applied sciences

MDPI

Article

Contingent Task and Motion Planning under Uncertainty for Human–Robot Interactions

Aliakbar Akbari *, Mohammed Diab and Jan Rosell

Institute of Industrial and Control Engineering, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain; mohammed.diab@upc.edu (M.D.); jan.rosell@upc.edu (J.R.)

* Correspondence: ali.akbari101@gmail.com

Received: 8 February 2020; Accepted: 24 February 2020; Published: 1 March 2020

Abstract: Manipulation planning under incomplete information is a highly challenging task for mobile manipulators. Uncertainty can be resolved by robot perception modules or using human knowledge in the execution process. Human operators can also collaborate with robots for the execution of some difficult actions or as helpers in sharing the task knowledge. In this scope, a contingent-based task and motion planning is proposed taking into account robot uncertainty and human–robot interactions, resulting a tree-shaped set of geometrically feasible plans. Different sorts of geometric reasoning processes are embedded inside the planner to cope with task constraints like detecting occluding objects when a robot needs to grasp an object. The proposal has been evaluated with different challenging scenarios in simulation and a real environment.

Keywords: task and motion planning; manipulation planning; robot-human interactions; perception

1. Introduction

Robotic manipulation tasks become highly challenging when a mobile manipulator is required to obtain a feasible plan to solve a given problem under potential uncertainties. Uncertainty shall be viewed in the initial state of the robot environment, e.g., objects may rest in different positions or some object features (like color) could be initially unknown for a robot. Uncertainty, moreover, must be considered in the result of manipulation actions (as nondeterministic effects) since there could be different action outcomes. To deal with such uncertainties, robots generally look for a sequence of actions to satisfy the goal of a task and perform replanning in the case of action execution failure or uncertain situations. This process may be costly while a robot requires repetition of expensive replanning.

To tackle those challenging issues, these problems can rely on contingent task planning which plans in belief space and can generate conditional plans under uncertainty in terms of initial state and action effects. Contingent-based task planners can provide a tree of plans rather than a single sequence of executive actions. Therefore, uncertainty is observed during the plan execution, and the tree of plans is followed according to the binary observation values.

Other challenges are related to some demanding or difficult tasks which are either not performable easily by robots or are out of their reach, but that can be done in collaboration with a human operator. In these cases, the robot can ask the human operator to do some particular difficult actions, to transfer some objects located in the human workspace or to share knowledge that is initially incomplete to the robot. Moreover, there could be some geometric constraints imposed in the environment, e.g., lack of space for placing objects, occlusions, kinematic issues, etc., and the finding of the geometric values for each manipulation action becomes substantial in order to make a manipulation plan feasible. Therefore, the way of combining task and motion planning plays a significant role when the manipulation task is highly constrained in terms of geometric information and there is amount of uncertainty.

In this paper, we are going to deal with manipulation tasks carried out by a mobile manipulator assisted by a human operator. The mobile manipulator will be responsible to execute the main task, while the human operator will be responsible for some difficult actions (like to open some box-like containers which cannot be opened by the robot), to share knowledge with the robot, and to transfer objects to the robot when they are not reachable. Uncertainty in the initial state and in some action effects are considered. Some manipulation and sensing actions are considered in the current proposal, which allow illustration of the approach, and that can be extended to handle a broader set of manipulation tasks. No geometric uncertainty is considered, e.g., in the robot motion or the object poses.

Contributions: To deal with the aforementioned challenges, we propose a contingent-based task and motion planner based on *Contingent-FF* [1] that works under uncertainty and considers human–robot collaboration. The *Contingent-FF* includes two main components, *heuristic evaluation* and *search space*, and results in a tree-shaped set of plans involving sensing actions. Three main contributions extend the basic *Contingent-FF* planner:

- *Robot action reasoning.* Two types of geometric reasoning are proposed and integrated with the basic planner: *relaxed geometric reasoning* and *lazy motion evaluation*. The former refers to *Reachability*, *Spatial*, and *Manipulation* reasoning. This reasoning process is embedded within the heuristic computation of the planner. Motion paths are lazily evaluated when actions are selected by the state space search. If the reasoning processes fail, geometric constraints are fed back to the planner. This part of the computation is done offline and aims to prune infeasible actions due to geometric constraints and to obtain a feasible set of actions in the tree of plans. As the basic contingent planner considers only symbolic reasoning, this module enables it to incorporate geometric reasoning to deal with practical applications. The reasoning process provides feasible initial and goal configurations for motion planning queries, improving its success rate and thus the overall performance of the planner in the generation of a feasible manipulation plan.
- *Human–robot collaboration.* There are some actions which can be executed by the robot and others that require the collaboration of a human operator. The proposed relaxed geometric reasoning is extended to inform the planner about which actions cannot be executed by the robot, and hand over them to the human operator, allowing the planner to handle those cases where the selection of actions to be performed by a human operator is required. In these cases, the geometric world resulting by these actions is simulated and used in the planning system for further geometric reasoning evaluation. This step makes the basic planner flexible to consider the result of human actions, extending its performance to situations it is not able to be handled autonomously.
- *State observation.* To observe the binary outcomes of actions, two modules are proposed: *perception* and *human knowledge*. Perception is used to detect, e.g., the actual locations of the objects or some objects feature like color. The knowledge provided by the operator is required for more difficult observations like determining if a can is filled or empty, or if a glass contains a given drink. Action observation takes place at execution time. The combining of both modules widens the capacity of the planner to identify the current situation of the robot's world and decide the best course of actions in execution, thus improving the planner performance in finding feasible solutions.

One of the main advantages of the proposed framework is that the offline computation is valid and works despite the actual values of the uncertainty variables or the actual outcomes of the executable actions.

The rest of the paper is structured as follows. First, Section 2 summarizes some related work and Section 3 explains a proposal for contingent task and motion planning. Afterwards, Section 4 presents and illustrates the proposed relaxed geometric reasoning for mobile manipulators, Section 5 demonstrates contingent heuristic computation using relaxed information, Section 6 details tree-based planning using search space, and Section 7 presents manipulation plan execution using sensing and human interaction. Finally, Section 8 shows some implementation issues as well as empirical results, and Section 9 sketches the conclusions and future works.

2. Related Work

Manipulation problems of different nature have been tackled in the literature with different strategies, e.g., the manipulation problem of Navigation Among Movable Obstacles (NAMO) has been addressed in [2,3] using a backward search algorithm, and dual-arm table-top manipulation problems by combining motion planning and task assignment [4]. These robotic applications, like many others, must deal with different sources of uncertainty and the use of sensors and perception strategies may be required, e.g., the studies in [5,6] have investigated the machine robotic cell scheduling problem for manufacturing systems with or without sensor inspection. The following sections classify more approaches in the field of task and motion planning with and without uncertainty.

2.1. Task and Motion Planning without Uncertainty

Recently, much study has been centered to solve robotics manipulation tasks by combining task and motion planning problems with no consideration on uncertainty. It is assumed that the initial state of the environment is perfectly known, and actions are deterministic, i.e., state of planning is only changed by the selected action. There is a huge number of task planners being able to solve manipulation problems under perfect information [7].

In principle, two methods of combining task and motion planning have been explored: interleaved or simultaneously. Several studies call first task planning, and then motion planning to determine whether a plan is feasible or not such as [8–12]. In the case of failure, geometric constraints are identified and reported to task planning and the procedure continues. This might be costly as a number of times the process could be repeated in order to find a geometrically feasible plan.

On the other hand, other approaches enable task planning to incorporate geometric reasoning within the task planning process [13–18]. Hence, in this case, task planning results in a feasible manipulation plan. In this line, we recently proposed a heuristic-based task and motion planner [19] to deal with constrained table-top problems for bi-manual robots by offering different type of geometric reasoners that can be used in heuristic computation or when an action is selected. Our previous approach does not consider any uncertainty, human actions, and reasoning about mobile manipulation problems which are the subjects of this paper.

The way of integrating task and motion planning information in the current proposal is based on the simultaneous approach in order to generate feasible plans, and is an extension of [19] that copes with mobile manipulators, uncertainty, and to consider collaborative tasks with human operators.

2.2. Task and Motion Planning under Uncertainty

There are some situations in which a robot has incomplete information about its manipulation environment; therefore, it needs to plan under uncertainty. Task planning under uncertainty is a well-established field in Artificial Intelligence. Conditional-based task planners can provide conditional plan to cope with uncertain information when either the initial state is not completely known, or the result of actions are nondeterministic. There are various classes of planning in this field like conformant, contingent, or probabilistic planning.

Conformant planning looks for plans under given uncertainty concerning the start state and the effects of symbolic actions, assuming no sensing capabilities during the execution of the plan. The plan should be successful regardless of which is the start state. Contingent planning also considers uncertainty regarding the start state and the effects of actions. However, it can provide some sort of observation over a conditional plan in execution. Probabilistic planning does planning under probabilistic uncertainty regarding the start state and the effects of actions.

More details on some approaches following conditional-based task planning are commented next as we are interested in this type of planner due to its feature of providing observation over a conditional plan. Some conditional task planners are *Contingent-FF* [1], *POND* [20], and *PKS* [21]. They plan in the belief space and compute conditional plans in the offline mode, which are guided by the result of sensing actions. On the other hand, there are some conditional task planners like *K-Planner* [22], *SDR* [23], and *HCP* [24] solving conditional plans online. Although these planners can prune some branches by considering online sensing actions, satisfying the goal of task may not be possible and the planners may face with dead-end even if there is a solution.

The concept of contingent-based task and motion planning has also emerged. For instance, the *Planning with Knowledge and Sensing (PKS)* planner considers incomplete information and performs contingent planning [25] in two main scenarios, using *force sensing* and *visual sensing*. In a similar direction, offline-based hybrid conditional task and motion planning has been proposed [26], i.e., task planning is foremost performed, and then geometric evaluation is considered by incorporating low-level feasibility checks inside conditional planning (assuming that actuation actions are deterministic). On the contrary, the approach proposed here interweaves simultaneously efficient geometric reasoning inside the task planning process to provide geometrically feasible plans. The approach also copes with collaboration between the mobile robot and a human operator to perform a manipulation task.

3. A Proposal for Contingent Task and Motion Planning

This section first presents a brief overview of the original *Contingent-FF* task planning, and the modifications introduced in the present proposal to compute geometrically feasible manipulation conditional plans.

3.1. Contingent-FF Overview

The *Contingent-FF* task planner [1] handles uncertainty in the initial state and in the result of actions. The task planner has two main components which are heuristic computation and search space. For the heuristic computation, the planner uses a modified version of the *Relaxed Planning Graph (RPG)* used in the *Fast-Forward (FF)* planner [27]. The relaxed plan including a number of relaxed actions is computed from the *RPG*, and the heuristic value is the length of this relaxed plan. Also, promising actions (called helpful actions in *FF*) are extracted from the relaxed plan as a pruning technique in the search space, as discussed in *FF*. The *Contingent-FF* planner extends the *RPG* process, called *CRPG*, by adding unknown facts in an additional layer in the heuristic phase. Known facts are basically those which do not have uncertainty and unknown facts are the ones which could be the result of nondeterministic actions or uncertain in the initial state. It introduces reasoning about unknown facts that allows such facts to become known in the *RPG* process. Once *CRPG* is successfully built, the relaxed plan is extracted.

In *Contingent-FF*, belief states including known and unknown facts are considered. The search space starts from the initial belief state and applies an *And-Or* search. The search space progress is guided by the heuristic value and helpful actions. The result of planning provides conditional plans that may involve a variety of sensing actions whose outcome causes different plan branches.

3.2. Planning Formulation

Our planning system domain $\mathcal{D}$ is a tuple $\langle \mathcal{A}, \Omega, \mathcal{F}, \mathcal{W}, S_g \rangle$ where $\mathcal{A}$ is the action space, Ω is the sensing action space, $\mathcal{F}$ is a set of literals, $\mathcal{W}$ is a workspace involving a mobile manipulator $\boldsymbol{\mathcal{R}}$ (described by the pose of the base Pos_{rob} along the arm configuration Q_{rob}) and a number of objects $\boldsymbol{\mathcal{O}}$, and, S_g is a set of grasping poses described for objects. Objects are denoted as: $\boldsymbol{\mathcal{O}}=\{\mathcal{O}^m_1(pos,fe) \dots \mathcal{O}^m_j(pos,fe),\ \mathcal{O}^f_1(pos,fe) \dots \mathcal{O}^f_k(pos,fe)\}$, where j and k are the number of *Movable* and *Fixed* objects respectively, whose initial position and orientation are denoted by *pos*, and whose features are denoted by *fe*.

An action $a \in \mathcal{A}$ is a tuple $\langle$*name(a)*, *pre(a)*, *effect(a)*, *coneffect(a)*, *geom(a)*, $\mathcal{Q}(a)\rangle$, where *name(a)* is the action symbolic name, *pre(a)* is a propositional formula which must hold for the action to be applied, *geom(a)* is the numerical counterpart of an action containing geometric information, *effect(a)* represents the negative and positive effects of a on the state it is applied to, and $\mathcal{Q}(a)$ is a query function to the motion planner which computes a motion between two robot configurations and stores the solution if any. A relaxed action $a' \in \mathcal{A}'$ (where $\mathcal{A}'$ is the relaxed action space) is similar to the action despite it does not consider any negative effects. Actions refer to executable actions, i.e., requiring motion, and can be done by either the robot or a person. The following actions types are considered to deal with some examples of mobile robot manipulation:

- *Transit*: an action done by the robot to travel from one configuration to another one without an attached object.
- *Transfer*: an action done by the robot to move an attached object from one pose to another one.
- *Push*: an action done by the robot to push an object from one pose to another one.
- *Open*: an action done by the robot to open a box-like container (articulated cap with prismatic joint is assumed with two positions corresponding to fully closed and fully opened, the state being stored in the containers objects features).
- *HumanTransfer*: an action done by a person to transfer/push an object to the robot workspace.
- *HumanOpen*: an action done by a person to open a box-like container.

Each sensing action is a tuple $\langle$*pre(a), o(a)*$\rangle$, where *o(a)* is a literal with uncertainty. These are actions not involving motion, devoted to observing the value of *o(a)*. The observation is done in run-time. Some sample sensing actions are considered in the proposed planning system. They are the following:

- *SenseColor*: a sensing action is done by a perception module to determine the color of an object.
- *SensePose*: a sensing action is done by a perception module to determine the pose of an object.
- *CheckContainer*: a sensing action is done by a person to evaluate whether a container is open or not.
- *CheckCan*: a sensing action is done by a person to evaluate whether can-like objects are filled or not.

A belief state S is a tuple $S = \langle \mathcal{P}, \mathcal{V} \rangle$ where $\mathcal{P}$ includes a set of known literals which hold in that state and a set of uncertain literals which may hold or not in the state, and $\mathcal{V}$ represents a full geometric description of the scene, i.e., configurations of robots and poses of objects corresponding to certain and uncertain literals. An executable action from a state S_1 results in a new world state using the state transition functions $S_2.\mathcal{P} := S_1.\mathcal{P} - \mathit{effect}^-(a) + \mathit{effect}^+(a)$ and $S_2.\mathcal{V} := S_1.\mathcal{V} - \mathit{geom}^-(a) + \mathit{geom}^+(a)$. A sensing action splits a belief state and introduces two branches into the plan marked with *o(a)*, and $\sim$ *o(a)*.

The planning problem $\mathcal{T}$ is expressed by a tuple $\langle \mathcal{D}, \mathcal{S}_0, \mathcal{G} \rangle$ where $\mathcal{D}$ is a domain, $\mathcal{S}_0$ consists of a set of literals representing the initial symbolic state $\mathcal{I}$ such that $\mathcal{I} \subseteq \mathcal{F}$ along their geometric assignments regarding the initial state of the world $\mathcal{W}_0$, and $\mathcal{G} \subseteq \mathcal{F}$ is the set of symbolic goal conditions. The solution of a *Combined Task and Motion Planning (TAMP)* problem under uncertainty, which we denote by π, is a tree-shaped conditional plan, i.e., a sequence of symbolic actions achieving $\mathcal{G}$, along with a feasible motion for each action.

3.3. Geometric Constraint Predicates

Basically, three general predicates, evaluated by geometric reasoning, are allocated that set constraints to the task states: *isCrit*($\mathcal{O}_j^m$, $\mathcal{O}'$, *Pos*), *infeasByRob*($\mathcal{R}$, $\mathcal{O}'$, *Pos*), and *assist*(*Human*, $\mathcal{O}'$, *Pos*). The first predicate indicates that there is a blocking object $\mathcal{O}_j^m$ which is located towards the target object $\mathcal{O}'$ placed in the pose *Pos*. The second one shows that the target object cannot be manipulated by the robot $\mathcal{R}$ in the corresponding pose *Pos*. The last predicate shows the manipulation action with the target object $\mathcal{O}'$ and the corresponding *Pos* must be done by a human operator *Human*.

The proposed predicates are interleaved inside the pre- and post-conditions of the actions. Concerning the actions done by the robot, the predicates $\sim$*isCrit* and $\sim$*infeasByRob* are inserted within the preconditions of the actions *Transit, Open, Transfer,* and *Push* in order to avoid moving the robot to any unreachable or infeasible configuration. Referring to the post-conditions, the last two actions may include the negation of the predicate *isCrit* if they are moving a blocking object to a placement where the obstruction does no longer hold. With respect to the actions performed by a human, the preconditions of the actions *HumanTransfer* and *HumanOpen* may include the predicate *assist* in order to indicate the requirement of a human operator.

To illustrate the use of these predicates, the actions *Transit* and *HumanOpen* are described next. The action template *Transit*($\mathcal{R}$, $\mathcal{O}_i^m$, *Surface, Pos)* is designed to move the robot $\mathcal{R}$ (arm and base), without holding any object, towards a grasp configuration of a manipulatable object $\mathcal{O}_i^m$ located on a surface *Surface* at pose *Pos*. In the final configuration, the robot holds the target object. The action is applicable if the following preconditions hold: the object is located on a surface, top of the object is clear, the robot arm is empty, it can reach the grasp configuration if there is no movable objects blocking its way to $\mathcal{O}_i^m$. The last precondition is represented by fact *isCrit*($\mathcal{O}_j^m$, $\mathcal{O}_i^m$, *Pos)*; objects that make this fact to hold are called *Critical Objects* which are the objects blocking the way of reaching the object. As a result of the action, the robot holds an object.

Transit($\mathcal{R}$, $\mathcal{O}_i^m$, *Surface, Pos):*

Pre: *onSurface*($\mathcal{O}_i^m$, *Surface, Pos), armEmpty*($\mathcal{R}$)*, clear*($\mathcal{O}_i^m$), $\sim$*infeasByRob*($\mathcal{R}$, $\mathcal{O}_i^m$, *Pos),*
$\forall \mathcal{O}_j^m$ $\sim$*isCrit*($\mathcal{O}_j^m$, $\mathcal{O}_i^m$, *Pos)*

Effect: *holding*($\mathcal{R}$, $\mathcal{O}_i^m$, *Pos)*, $\sim$*clear*($\mathcal{O}_i^m$), $\sim$*armEmpty*($\mathcal{R}$), $\sim$*onSurface*($\mathcal{O}_i^m$, *Surface, Pos)*

The action template *HumanOpen(Human,* $\mathcal{O}_i^m$, *Pos, Closed, Open)* is used to open a box-like container $\mathcal{O}_i^m$ when it is closed. The action is applicable if the robot needs the assistance from a human operator, represented by the *assist* predicate, and its status is closed, shown by the predicate *status*. These conditions are introduced in the action preconditions. As a result of the action, the corresponding container will be open.

HumanOpen*(Human,* $\mathcal{O}_i^m$, *Pos, Closed, Open):*

Pre: *assist(Human,* $\mathcal{O}_i^m$, *Pos), status*($\mathcal{O}_i^m$, *Closed)*

Effect: *status*($\mathcal{O}_i^m$, *Open)*, $\sim$*status*($\mathcal{O}_i^m$, *Closed)*

3.4. The Proposed Framework

The proposed framework for task and motion planning under uncertainty extends the basic *Contingent-FF* planner, aiming to incorporate different geometric reasoning procedures, observations on sensing actions, as well as human–robot collaboration within planning. The overview of the system is sketched in Figure 1. It involves three main parts: *Heuristic Computation, Space Search*, and *Conditional Plans Evaluation*.

Heuristic Computation basically provides a value which is distance to goal and promising actions for each belief state. The basic *CRPG* is initially computed and the associated relaxed plan is obtained. This plan is forwarded to the relaxed geometric reasoner determining the feasibility of actions in terms of reachability, collisions, manipulation constraints, and graspability. The heuristic value is returned along with helpful actions if such constraints are met. If a constraint is violated, the associated belief state is updated with facts describing the cause of failure, and an alternative relaxed plan is looked for. Hence, the heuristic function is informative both in terms of symbolic and geometric constraints.

Space Search maintains the basic algorithm of the *Contingent-FF* planner that is based on the *And-Or* search strategy. From each belief state, the action resulting in the state with lowest heuristic value is selected and is a candidate to be added to the conditional plans. The only difference is that the heuristic value now accounts for geometric constraints.

Conditional Plans Evaluation tries to solve motion planning for an action if possible. It considers sensing procedure to evaluate sensing actions and may assign actions, which are infeasible for robots, to operators.

If motion planning fails, the current belief state is updated with the cause of failure and the search resumes. In general, the geometric failure could be due to collisionable objects, so the literal *isCrit* is added to the belief state. If the failure is because of fixed obstacles blocking the way of reaching an object, inverse kinematic problems, or motion planning time-out problem, the literal *infeasByRob* is added to the state. In such failure, if the type of the evaluated action is either transit or open, the literal *assist* is also inserted.

Otherwise, when motion planning succeeds, the action is added to the tree-shaped conditional plans at hand. After finding the complete conditional plans, feasible actions are executed by a robot or a human operator in the real world and sensing actions are observed either using a perception module or the information provided by the human operator in run-time. Therefore, the robot plan can determine the correct branch to follow up its plan.

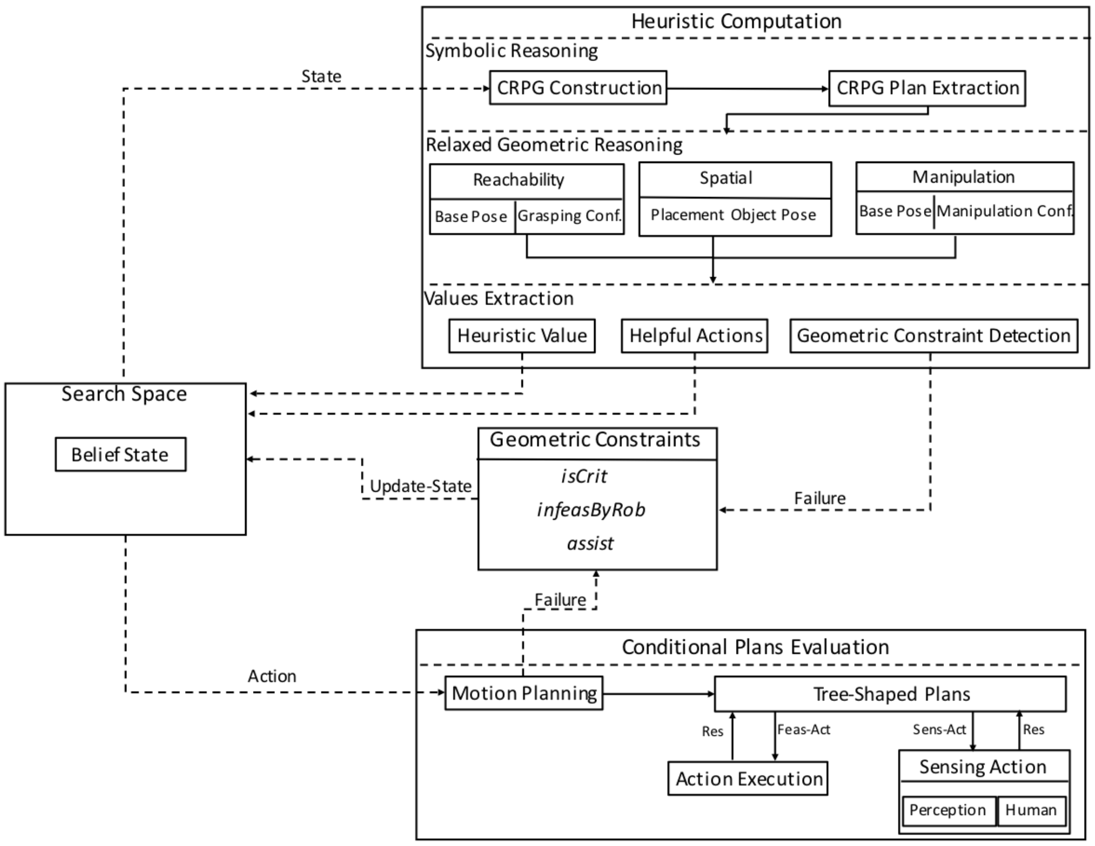


Figure 1. The proposed system overview of contingent task and motion planning using the extended version of *Contingent-FF*.

4. Relaxed Geometric Reasoning for Mobile Manipulators

Relaxed geometric reasoning is the evaluation of geometric conditions of actions with no call to motion planning. It indicates that a feasible motion is likely to be obtained for the selected actions if certain task constraints are satisfied. Therefore, the relaxed geometric reasoning process contains three modules: *reachability reasoning*, *spatial reasoning*, and *manipulation reasoning*. This set of reasoning extends our previous relaxed geometric reasoning process [19] to consider reasoning on mobile manipulation along human actions.

Reachability reasoning ($\mathcal{R}_{rch}$): This reasoning is applied for only *transit* action. To transit the robot to a target position, a feasible arm configuration and robot base pose must be first obtained. A set of robot poses is considered in the workspace of the robot. From each pose, an *Inverse Kinematic (IK)* solver is called for each candidate grasping pose, and moreover the result of *IK* is determined whether it is collision-free or not. The first collision-free *IK* solution along the corresponding grasping pose is reported if possible. Otherwise, failure is reported if there is neither *IK* solution nor collision-free configuration among a set of robot poses. Accordingly, the reasoner returns the collisionable objects.

Spatial reasoning ($\mathcal{R}_{sp}$): This reasoning is applied for the *transfer, push, open, humanOpen*, and *humanTransfer* actions. This is considered to find a valid placement for an object within a given region with no consideration of the robot. A pose is sampled, i.e., an object lies in the target region and the initial stable posture is maintained. The feasibility of the sampled pose is also determined through a collision-checking procedure to verify whether there are any collisions with other objects in the robot environment or not. If it is valid, the sampled pose is stored in the geometry details of the action which transfers the object. Otherwise, another sample will be attempted. In the case that all tried samples are not valid, failure occurs and the collisionable objects are reported. Moreover, some constraints are taken into consideration while the sample placement is accomplished. For example, in the case of the *push* action, the sample is considered in the direction in which the object is being pushed. In the case of *humanOpen* and *Open*, the valid object placement is extracted from the object feature (the box cap is assumed to have a single full-open position).

Manipulation reasoning ($\mathcal{R}_{mnp}$): This reasoning is considered to evaluate the compatibility of the grasp poses to move an object from the initial position to the final one (using $\mathcal{R}_{sp}$). The process applies $\mathcal{R}_{rch}$ reasoning to return on of the feasible ways to transfer an object from initial to final position in terms of collision-free *IK* solution. In the case that there is no possible solution meeting these conditions because of collisions, then the collisionable objects are returned. In this way, it can obtain the valid robot pose and grasping configuration when the robot manipulates an object.

Algorithm 1 describes the relaxed geometric function when applying the actions. Algorithm is detailed below:

- Reasoning about the robot actions [lines 5–15]: The *transit* action calls the reachability reasoning by the function $\mathcal{R}_{rch}$ [line 6]. The *transfer, push* and *open* actions call the spatial reasoning by the function $\mathcal{R}_{sp}$ [line 11], and then call the function $\mathcal{R}_{mnp}$ [line 13]. If the reasoning processes are successfully done, the corresponding response is set to feasible and geometric details are appended to the evaluated action [line 8] and [line 15]. On the contrary, if the failure is due to manipulatable objects, the response is set to *infeasible-criticalObjects* and the collisionable objects are stored in $\mathcal{CO}$. In other cases of failure, the response is set to *infeasible-infeasByRob* that could be because of collisions with fixed obstacles or because the *IK* module is not able to find a configuration.
- Reasoning and finding the geometric values of the human actions [lines 16–23]: The *humanTransfer* action calls the spatial reasoning by the function $\mathcal{R}_{sp}$ [line 17]. This function is responsible to find the pose of the object placement for the human action and inserts it to the action [line 19]. For the *humanOpen* action, the pose of the container object being opened is extracted from the object feature by the spatial reasoner function $\mathcal{R}_{sp}$ [line 21] and is stored into the action details [line 23].

Algorithm 1: *RelaxGeomReas*(a)

```
1  $\mathcal{CO} \leftarrow \emptyset$
2  $a.geom^+ \leftarrow \emptyset$
3  $i \leftarrow 0$
4  $Res = False$
5  if $a$.name = Transit then
6      $\{Res, Q_{rob}, Pos_{rob}, \mathcal{CO}, g\} \leftarrow \mathcal{R}_{rch}(a)$
7      if $Res = feasible$ then
8          $a.geom^+.add(Q_{rob}, Pos_{rob}, g)$
9  else if $a$.name = Transfer or Push or Open then
10     while $i < Max$ do
11         $\{Res_{sp}, \mathcal{O}^m_j(pos_{goal}), \mathcal{CO}\} \leftarrow \mathcal{R}_{sp}(a)$
12         if $Res_{sp} = feasible$ then
13             $\{Res, Q_{rob}, Pos_{rob}, \mathcal{CO}, g\} \leftarrow \mathcal{R}_{mnp}(a, \mathcal{O}^m_i(pos_{goal}))$
14             if $Res = feasible$ then
15                 $a.geom^+.add(Q_{rob}, Pos_{rob}, \mathcal{O}^m_j(pos_{goal}), g)$
16 else if $a$.name = HumanTransfer then
17     $\{Res, \mathcal{O}^m_j(pos_{goal})\} \leftarrow \mathcal{R}_{sp}(a)$
18     if $Res = feasible$ then
19         $a.geom^+.add(\mathcal{O}^m_j(pos_{goal}))$
20 else if $a$.name = HumanOpen then
21     $\{Res, \mathcal{O}^m_j(pos_{open})\} \leftarrow \mathcal{R}_{sp}(a)$
22     if $Res = feasible$ then
23         $a.geom^+.add(\mathcal{O}^m_j(pos_{open}))$
24 else
25     //$a$ is not required to be checked;
26     return $Null$
27 return $\{Res, \mathcal{CO}\}$
```

5. Contingent Heuristic Computation using Relaxed Information

Heuristic computation returns the heuristic value as well as helpful actions using relaxed symbolic along geometric reasoning from each state. Algorithm 2 explains the modified version of *Contingent-FF* heuristic computation for a given belief state S and goal $\mathcal{G}$ by taking into account geometric information. This involves three steps: computing the *CRPG* and the relaxed plan π', determining π', and computing the heuristic value and the helpful actions, as follows.

Computing the CRPG and π' [lines 1–2]: The *CRPG* graph $CRPG_{gr}$ involving state layers and action layers is built by the function CRPGConst [line 1]. The function CRPGPlan extracts π' from that graph [line 2]. The process is performed in a similar way to the standard *Contingent-FF*.

Evaluating π' [lines 3–13]: Actions in π' are sent to the relaxed geometric reasoning for the feasibility evaluation [line 5]. Basically, this process tries to figure out whether there is any feasible world to meet the action conditions or not as we proposed in [19]. Upon failure, the function MaxUp [line 9] determines whether a predefined maximum number of trials is reached or not to update the belief state and find another relaxed plan. If updating the state is required, the feedback of the geometric reasoner is evaluated. In the case of failure because of *infeasible-criticalObjects*, the literal *isCrit(*$\mathcal{CO}$*,* $\mathcal{O}'$*, Pos)* with critical objects is added to the current belief state. Otherwise, the failure is because of *infeasible-infeasByRob* and the literal *infeasByRob(*$\mathcal{R}$*,* $\mathcal{O}'$*, Pos)* is added to the state. In this case, if the type of action is either transit or open, the literal *assist(Human,* $\mathcal{O}'$*, Pos)* is also added to the state.

Computing the heuristic value and with helpful actions [lines 14–15]: In the case that the relaxed plan is geometrically feasible with respect to the geometric reasoning evaluation, the heuristic value along the helpful actions are achieved. The function HValue extracts the heuristic value $h(S)$ [line 14] and the function HelpAct reports helpful actions $H(S)$ [line 15] as the generic *Contingent-FF*.

Algorithm 2: $CRPG(S, \mathcal{G})$

```
1  $CRPG_{\text{gr}} \leftarrow \text{CRPGConst}(\mathcal{G})$
2  $\pi' \leftarrow \text{CRPGPlan}(CRPG_{\text{gr}})$
3  foreach $\{a' \in \pi'\}$ do
4      while $True$ do
5          $\{Res, \mathcal{CO}\} \leftarrow \text{RelaxGeomReas}(a')$
6          if $Res = True$ then
7              break
8          else
9              if $\text{MaxUp}(S) < Max$ then
10                 $S \leftarrow \text{UpState}(S, Res, \mathcal{CO})$
11                 return $\text{CRPG}(S, \mathcal{G})$
12             else
13                 return $\{\infty, \emptyset\}$
14 $h(S) \leftarrow \text{HValue}()$
15 $H(S) \leftarrow \text{HelpAct}()$
16 return $\{h, H(S)\}$
```

An example is considered to show how geometric constraints are captured and handled during the heuristic computation. The initial scene of the example is shown in Figure 2 where the robot is required to move *Can A* inside *Box*. To make the problem challenging, it is assumed that top grasps are not allowed and some side grasping poses are considered for each can. Several task constraints are imposed, e.g., there is no direct collision-free motion to reach *Can A*, and also the robot is not able to open *Box* and needs an operator assistant.

The computation of the heuristic process in terms of geometric feasibility is represented in Figure 3. The corresponding physical world for each relaxed plan has been shown also. Figure 3a shows the initial relaxed plan extracted. When the first action is forwarded for the relaxed geometric reasoning, the reachability reasoner fails. This is because when the inverse kinematic module checks side grasping poses considered for the box cap, all retrieved joint configurations have collisions with the box object. The geometric reasoning process is done by the proposed function RelaxGeomReas in Algorithm 2 that is added to the basic planner.

To handle this task constraint, the predicates *infeasByRob(tiago, box, posBox)* and *assist(person, box, posBox)* are asserted to the planning state by the associated reasoning process. The updating sate step is done by the proposed function UpState as it lets the planner know the detected constraints of the environment. Figure 3b shows the next heuristic computation taking into account the task constraint. In this case, the spatial reasoner module successfully finds the geometric state of *Box* after applying the action *humanOpen*. However, the reachability reasoner reports a failure for evaluating the *transit* action for the object *Can A* due to collision between the robot arm and *Can B*. This object is marked as a critical object, so the state is updated with the predicate *isCrit(can B, can A, pos A)*. The heuristic computation is again repeated, and finally the reasoning processes can correctly find feasible geometric details for the actions. This process results in geometrically feasible heuristic computation.

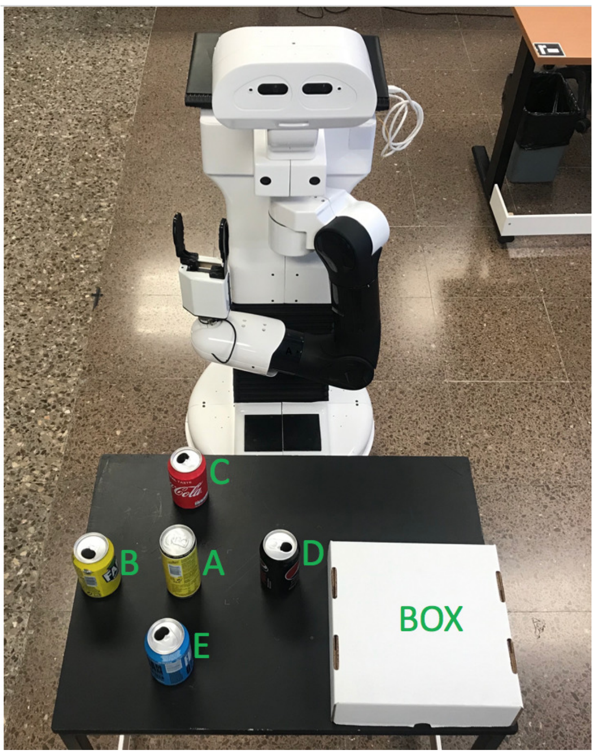
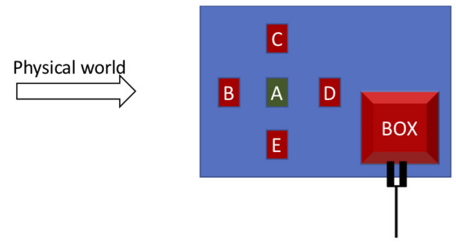


Figure 2. The initial scene where the robot requires placement of the object A within the box in the presence of geometric constrains.

π'	Relaxed geometric			Failure
	$\mathcal{R}_{rch}$	$\mathcal{R}_{sp}$	$\mathcal{R}_{mnp}$	
Transit-Box	**False**	–	–	**Collision with Box**
Transit-CanA	–	–	–	–
Open-Box	–	–	–	–
Transfer-CanA	–	–	–	–

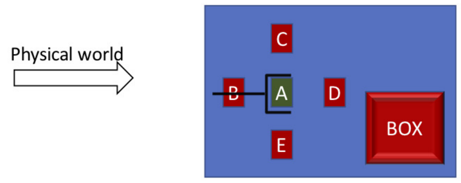
(a)



$\Downarrow$ Re-computing heuristic

π'	Relaxed geometric			Failure
	$\mathcal{R}_{rch}$	$\mathcal{R}_{sp}$	$\mathcal{R}_{mnp}$	
HumanOpen-Box	–	**True**	–	–
Transit-CanA	**False**	–	–	**Collision with CanB**
Transfer-CanA	–	–	–	–

(b)



$\Downarrow$ Re-computing heuristic

π'	Relaxed geometric			Failure
	$\mathcal{R}_{rch}$	$\mathcal{R}_{sp}$	$\mathcal{R}_{mnp}$	
HumanOpen-Box	–	**True**	–	–
Transit-CanB	**True**	–	–	–
Transfer-CanB	–	**True**	**True**	–
Transit-CanA	**True**	–	–	–
Transfer-CanA	–	**True**	**True**	–

(c)

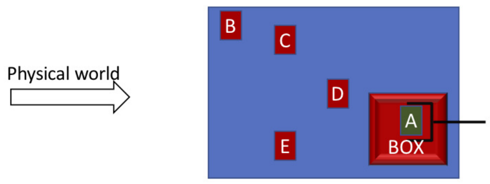


Figure 3. The steps of the computation of heuristic using the relaxed geometric reasoning and the corresponding physical world. The information highlighted in bold shows the relaxed planning actions which have been currently tested by the proposed relaxed geometric reasoning. Others are those which have not been tested yet. True and false values show whether the reasoner is successful or failed. (**a**) The *transit* action to reach the Box fails. (**b**) The *transit* action for the Can A fails due to collision with other objects. (**c**) The final geometrically feasible heuristic computation.

6. Tree-Based Planning using Search Space

The *And-Or* search procedure as considered in the *Contingent-FF* planner is used to result in a feasible manipulation plan. The heuristic computation has been modified to incorporate geometric check and, moreover, selected actions must be evaluated using motion planning. The process is represented in Algorithm 3.

The algorithm gets $\mathcal{T}$ as input and outputs π if possible. First, the trials counter *trial* is set [line 1] and the state S_i is the initial belief state [line 4]. The function Search performs the standard search mechanism as *Contingent-FF* does [line 6]: it provides the next state using the transit function to visit S_{i+1} along the promising applicable action(s) with H_{S_i}. This step is done with the modified CRPG function (see Algorithm 2), by taking geometric constraints into account.

In the case that H_{S_i} does not exist [line 7], the algorithm performs another search from the beginning. Until the maximum number of iterations is not reached [line 9], the process is repeated with the initial state updated by the function UpdateInitState [line 11]. If the maximum number of trials is reached, the process returns failure [line 14].

Algorithm 3: The Proposed Planning Algorithm

```
inputs : $\mathcal{T}$=⟨$\mathcal{D}, \mathcal{S}_0, \mathcal{G}$⟩, $\mathcal{D}$=⟨$\mathcal{A}, \Omega, \mathcal{F}, \mathcal{W}, S_g$⟩
output: $\pi$
1  $trial \leftarrow 0$
2  $i \leftarrow 0$
3  $\pi \leftarrow \emptyset$
4  $S_\mathrm{i} \leftarrow S_\mathrm{init}$
5  while $\mathcal{G} \not\subseteq S_i$ do
6      $\{H_{S_\mathrm{i}}, S_{i+1}\} \leftarrow$ Search$(S_i, \mathcal{G}, \mathcal{A}, \Omega)$
7      if $H_{S_i} = \emptyset$ then
8          $trial \leftarrow trial + 1$
9          if $trial < Max$ then
10             $i \leftarrow 0$
11             $S_i \leftarrow$ UpdateInitState()
12             Continue
13         else
14             return fail
15     else
16         if $H_{S_i} \notin \Omega$ And $H_{S_i}.name \neq$ HumanTransfer And $H_S.name \neq$ HumanOpen then
17             $\{\mathcal{Q}, Res, \mathcal{CO}\} \leftarrow$ MotionPlanner$(H_{S_\mathrm{i}})$
18         if $H_{S_i} \in \Omega$ Or $H_{S_i}.name =$ HumanTransfer Or $H_S.name =$ HumanOpen Or $Res = feasible$ then
19             $\pi$.append$(H_{S_\mathrm{i}})$
20         else
21             $S_i \leftarrow$ UpdateState$(Res, \mathcal{CO})$
22             Continue
23     $i \leftarrow i + 1$
24 return $\pi$
```

For those actions that either do not belong to the set of sensing actions and are not assigned to human, the MotionPlanner function is used to compute a collision-free path for the currently selected action(s) [line 17]. If a path is found, *Res* is set to *feasible* and the path $\mathcal{Q}$ is returned. Afterwards, π is appended with the sensing, human, or normal action(s) [line 19]. In the case of failure due to *infeasible-criticalObjects*, the literal *isCrit($\mathcal{CO}$, $\mathcal{O}'$, Pos)* with critical objects is added to the current belief state. Otherwise, the failure is because of motion planning time-out problem or collisionable fixed obstacles, the type of failure is *infeasible-infeasByRob* and the literal *infeasByRob($\mathcal{R}$, $\mathcal{O}'$, Pos)* is added to the state. In this case, if the type of action is either transit or open, the literal *assist(Human,* $\mathcal{O}'$*, Pos)* is also added to the state.

An example is considered to illustrate how the geometrically feasible conditional plan is obtained offline under belief information of the initial state. The scene depicted in Figure 4 shows the initial belief state of the mobile manipulation problem, where the color of the gray cylinder is uncertain (it could be red or green), it is not known whether the can is filled or not, nor if the containers are open or closed. The goal is to transfer the cylinder A to either the red or the green tray. Some particular placements regions allocated for the manipulatable objects if required:

- The green cylinders must be placed on the green tray.
- The red cylinders must be placed over the red tray.
- The blue cylinders must be placed within the containers.
- The can objects may be optionally placed anywhere over the table.

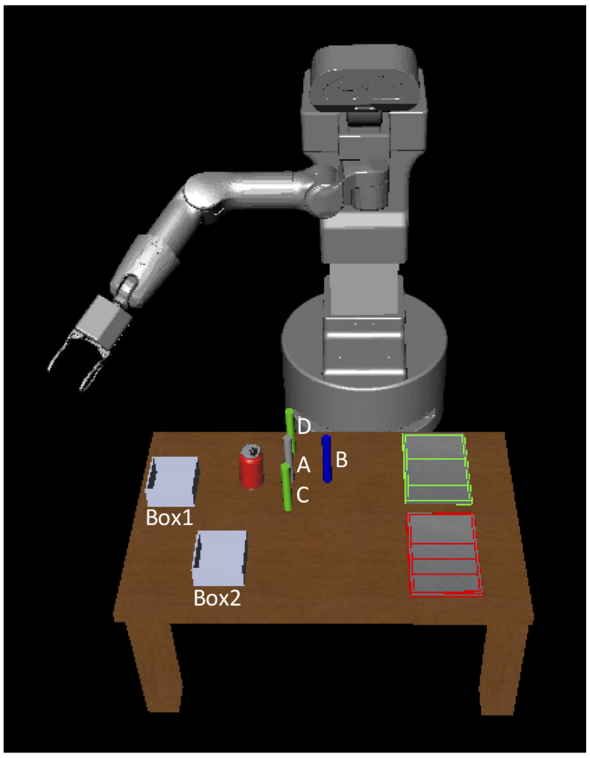


Figure 4. The manipulation example where the goal is to transfer cylinder A to one of the trays with respect to its color.

The complete conditional plan is represented in Figure 5 that is obtained by Algorithm 3. It is briefly discussed how this geometrically feasible plan is obtained. While the planning process is taking place, there are several challenges in terms of geometric constraints which are captured and handled by the proposed geometry reasoner. These steps are mainly done using the Search function which internally calls Algorithm 2. To reach the target object, the reachability reasoning process, place in the function RelaxGeomReas, detects cylinder B and reports that the object is blocking the way of reaching object A in the heuristic computation. Therefore, the predicate *isCrit(cylinderB, cylinderA, posA)* is inserted to the initial belief state of the planner using the function updating the belief state. This predicate says that cylinder B blocks the way of reaching cylinder A.

Furthermore, when the robot attempts to find a feasible configuration for opening box 1 in the case that the box is closed, the reachability reasoner fails due to colliding with the box. Here, it is the case that robot needs to ask a human operator for collaboration. Accordingly, the reasoner appends the predicates *infeasByRob(tiago, box1, posBox1)* and *assist(person, box1, posBox1)* to the corresponding state. The *humanOpen* action then appears.

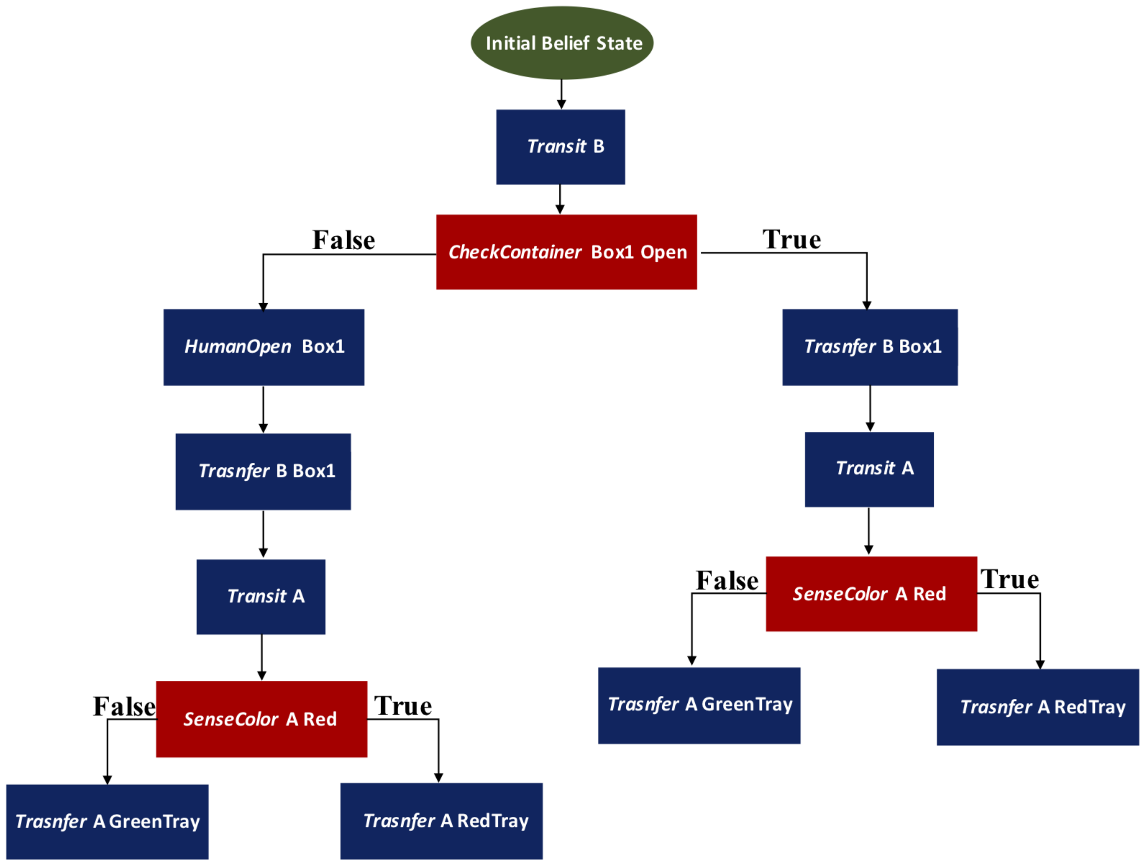


Figure 5. The conditional plan results from the proposed planning process. The actions highlighted with the blue color are the ones assigned to the robot or a human operator. Some important actions parameters are represented. The actions specified by the red color are sensing actions.

7. Manipulation Plan Execution using Sensing and Human Interaction

When the manipulation conditional plan is achieved, it will be forwarded to for the execution module. Algorithm 4 outlines the process of actions execution performed by the robot or human, and calls to the sensing actions. The conditional plan is initialized from its root [line 1]. For each action of the plan, its type first identified whether it is execution or sensing one. In the case of execution action, if it has to be executed by human, the function executeByHuman asks a person to do the corresponding action [line 5] and an operator then sends a command to the robot that the action has been done successfully. Otherwise, the action is executed by the robot [line 7].

On the other hand, if the type of action becomes sensing, the function senseAct determines the binary value of the sensing action which is *True* or *False*. This is done using the perception module allocated for the robot. Depending on the type of uncertainty, the function may request to human or activate a sensing module to observe the action value. Regarding the *CheckCan* or *CheckOpen* sensing actions, human information is used, while a sensing module is used for the *SensePose* and *SenseColor* sensing actions.

Algorithm 4: Manipulation Plan Execution

```
inputs : $\pi$
1  initializePlan($\pi$)
2  foreach $\{H_S \in \pi\}$ do
3      if $H_S \in \mathcal{A}$ then
4          if $H_S.name =$ HumanTransfer Or HumanOpen then
5              executeByHuman($H_S$)
6          else
7              executeByRob($H_S$)
8      else
9          $Res$ = senseAct($H_S$)
10         selectBranch($\pi, H_S, Res$)
```

8. Empirical Results and Discussion

This section describes some manipulation problem solved using the proposed framework, and the implementation issues. The mobile robot considered is *TIAGo*. It has 7 degrees of freedom arm, equipped with a gripper, mounted on a mobile platform through a lift torso.

The executive simulated result of the manipulation problem represented in Figure 4, called *Problem-1*, is shown in Figure 6. For the domain of the problem, a number of actions is considered for the robot being *transit, transfer, open,* and *push* along with some actions for an operator that are *humanTransfer* and *humanOpen*. Actions are selected according to the planning mechanism in terms of symbolic and geometric reasoning. We assume that the values of the uncertainty information are provided in run-time in simulation. Therefore, the executive plan is provided below:

Executive Plan: { *Transit-B, CheckContainer-Box1-Open (False), HumanOpen-Box1, Transfer-B-Box1, Transit-A, SenseColor-A-Red (True), Transfer-A-RedTray* }

The states represented in the figure are classified as follows:

(a) is the initial belief state of the robot and environment.
(b) is the state where the robot applies *transit* action to reach cylinder B.
(c) is the state where the sensing action *CheckContainer-Box1-Open* showed that Box 1 is currently closed.
(d) is the state where the *HumanOpen* action is executed as the robot is not capable enough to open the box.
(e) is the state where the robot places cylinder B within box 1.
(f) is the state where the robot transits to cylinder A.
(g) is the state where the sensing action *SenseColor-A-Red* showed that the cylinder is actually red.
(h) is the state where the robot moves its base and the arm configuration to place cylinder A over the red tray.

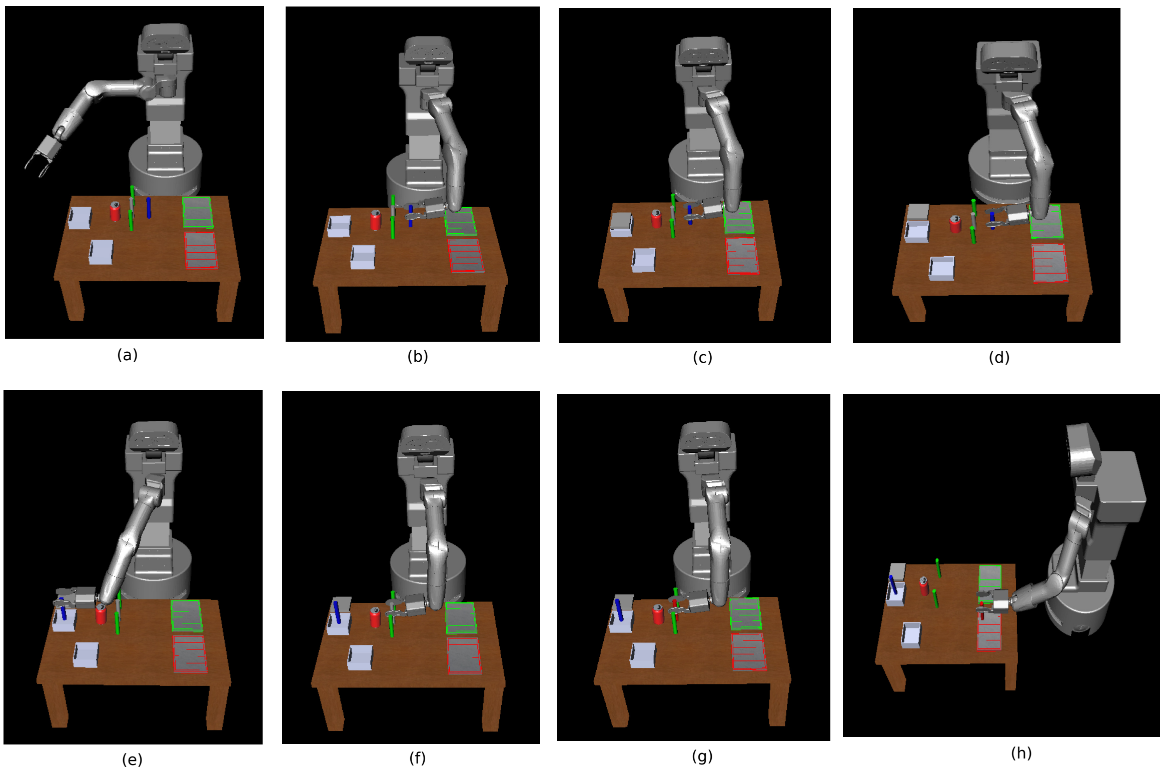


Figure 6. The simulation results of the executive plan performed by the *TIAGo* robot: (**a**) is the initial robot and environment state, (**b**) is the state after *Transit* action towards cylinder B, (**c**) is the result of the sensing action *CheckContainer-Box1-Open*, (**d**) is the state after applying *HumanOpen* action, (**e**) is the state after the robot executes the *Transfer* action for cylinder B, (**f**) is the state when the robot transits to cylinder A, (**g**) is the state resulting from the sensing action *SenseColor-A-Red*, and (**h**) is the state when the robot place cylinder A on the associated tray.

In addition, the proposal has been evaluated for other cluttered problems where the robot needs to sort objects according to the colors. Regarding the action domain, Robot actions are *transit* and *transfer*, and the action template *humanTransfer* is considered for an operator. The problem represented in Figure 7, called *Problem-2*, shows the initial and goal states of manipulation where the green and red objects must be located on the green and red regions respectively. The red object is not initially located on the table. The pink region is considered on the robot workspace where an operator can transfer objects. The planning uncertainties are the color of the green object which could be actually green or red and the location of the red object which could be on the robot table or in the human workspace. Therefore, the *humanTransfer* action is applied to transfer the object to the robot workspace as the robot is not allowed to move to the human workspace. The final executable plan would be to transfer the green object to the target placement region by the robot. It then looks for the red object and figures out the object is not located on the table and asks an operator to transfer the object. The *humanTransfer* action is selected in the conditional plan, so the requested object is transferred to the robot workspace. The operator updates the robot knowledge through the robot system terminal. The robot is aware that the human action has been successfully performed, and afterwards it travels to grasp the object. Eventually, the robot transfers the object to the target region.

The proposed approach has been tested for similar problems by increasing the number of objects and varying color and/or location uncertainties. The problems performance are represented in Table 1 in terms of conditional and executive plan length, and moreover planning time. *Problems-3* includes a cluttered problem where there are nine objects and three of them need to be sorted. Similar uncertainty of *Problem-2* is considered regarding the color and location of objects. *Problem-4* is the one where 12 objects exist and four of them must be sorted. In this case, the uncertainty information like the objects color and locations are considered for more objects.

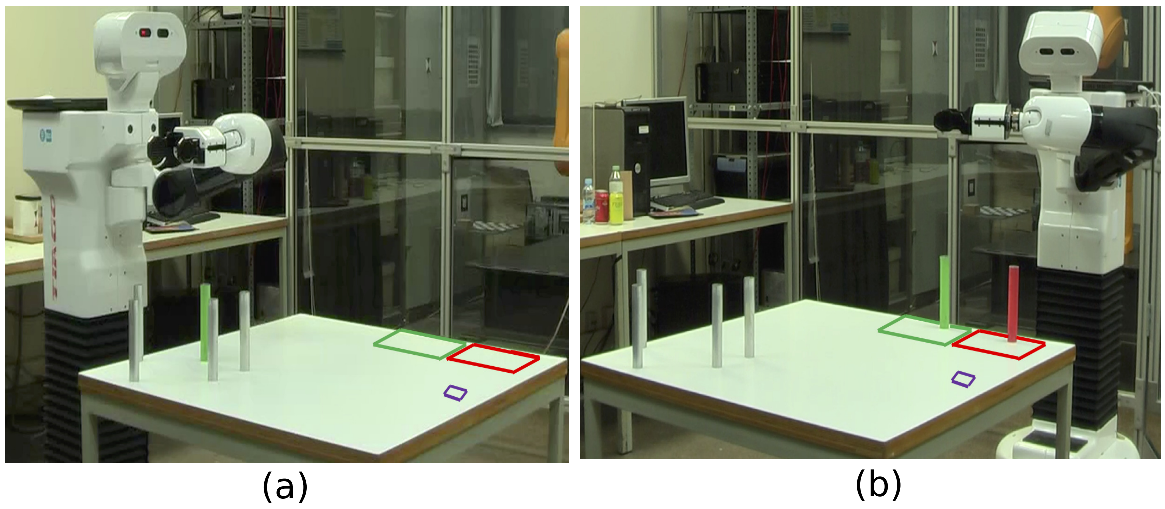


Figure 7. The manipulation example where green and red objects must be placed in the green and red regions. (**a**) shows the initial state of the problem. (**b**) shows the final state of the problem. The red object is not initially located in the robot workspace. The pink region is the place where human can transfer objects to the robot workspace. The solution can be visualized here: https://sir.upc.es/projects/ontologies/GreenRedHuman.mp4. The solution for the case that the red object is initially located on the table is visualized here: https://sir.upc.es/projects/ontologies/TiagoRedGreenRob.mp4.

Table 1. The conditional plan and executive plan length in terms of number of sensing and executive actions and planning time in seconds for the evaluated problems.

Problem	**Conditional Plan**		**Executive Plan**		**Planning Time**
	Sensing	**Executive**	**Sensing**	**Executive**	
Problem-1	3	10	2	5	35
Problem-2	3	13	2	5	59
Problem-3	3	19	2	8	163
Problem-4	7	41	3	12	449

Concerning the implementation framework, four components are considered: task planning, relaxed geometric reasoning, motion planning, and executive module. Task planning is developed using a modified version of the *Contingent-FF* planner coded in C++. All the action templates are described using *PDDL* by considering *ADL (Action Description Language*, ref. [28]) enabling us to define operators in a more compact way, using quantifiers and conditional effects. There is not any pre-processing step to compute geometric details of actions and they are computed and assigned during the manipulation planning process.

We use *The Kautham Project* [29], a C++-based open-source tool for motion planning that enables planning under geometric and kinodynamic constraints for relaxed geometric reasoning and motion planning. It uses the *Open Motion Planning Library (OMPL)* [30] as a core set of sampling-based planning algorithms. In this work, the *RRT-Connect* [31] motion planner is used for motion planning. This planner is one of the most efficient motion planners, but it does not guarantee optimal motions. *The Kautham Project* involves different collision checking modules to detect robot-object and object-object collisions, and features a placement sampling mechanism to find feasible object poses in the workspace. Relaxed geometric reasoning uses these modules to find feasible sample geometric instances for symbolic actions. The executive module uses a sensing module which uses the 3D camera mounted inside the *TIAGo* robot, and also some components provided by *PAL Robotics* to send a motion path to the robot. The communication between task, relaxed geometric reasoning, motion planning, and executive modules is done via *Robotic Operating System (ROS)* [32].

9. Conclusions

This paper has proposed a contingent-based task and motion planning approach able to cope with high-dimension mobile manipulation problems in the presence of high-level uncertainty and human interactions (referred to the sharing of knowledge and to collaborative actions which are out of the robot capabilities). For this purpose, the basic *Contingent-FF* planner has been modified to include robot action reasoning, human–robot collaboration, and state observation. A set of geometric reasoning processes has been offered to the planning process to capture the task constraints imposed in the robot environment and to update belief state while task planning is done. Moreover, some modules linked with the human knowledge along with the perception system, have been also designed to observe the binary outcomes of actions. It is worth noting that the proposed approach results in a tree-shaped conditional plan which is geometrically feasible regardless of the values of sensing actions.

To evaluate the proposed approach, several manipulation tasks have been executed in simulation and real environments to show the way of tackling human–robot interactions, and identifying and handling both geometric constraints and high-level uncertainty. Problems performance has been reported in terms of the length of the manipulation plan and planning time, considering an increasing number of objects. In all the cases, the robot in collaboration with the human operator has been able to solve the tasks despite the uncertainty and the constraints.

Future work will concentrate on manipulation tasks also subject to low-level geometric uncertainty, its effects in sensing and how it is transferred to task planning.

Author Contributions: A.A. and J.R. conceived the theoretical contributions, A.A. wrote the paper and implemented the whole framework, being assisted by M.D. for the perception part. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Spanish Government through the project DPI2016-80077-R. Mohammed Diab is supported by the Spanish Government through the grant 2017.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hoffmann, J.; Brafman, R. Contingent planning via heuristic forward search with implicit belief states. In Proceedings of the International Conference on Automated Planning and Scheduling, Monterey, CA, USA, 5–10 June 2005.
2. Stilman, M.; Schamburek, J.U.; Kuffner, J.; Asfour, T. Manipulation planning among movable obstacles. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3327–3332.
3. Stilman, M.; Kuffner, J. Planning among movable obstacles with artificial constraints. *Int. J. Robot. Res.* **2008**, *27*, 1295–1307. [CrossRef]
4. Rodríguez, C.; Suárez, R. Combining motion planning and task assignment for a dual-arm system. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 4238–4243.
5. Zarandi, M.F.; Mosadegh, H.; Fattahi, M. Two-machine robotic cell scheduling problem with sequence-dependent setup times. *Comput. Oper. Res.* **2013**, *40*, 1420–1434. [CrossRef]
6. Foumani, M.; Smith-Miles, K.; Gunawan, I.; Moeini, A. A framework for stochastic scheduling of two-machine robotic rework cells with in-process inspection system. *Comput. Ind. Eng.* **2017**, *112*, 492–502. [CrossRef]
7. Ghallab, M.; Nau, D.; Traverso, P. *Automated Planning: Theory & Practice*; Elsevier: San Francisco, CA, USA, 2004.
8. Lagriffoul, F.; Andres, B. Combining task and motion planning: A culprit detection problem. *Int. J. Robot. Res.* **2016**, *35*, 890–927. [CrossRef]

9. Dantam, N.; Kingston, Z.K.; Chaudhuri, S.; Kavraki, L.E. Incremental Task and Motion Planning: A Constraint-Based Approach. In Proceedings of the Robotics: Science and Systems, Cambridge, MA, USA, 18–22 June 2016.
10. He, K.; Lahijanian, M.; Kavraki, L.E.; Vardi, M.Y. Towards manipulation planning with temporal logic specifications. In Proceedings of the International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015; pp. 346–352.
11. Akbari, A.; Muhayyudin; Rosell, J. Task and Motion Planning Using Physics-based Reasoning. In Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation, Luxembourg, 8–11 September 2015.
12. Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S.; Abbeel, P. Combined task and motion planning through an extensible planner-independent interface layer. In Proceedings of the IEEE International Conference on Robotics and Automation Robotics and Automation, Hong Kong, China, 31 May–5 June 2014; pp. 639–646.
13. Diab, M.; Akbari, A.; Muhayyuddin; Rosell, J. PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation. *Sensors* **2019**, *19*, 1166. [CrossRef] [PubMed]
14. Cambon, S.; Alami, R.; Gravot, F. A hybrid approach to intricate motion, manipulation and task planning. *Int. J. Robot. Res.* **2009**, *28*, 104–126. [CrossRef]
15. Garrett, C.R.; Lozano-Pérez, T.; Kaelbling, L.P. FFRob: An efficient heuristic for task and motion planning. In *Algorithmic Foundations of Robotics XI*; Springer: Cham, Switzerland, 2015; pp. 179–195.
16. Akbari, A.; Muhayyuddin; Rosell, J. Reasoning-based Evaluation of Manipulation Actions for Efficient Task Planning. In Proceedings of the ROBOT2015: Second Iberian Robotics Conference, Lisbon, Portugal, 19–21 November 2015.
17. Akbari, A.; Muhayyudin; Rosell, J. Task Planning Using Physics-based Heuristics on Manipulation Actions. In Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation, Berlin, Germany, 6–9 September 2016.
18. Akbari, A.; Muhayyuddin; Rosell, J. Knowledge-oriented task and motion planning for multiple mobile robots. *J. Exp. Theor. Artif. Intell.* **2019**, *31*, 137–162. [CrossRef]
19. Akbari, A.; Lagriffoul, F.; Rosell, J. Combined heuristic task and motion planning for bi-manual robots. *Auton. Robot.* **2018**, 1–16. [CrossRef]
20. Bryce, D.; Kambhampati, S.; Smith, D.E. Planning graph heuristics for belief space search. *J. Artif. Intell. Res.* **2006**, *26*, 35–99. [CrossRef]
21. Petrick, R.P.; Bacchus, F. Extending the Knowledge-Based Approach to Planning with Incomplete Information and Sensing. In Proceedings of the International Conference on Automated Planning and Scheduling, Whistler, BC, USA, 3–7 June 2004; pp. 2–11.
22. Bonet, B.; Geffner, H. Planning under partial observability by classical replanning: Theory and experiments. In Proceedings of the IJCAI Proceedings-International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, 16–22 July 2011; Volume 22, p. 1936.
23. Shani, G.; Brafman, R.I. Replanning in domains with partial information and sensing actions. *IJCAI* **2011**, *2011*, 2021–2026.
24. Maliah, S.; Brafman, R.I.; Karpas, E.; Shani, G. Partially Observable Online Contingent Planning Using Landmark Heuristics. In Proceedings of the International Conference on Automated Planning and Scheduling, Portsmouth, NH, USA, 21–26 June 2014.
25. Gaschler, A.; Petrick, R.; Kröger, T.; Knoll, A.; Khatib, O. Robot task planning with contingencies for run-time sensing. In Proceedings of the International Conference on Robotics and Automation Workshop on Combining Task and Motion Planning, Karlsruhe, Germany, 6–10 May 2013.
26. Nouman, A.; Yalciner, I.F.; Erdem, E.; Patoglu, V. Experimental evaluation of hybrid conditional planning for service robotics. In Proceedings of the International Symposium on Experimental Robotics, Tokyo, Japan, 3–6 October 2016; pp. 692–702.
27. Hoffmann, J.; Nebel, B. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.* **2001**, 253–302. [CrossRef]

28. Pednault, E.P.D. ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. In Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning, Toronto, ON, Canada, 15–18 May 1989; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1989; pp. 324–332.
29. Rosell, J.; Pérez, A.; Aliakbar, A.; Muhayyuddin; Palomo, L.; García, N. The Kautham Project: A teaching and research tool for robot motion planning. In Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation, Barcelona, Spain, 16–19 September 2014.
30. Sucan, I.; Moll, M.; Kavraki, L.E.; others. The open motion planning library. *Robot. Autom. Mag.* **2012**, *19*, 72–82. [CrossRef]
31. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 995–1001.
32. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the International Conference on Robotics and Automation Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.