means of different planning strategies rooted on hard-coded behaviors. Both robotic manipulators and environment fixed support surfaces are treated as end-effectors of movable and non-movable types, respectively. The task of the robot can thus be broken down into elementary building blocks, which are end-effector manipulation skills, that are then planned at the kinematic level. Feasibility is ensured by propagating unforeseen low-level failures at the higher level and by synthesizing different behaviors. The validity of the proposed solution is shown via experiments on a bimanual robot setup and in simulations involving a more complex setup similar to an assembly line.

***Index Terms*—Manipulation Planning; Cooperative Manipulators; Dual Arm Manipulation**

I. Introduction

INDUSTRIAL and service robots are often used to perform object moving tasks, using different strategies depending on the specific application. Behaviors shaped on the objects characteristics, typical of an industrial environment, are not suitable for a generic approach. As a matter of fact, exploiting all the robot capabilities like using one or more arms, passing the object between them, or using the environment to place the object and re-grasp it, is necessary to adapt to the large variety of possible situations.

Our work focuses on multi-robot coordinated manipulation as defined in [1]. Multiple end-effectors may interact with the same object during the task, either subsequently or at the same time. In this paper, a novel approach to perform object moving that can handle multi-arm robots is reported.

Manuscript received: August, 30, 2015; Revised November, 2, 2015; Accepted January, 6, 2016.

This paper was recommended for publication by Editor Antonio Bicchi upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by the European commission projects PaCMan EU FP7-ICT 600918 and by the grant no. 645599 "SoMa" -Soft-bodied intelligence for Manipulation- within the H2020-ICT-2014-1 program.

[1] Centro di Ricerca "E. Piaggio", University of Pisa, 56122 Pisa, Italy.
Corr. auth.: hamal.marino@centropiaggio.unipi.it
[2] Dep. of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova
[3] DICI, University of Pisa, 56122 Pisa, Italy

Digital Object Identifier (DOI): see top of this page.

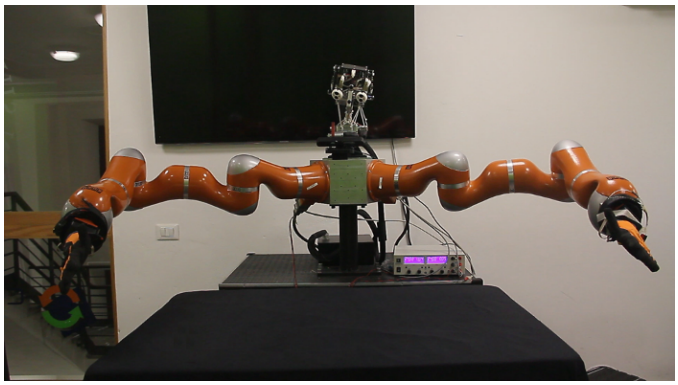

Fig. 1: Vito, the robot used to perform the reported experiments. Two Kuka Light-Weigth Robots (LWR) are mounted on a fixed torso and are equipped with a left and a right Pisa-IIT Soft-Hands. An Asus Xtion Pro-Live is used for vision.

A vision software recognizes the object present in the scene, then an operator selects the final goal for the object through a user interface. A high-level plan is generated on a suitable graph representation built using information stored in a database (DB), and it is then translated into a Cartesian plan that will be executed by the robot. The planning phase is capable of finding different strategies (pick-and-place with one or multiple arms, handoff, regrasp) and, exploiting the environment (e.g. using support surfaces), to move the object from the starting to the final position.
It is worth remarking that we do not use any *a priori* fixed strategies: instead, the strategy emerges from scratch as the outcome of the high level planning phase.

Manipulating objects usually involves a series of action in which objects are recognized, grasped, moved, and released. The simplest robot configuration that can perform these actions is a single arm in a single support surface setup, where an object is grasped, moved, and released in a different position. In such single arm setups, grasp planning, object recognition in a cluttered environment, optimization of the low-level planning, and optimization of the order of objects displacements have been the focus of recent research: [2], [3], [4], [5]. As described in [6], the exploration of a graph whose nodes represent a combination of grasp type and object position produces a sequence of multiple grasp-move-release actions, so that the same end-effector can move an object in a constrained environment, where a single pick-and-place strategy would fail.

Dual arm robot platforms gained an increasing interest because of their larger workspace and number of DoFs, but also

because of their similarity with humans, thus their capability of replacing them in already established assembly lines.
These platforms are capable of performing handoff-based manipulations: a sequence of grasp, move, handoff to the second arm, move, and release.
Planning algorithms that use handoff can be found in [7], [8], or even devising the grasp trajectories online in [9], [10]. An alternative to handoff is using a common support surface for passing an object: this approach can be interpreted as a connection between two single arms pick-move-release strategies, but it also poses some additional constraints on the kinematics and the collisions between the robots, the object and the surface. Approaches mainly focused on solving these motion planning issues are [11], [12], [13], [14].

As more robots or objects are added, a coordinated planning becomes the main requirement for such platforms.
An effort to handle this increased complexity is the integration of a semantic/symbolic reasoning into a manipulation planning, in order to perform a single task, such as making a pancake [15], moving cylinders on a cluttered table [16], solving a Tower of Hanoi [17].

Other works like [18], [19] focus mostly on reasoning, using formal languages, to plan a task by using spatial and temporal knowledge, assuming that the motion planning and grasping are handled in a perfect way.

A two-level approach similar to ours has already been presented in [20] and more recently extended in [21]: however, even if the approach in [21] may cope with the specific problem in our context, most of the experimental results reported there in manipulation do not include grasping capabilities of the robot (the robot can only push the object on a table). In order to include such capabilities, some challenges regarding the continuous modelling of grasping would arise, which we decided to handle by discretizing object grasps and, similarly, object passing poses; moreover, exploitation of support surfaces would require an adequate formalism in order to find feasible poses for an object lying on a support surface. We believe that the ability to cope with full grasping sequences with an anthropomorphic underactuated robotic hand and the backtracking ability of our method represent non trivial features of our work with respect to existing approaches.

Since our algorithm's main inputs are the starting and desired positions of an object, an autonomous procedure can be considered to provide such inputs instead of a human operator. For example, [2] and [18] deal with reordering a set of objects in order to access one of them obstructed by others, automatically resulting in a sequence of single object pick-move-place actions equivalent to multiple selections of target by the user in our framework.

Our main contribution with this paper is a graph-based modelization of the problem, associated to a novel planning algorithm, that effectively unifies previously described strategies. In this graph, each node represents the *state* of the object to be moved, while the arcs correspond to feasible *change of state* (more details on this are given in Sec. IV). In a hierarchical planning layered architecture, our planner needs to be placed below a symbolic reasoning planner as the ones cited above. The experimental validation of the proposed approach has been carried out on the Vito robot of Centro di Ricerca "E. Piaggio" (see Fig. 1). In order to test the planner on a more complex setup with multiple end-effectors, a simulation with five Kuka LWR and one conveyor belt has been also performed.

II. Problem Statement

The problem we are tackling deals with providing a robot the ability to autonomously answer the request: "move that object from the initial pose to a desired target pose", given any number of end-effectors (e.e.) and support surfaces.

Instead of pre-specifying a particular strategy to achieve this result, we only build a set of correspondences between end-effectors, objects, and grasps. Given this set, a task planning is executed that results in a feasible sequence of actions (*move*, *grasp*, and *ungrasp*) in Cartesian space to be performed in order to reach the goal.
A sample-based, low-level planner then finds collision-free joint space trajectories for the robot in order to obtain the desired motions.

In case of low-level planning failures, a recover procedure takes place in order to activate a new task planning, trying to find a different feasible sequence of actions.

Our working assumptions are that:

- a vision system provides all necessary information about the current state of the environment to the planner, handling object recognition and pose estimation with sufficient accuracy[1];
- all objects appearing in the scene can either be grasped or considered as fixed obstacles;
- a grasp database, to be provided to the algorithm, encodes information related on how each object can be grasped by each end-effector;
- information about the desired target configuration is provided by the user. Specifically, the final position and orientation of the object in Cartesian space and the desired final end-effector supporting it (any robot e.e. or support surface) have to be specified using the GUI (see Fig. 2).

III. Definitions and Database Generation

To perform the high-level task planning, the devised algorithm relies on fundamental information stored in a database.
We will now define the concepts used by the planner and how the information related to those concepts is generated and stored in the database. A very simple, exemplary database is shown in Fig. 3. We store geometric information about the support surfaces, approximations of each arm reachable region, the grasps associated between an object and an end-effector, and the feasible grasps that can be used simultaneously by two end-effectors during an hand-off.
In particular, grasp information about the end-effector poses

[1]We will not describe the pose estimation vision system here, as the implementation can be found in [22].

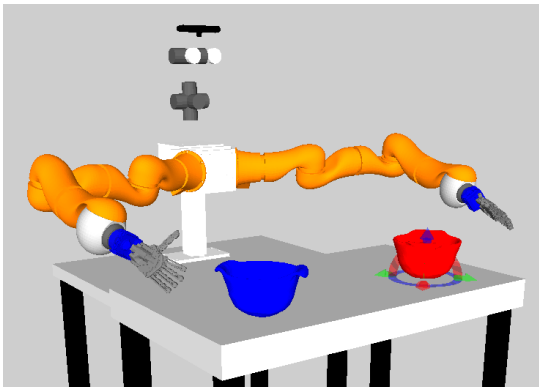

Fig. 2: RViz visualization. The blue object (left) represents the initial object configuration given by the vision system; the red object (right) represents the goal configuration selected by the user. For a picture of the whole User Interface see attached multimedia file.

w.r.t. the objects can be generated automatically for fully actuated hands (as in e.g. [23], [24]) or manually, like we did, using a motion capture system.

Objects		
Obj_id	3D views	Name
o1	PointClouds	Cylinder

End-Effectors			
EE_id	Reachable Workspaces	Movable	Name
e1	w1,w2,w3	FALSE	Table
e2	w1,w2	TRUE	LeftArm
e3	w2,w3	TRUE	RightArm

Workspaces			
Workspace_id	Adjacency	Geometry	Name
w1	w2	box1	Left
w2	w1,w3	box2	Middle
w3	w2	box3	Right

Grasp_id	EE_id	Name	Allowed_transitions
g1	e1	TopTable	g5 , g6 , g8 , g9
g2	e1	SideTable	g5 , g8
g3	e1	BottomTable	g4 , g5 , g7 , g8
g4	e2	TopLeftArm	g3, g8 , g9
g5	e2	SideLeftArm	g1 , g2 , g3 , g7, g9
g6	e2	BottomLeftArm	g1 , g7 , g8
g7	e3	TopRightArm	g3 , g5 , g6
g8	e3	SideRightArm	g1 , g2 , g3 , g4 , g6
g9	e3	BottomRightArm	g1 , g4 , g5

Fig. 3: Example of a database overall structure. Objects, End-Effectors, Workspaces, and Grasps tables are reported.

A. Objects

All (known) objects in a given set are included in the database; for each of them, we store multiple 3D views that are used for object recognition and pose estimation using a 3D vision library ([22]) to gather the necessary start state information before the planning starts.

B. End-Effectors

An *end-effector* is an entity which can act/apply a grasp/support on an object: there are both "movable" end-effectors (such as robot hands connected to arms) and "non-movable" ones (such as a fixed surface in the environment which can be exploited for statically stable object support). From a task planning perspective, there is no distinction between movable and non-movable end-effectors, as both types can interact with an object in specific ways, and can exchange the object with specific interaction transitions.

C. Workspaces

Our *environment* is the union of all the points reachable by all the end-effectors (movable and non-movable). For fixed surfaces (which are non-movable end-effectors), reachable points are represented by thin normal extrusions of the surfaces themselves.

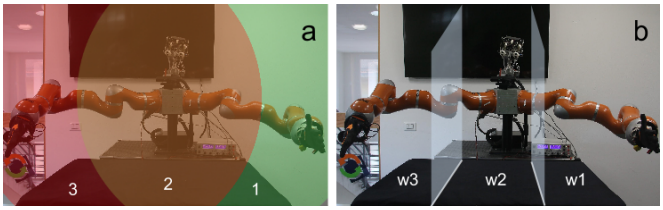

Fig. 4: (a) The reachable regions of the left and right arm are respectively reported in green and red, while the one of the table is represented by the table itself. (b) Using the intersection between simple approximations of these regions (cuboids), we defined for this scenario *w1*, *w2*, and *w3*.

In order to split the environment into regions useful for the high level planning, we approximate the reachable region of each end-effector with simpler convex models. We use cuboids because they are a natural 3D extension of support surfaces. We then check for intersections in order to find regions where multiple end-effectors can interact with each other. These intersections are called *workspaces*.

Workspaces are candidates for object *passing*, since they may be reached by more than one end effector simultaneously. Workspaces enjoy the additional property of *adjacency* if they can be reached by the same movable end-effector. Thus, adjacent ones are candidates for object *moving*.

With reference to Fig. 3 (tables) and Fig. 4(b), since workspaces *w1* (left workspace) and *w2* (middle workspace) are adjacent (check Workspaces table, rows 1 and 2), and also reachable by the movable e.e. *e2* (left arm), (check End-Effectors table, row 2), object *o1* (Cylinder) can be moved between these two workspaces. This allows to affirm that the Cylinder can be moved between the left and the middle workspace using the left arm with the grasps from *g4* to *g6* (check Grasps table, rows with *e2* in EE_id column).

Reachability information between end-effectors and workspaces is stored in End-Effectors table.

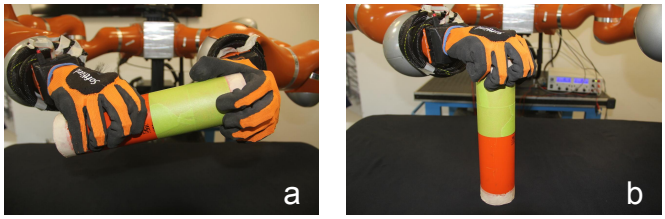

Fig. 5: (a) transition between Top-LeftArm and Side-RightArm grasps (g4-g8). (b) transition between Bottom-Table and Top-RightArm grasps (g3-g7).

D. Grasps

In this work, a *grasp* is defined as the relative configuration of the end-effector and a specific object once it is grasped. This Cartesian information is not shown in Fig. 3 for better readability. We associate to each object of our database one or more grasps.

If an object has to be handed off from an end-effector to another, there will be an instant in time where both e.e. are grasping the object.
A *transition* between two grasps is an interaction primitive that has no kinematic overlap between the two different end-effectors performing the grasps at the same time, i.e. there is no collision between the e.e.'s when they both are in their final grasp configuration with respect to the object.

The transitions we consider in this work are used by the high-level planner without any further distinction, but we can interpret them in the following way:

- pick and place actions when a non-movable end-effector is involved together with a movable one,
- a sequence of grasp-ungrasp actions when both end-effectors are movable.

No transition is allowed between any two non-movable end-effectors.

As an example, with reference to Fig. 3, End-Effectors table, *e1* (Table) and *e3* (RightArm) can both reach *w2* (Middle workspace) (check End-Effectors table). *e1* can support *o1* (Cylinder) with *g3* (Bottom-Table grasp), while *e3* can grasp *o1* with *g7* (Top-RightArm). Since a transition between *g3* and *g7* exists (check Grasps table, row 3), the cylinder can be safely grasped with the right arm from the table, as shown in Fig. 5, right panel.

The Cartesian product between reachable workspaces, adjacency, and grasp transitions, allows to create a graph as the one depicted in Fig. 6.

IV. Algorithm Description

An object is in a certain state $S_{e,g,w}$ when: (i) it is being grasped by a particular end-effector e, with (ii) a specific grasp g, and (iii) it is inside a specific workspace w. Moving an object means making a transition from its current state to a different one. In this work, a high-level plan is a sequence of states and transitions.

We show in Algorithm 1 the complete proposed algorithm, denoting with:

- C_{Init}, C_{Current}, and C_{Final} the initial, current and final object configurations in Cartesian space,
- G the graph,
- P_C the Cartesian plan,
- $S_{\text{Init}}= S_{e_i,g_i,w_i}$ and $S_{\text{Final}}= S_{e_t,g_t,w_t}$ the initial and final object configurations in high-level space,
- P_{HL} the high-level plan,
- Arc_{Id} the id of the arc that makes the planning fail,
- p_i a single item of the Cartesian plan,
- M a joint space motion plan.

Algorithm 1: MoveObject

```
(states,transitions) = GetInformationFromDB();
G= GenerateGraph(states,transitions);
C_Current= GetInitialStateFromVision();
C_Final= GetFinalStateFromUser();
repeat
    G_HL= G;
    C_Init= C_Current;
    S_Init= ConvertCartesianToHighLevel(C_Init);
    S_Final= ConvertCartesianToHighLevel(C_Final);
    repeat
        P_HL= ComputeShortestPath(G_HL,S_Init,S_Final);
        if P_HL is empty then
            GlobalFailure;
        end
        P_C= ConvertHighLevelToCartesian(P_HL);
        if P_C is empty then
            Arc_Id= GetFailedConversion();
            G_HL= Backtracking(G_HL,Arc_Id);
        end
    until P_C is not empty;
    /* Executing the cartesian plan    */
    for p_i ∈ P_C do
        M=MotionPlanning(p_i);
        if M is valid then
            Execute(M);
            C_Current= CurrentObjectPosition();
        else if C_Current== C_Init then
            GlobalFailure;
        else
            break for;
        end
    end
until C_Final== C_Current;
```

A. Graph Generation

Using the full DB structure as illustrated in Sec. III, a graph G is generated by intersecting transitions between grasps, reachability from end-effectors to workspaces and workspaces adjacency information. For the exemplary database content illustrated in Fig. 3, the generated graph is depicted in Fig. 6. Note that this is a simplified example: a real experiment graph could have hundreds of nodes and thousands of arcs, as reported in TABLE I. For a picture of a real generated

graph see attached multimedia file.
Each node of the graph represents a state $S_{e,g,w}$ of the object. Each arc represents one of the two different kinds of transition (akin to "transit" and "transfer" concepts of [6]):

1) from S_{e,g,w_i} to S_{e,g,w_j}: change of workspace with the same grasp (thus the same end-effector), where e must be movable and w_i and w_j must be adjacent (in the sense of Sec. III-C) and reachable by e. This is represented by the dashed lines in Fig. 6.
2) from $S_{e_i,g_k,w}$ to $S_{e_j,g_l,w}$: change of end effector inside a certain workspace, where a transition between grasp g_k and g_l must exist (as described in Sec. III-D). This is represented by the solid lines in Fig. 6.

The weight of any arc could be chosen based on the failure probability of the associated transition or on the stability of the grasps involved. In this work we make the simple consideration that a change of end effector (transition from $S_{e_i,g_k,w}$ to $S_{e_j,g_l,w}$) has a higher failure rate w.r.t. a change of workspace (transition from S_{e,g,w_i} to S_{e,g,w_j}), thus we use weights with different order of magnitude, e.g. 1.0 and 0.1, respectively.

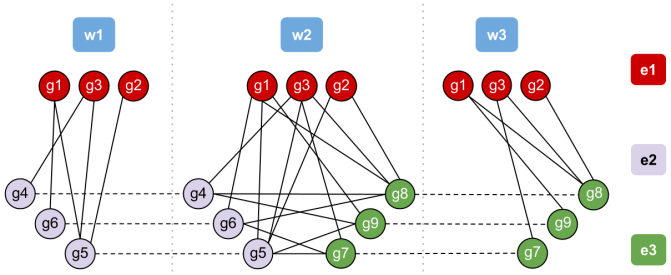

Fig. 6: Graph generated from the exemplary DB in Fig. 3: each node represents a state of the object being grasped by the end effector e_i ($i = 1, \ldots, 3$) with the grasp g_k ($k = 1, \ldots, 9$), in workspace w_m ($m = 1, \ldots, 3$). Solid lines represent grasp transitions within the same workspace; dashed lines represent object displacements between workspaces with the same grasp. Simultaneous changes in workspace and grasp being applied to the object are not allowed in the current framework.

Note that the built graph no longer distinguishes between the type of end effectors or transitions.

B. Cartesian-to-High-Level Pose Conversion

In order to provide the initial and final states $S_{\text{Init}} = S_{e_i,g_i,w_i}$ and $S_{\text{Final}} = S_{e_t,g_t,w_t}$ our algorithm uses the information gathered from the vision system and the input by the user, respectively.
Workspaces w_i and w_t are found by using the database geometry, searching for those containing the required initial and target poses (note that workspaces do not overlap because they are the cells of a grid).
Initial end-effector e_i is assumed to be known, while target end-effector e_t is assumed to be known and non-movable in order to avoid ambiguous situations where multiple end effectors could simultaneously hold the object in the target position (see Fig. 7).

Finally, a search for grasps (g_i, g_t) that fulfill the initial and final pose is performed; this involves searching through all possible grasps associated to the end-effectors (e_i, e_t), which are checked for similarity with respect to the initial and target poses of the object.

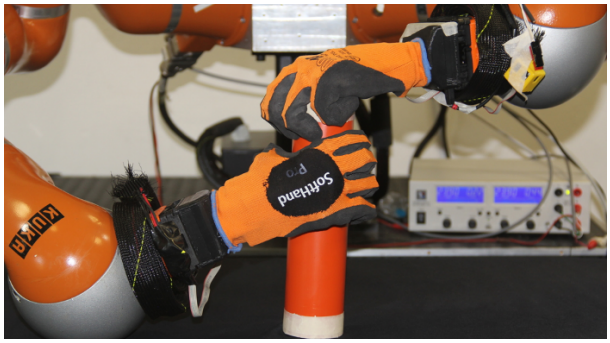

Fig. 7: The object in its final position can be grasped by three different end-effector (table, left hand, right hand)

C. Computation of the shortest path

We want to find a sequence of states from S_{Init} to S_{Final}, while minimizing the number of grasp transitions between end-effectors. To this sake any shortest path algorithm can be used: we choose Dijkstra [25] as one of the most widely known and easiest to implement.
In Fig. 8 we show the solution (path) for given initial and final states, which uses a direct handoff between the two end-effectors, while in Fig. 9 we show a plan with different initial and target states that automatically uses the table as intermediate end-effector in order to minimize path length. Each path represents a set of workspace/grasp transitions, which need to be converted into a discrete sequence of Cartesian position of the object and its grasping end-effector.

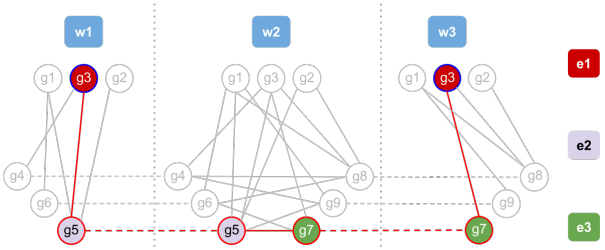

Fig. 8: Planning given by Dijkstra, the transition on the table is avoided because it would require more grasp transitions for the task.

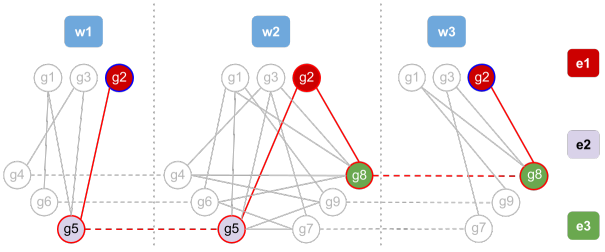

Fig. 9: Planning given by Dijkstra, the transition on the table is used because it is the shortest one for these initial and final states.

D. High-Level-to-Cartesian Conversion

The aforementioned conversion from the high-level plan P_{HL} to a Cartesian sequence P_C is a critical aspect of our approach, since it allows the decoupling between high-level object passing/moving and low-level motion planning. Each arc in the high-level plan is associated to low-level commands: in particular, a change of workspace generates a *move* command for an end-effector, while a change of grasp generates a pair of *grasp-ungrasp* commands for the end-effectors involved.

The discrete high-level plan is first simplified by removing unnecessary intermediate transitions, so that an end-effector $\hat{e}$ moving an object through multiple workspaces $S_{\hat{e},g,w_1}...S_{\hat{e},g,w_n}$ generates a single *move* command for the end effector to reach the final workspace w_n.
After reaching that workspace, the object is either in its final target position, or needs to be passed to another end-effector by a sequence of *grasp-ungrasp* commands. In order to find a valid position for the object to be grasped by the next end-effector while still being grasped by the previous one, the following procedure has been applied.

- Create a virtual serial chain connecting the initial and final end-effectors together as if they were rigidly attached to each other by a constant $SE(3)$ matrix M. This represents the poses the end-effectors have to maintain in order to exchange the object.
- Consider as a virtual base-link the base-link of the initial arm, and as a virtual end-effector the base-link of the final arm.
- Solve the IK for the virtual chain with any jacobian-based approach avoiding self-collisions and collisions with environment. The target of this IK corresponds to the base-link of the final arm: in this way the closed-loop kinematic chain given by the two-arms connected through the object is respected.

E. Backtracking procedure

In case the conversion between high-level and Cartesian plans fails due to the inability of finding a collision-free configuration for performing a particular transition, a backtracking procedure is performed; this removes the arc associated to the failing transition and goes back to computing a shortest path on the updated graph.
The loop between phases in sub-sections IV-C, IV-D and backtracking is iterated until no backtracking is necessary (i.e. all states and transitions of a high-level plan are found to be feasible) or no valid high-level plan is found, in which case the target state is not kinematically reachable from the initial state with a collision free set of transitions.

F. Execution of the Plan

The execution of the *move*, *grasp*, *ungrasp* commands requires the use of a low level motion planning. If more than a single end-effector *move* command has to be performed, they are performed simultaneously, computing collision-free trajectories that involve the various end-effectors. This can be efficiently done using, e.g. a sample-based random planner.
In our implementation we used RRT* [26] with RRT-Connect [27] as a backup, both implemented in MoveIt! [28].
Even though the initial and final joint configurations have been tested to be collision-free, a collision-free motion plan is not granted to be found. In this unlikely case, the current object position is considered: if it is different from the starting one, a new high-level action plan is initialized with the current object position as the new starting one; otherwise, the algorithm ends with a failure to avoid an infinite loop.

V. Experimental Results

In this section the experiments to validate our approach are reported. Experiments have been performed without specifying any preferred strategy. The algorithm elaborates a plan and, depending on the situation, it produces a strategy with either a single or dual arm, with or without handoffs. We will now describe the different emerging behaviors obtained during the experiments. A video of these experiments is available in [29].

A. Experiments

Moving an object using handoff: In the reported examples (see Fig. 10) the robot uses two movable end effectors to move the object to the target position: this behavior is preferred since less e.e. switches are needed to perform it.

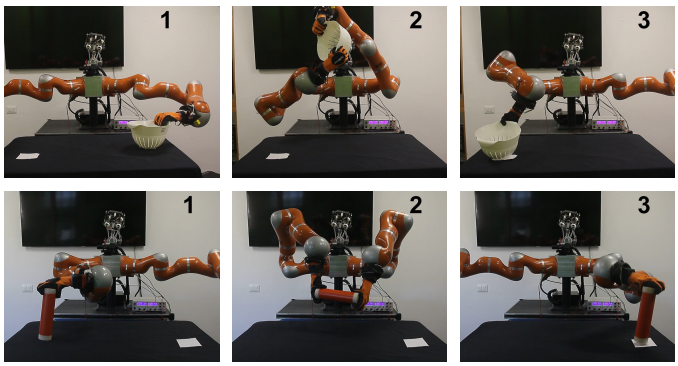

Fig. 10: Handoff experiments with a colander (on the top) and a cylinder (on the bottom).

Using support surfaces: Support surfaces are exploited in the case one single handoff is not possible. In this way the robot can re-grasp the object and then achieve the goal. In Fig. 11 single arm regrasping are reported, while in Fig. 12 the robot uses both arms to perform the task.

Pick-and-place: In the experiments reported in Fig. 13 the robot can execute the moving task using only one arm without any re-grasping. The particular case of trashing an object is shown in Fig. 14: the trash bin is being considered as a non-movable e.e. as well.

Assembly Line Simulation: To validate the generality and scalability of our approach with respect to the number and type of robots, we simulated kinematically the scenario depicted in Fig. 15. Here, five Kuka LWR arms are placed in front

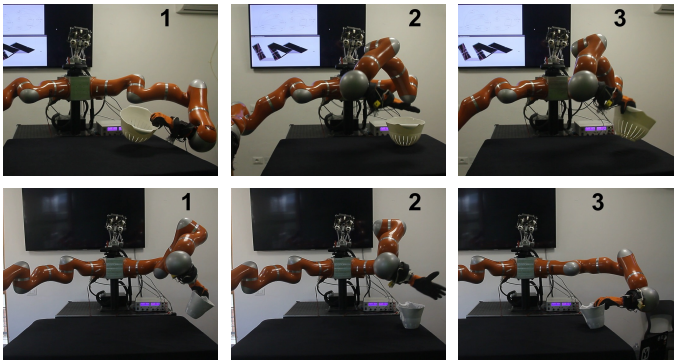

Fig. 11: Single arm object regrasping with a colander (on the top) and a pitcher (on the bottom).

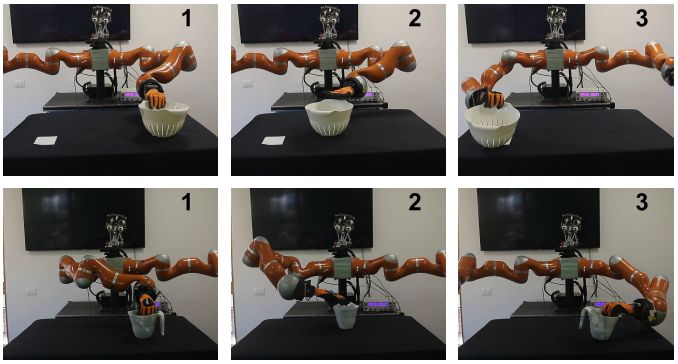

Fig. 12: Dual arm object regrasping with a colander (on the top) and a pitcher (on the bottom).

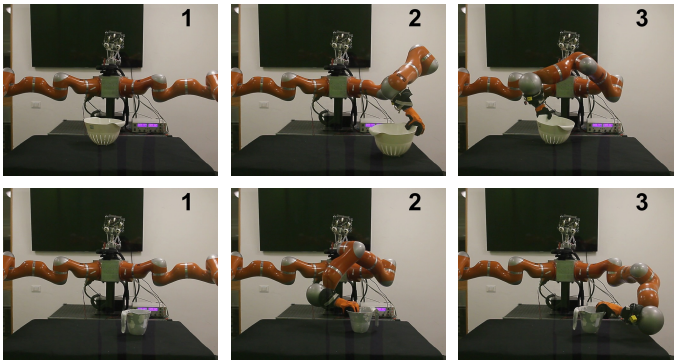

Fig. 13: Single arm pick-and-place experiments with a colander (on the top) and a pitcher (on the bottom). On the left the desired position is reported, then the initial and the final one are respectively in the middle and on the right.

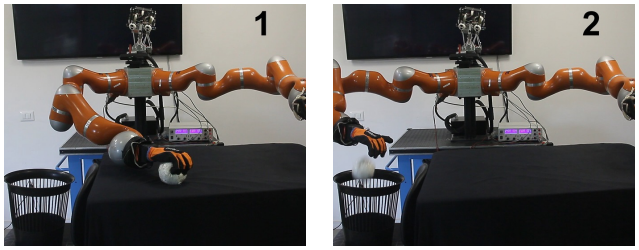

Fig. 14: Trashing a ball experiment.

of four workbenches, one of which contains a conveyor belt, which is modeled as a 1 DoF prismatic robot which can move horizontally objects that are placed on its surface. Using the procedure described in Sec. III-C a high-level description with 9 workspaces was obtained. The cylindrical object, initially resting on the first bench below a shelf (blue cylinder), has to find itself in the final configuration resting on top of the last bench (red cylinder). The outcome of the planning algorithm is a sequence consisting of the following actions: pick from the source location with first arm, handoff between first and second arm, place on the table, pick with third arm, handoff between third arm and fourth arm, handoff between fourth arm and conveyor belt, handoff between conveyor belt and fifth arm, place in the final location. Please refer also to the video attachment for more details.

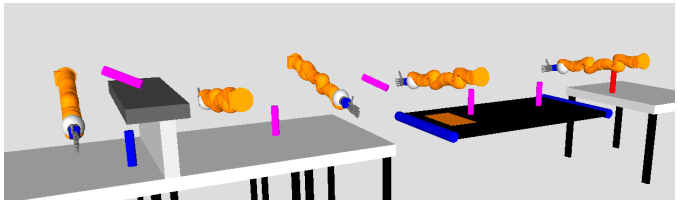

Fig. 15: An assembly line simulation with 5 Kuka LWR and a conveyor belt. The cylinder is moved from the initial position on the left (blue) to the final one on the right (red). The intermediate positions in which the object is exchanged between two e.e.'s are depicted in magenta.

B. Computational Costs

All experiments have been performed on a Intel Core i7-4770K machine with 16GB RAM using a single thread. The algorithm implementation could be highly parallelized thanks to its many parts based on randomized approaches, thus we expect a 3x speed-up to be a realistic value using 4 threads. Please consider this in comparison with other similar works.
In TABLE I we report computation times (in seconds) for the planning procedure described in Algorithm 1, both for the Vito robot and the assembly line scenarios, averaged on 100 trials with 5 different initial and target configurations for each task (table column).

	Vito Robot				Assembly Line
	Pitcher	Colander	Ball	Cylinder	Cylinder
GRAPH					
Nodes	108	144	80	224	736
Arcs	1609	3426	1056	5248	14016
HL-TIME					
avg	**3.2**	**5.4**	**1.9**	**2.6**	**8.5**
max	4.6	9.4	2.4	3.2	21.3
min	1.8	2.5	1.1	2.1	3.1
TOT-TIME					
avg	24.2	28.5	17.6	21.5	53.0

TABLE I: High level planning and total computation times (s).

While a high number of nodes and arcs should intuitively increase the computational time, indeed most of the CPU is used for the collision checking inside HighLevel_to_Cartesian conversion. Objects characterized by a simple geometry such as a ball or a cylinder are usually easier to handle and move, thus less arcs are tested in the loop of Algorithm 1.

The assembly line experiment requires more handoffs and surface supports (transitions), thus it is more likely to fail and cause a replan, as we can see from the greater max-min range. Obviously, the number of transitions depends on the strategy chosen by the high level planner, e.g. 2 for a Pick-and-Place and 3 for an handoff.
For each transition, the motion planning uses five seconds for

RRT* and, as a backup solution, ten attempts of RRT-Connect (up to a total of three more seconds). In our experience, and given the time allotted for the first phase, RRT* fails in about 40% of the transitions, mostly during planning approach trajectories to the object in a pose difficult for the robot to reach.

VI. Conclusions

In this work we proposed a high-level planner that unifies different object moving strategies such as pick-and-place and handoff, and exploits support surfaces if required to achieve the goal. Both a real and simulated examples with different objects and multiple manipulators show the efficacy and scalability of this approach.

Ongoing work is focused on extending the approach to planning for simultaneously moving more than a single object, on parallelizing planning and execution phases, and on considering more complex interactions between possibly multiple end-effectors and the object to be moved.

All the code of this framework is available at http://dualmanipulation.bitbucket.org.

Acknowledgments

Authors thank Gian Maria Gasparri and Federico Spinelli for fruitful discussions on the preliminary version of this work.

References

[1] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation a survey," *Robotics and Autonomous Systems*, vol. 60, no. 10, p. 1340, 2012.

[2] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 445–452.

[3] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," *Robotics: Science and Systems VII*, 2011.

[4] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation." in *ICAPS*, 2010, pp. 254–258.

[5] J. A. Bagnell, F. Cavalcanti, L. Cui, T. Galluzzo, M. Hebert, M. Kazemi, M. Klingensmith, J. Libby, T. Y. Liu, N. Pollard, *et al.*, "An integrated system for autonomous robotics manipulation," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2955–2962.

[6] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.

[7] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[8] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *Intelligent Robots and Systems. IROS 2009. IEEE/RSJ International Conference on*, pp. 2464–2470.

[9] J.-P. Saut, M. Gharbi, J. Cortés, D. Sidobre, and T. Siméon, "Planning pick-and-place tasks with two-hand regrasping," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4528–4533.

[10] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simultaneous grasp and motion planning: Humanoid robot armar-iii," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 2, pp. 43–57, 2012.

[11] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt, "Combining task and path planning for a humanoid two-arm robotic system," in *Proceedings of TAMPRA: Combining Task and Motion Planning for Real-World Applications (ICAPS workshop)*. Citeseer, 2012, pp. 13–20.

[12] B. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for dual-arm manipulation with upright orientation constraints," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3784–3790.

[13] F. Zacharias, D. Leidner, F. Schmidt, C. Borst, and G. Hirzinger, "Exploiting structure in two-armed manipulation tasks for humanoid robots," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5446–5452.

[14] B. Cohen, S. Chitta, and M. Likhachev, "Single-and dual-arm motion planning with heuristic search," *The International Journal of Robotics Research*, p. 0278364913507983, 2013.

[15] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth, "Robotic roommates making pancakes," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 529–536.

[16] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 639–646.

[17] G. Havur, K. Haspalamutgil, C. Palaz, E. Erdem, and V. Patoglu, "A case study on the tower of hanoi challenge: Representation, reasoning and execution," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4552–4559.

[18] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1470–1477.

[19] L. Lindzey, R. A. Knepper, H. Choset, and S. S. Srinivasa, "The feasible transition graph: Encoding topology and manipulation constraints for multirobot push-planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 301–318.

[20] K. Hauser, T. Bretl, J. Latombe, and B. Wilcox, "Motion planning for a six-legged lunar robot," *Algorithmic Foundation of Robotics*, 2008.

[21] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *The International Journal of Robotics Research*, 2011.

[22] F. Spinelli, "Pose estimation library," https://bitbucket.org/Tabjones/pose-estimation-library.

[23] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics & Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110–122, 2004.

[24] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Humanoid Robots. Humanoids 2008. 8th IEEE-RAS International Conference on*, pp. 189–196.

[25] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[26] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, p. 1478.

[27] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, pp. 995–1001.

[28] I. A. Sucan and S. Chitta, "Moveit!" http://moveit.ros.org.

[29] H. Marino, M. Ferrati, A. Settimi, C. Rosales, and M. Gabiccini, "RA-L submission accompanying video," https://youtu.be/vIHg6gzIemQ, Sept. 2015.