Extending Task and Motion Planning with Feasibility Prediction: Towards Multi-Robot Manipulation Planning of Realistic Objects

Smail Ait Bouhsain[1], Rachid Alami[1] and Thierry Siméon[1]

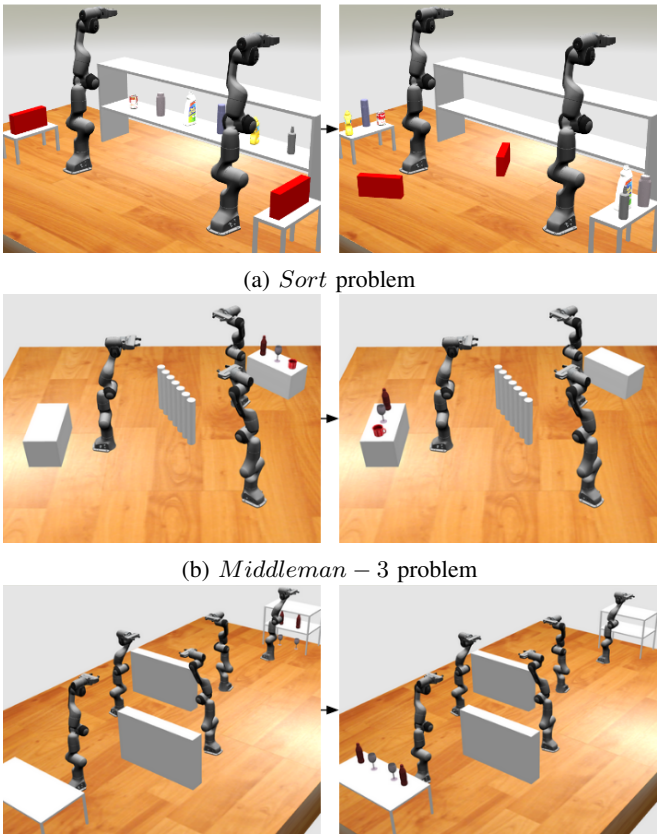
***Abstract*— The hybrid discrete/continuous nature of task and motion planning (TAMP) results often in a combinatorial explosion. This challenge is even more pronounced in multi-robot TAMP problems due to the increase in dimensionality of the action space. Previous works use action feasibility prediction as a heuristic to accelerate TAMP. However, these methods are limited to box-shaped objects and specific single or dual robot settings. In this paper, we propose a feasibility-enabled multi-robot TAMP algorithm capable of tackling complex multi-robot manipulation problems. Also, we expand on our previous work on action and grasp feasibility prediction [1] by extending its use to mesh-shaped objects. We demonstrate the performance of our method compared to a non feasibility-informed baseline, and show its ability to handle TAMP problems requiring the collaboration of multiple robots.**

I. Introduction

Task and motion planning (TAMP, see e.g survey [2]) in robotics involves finding a sequence of steps a robot should take to achieve a goal, along with their corresponding motions. It mixes discrete symbolic planning and continuous geometric planning. This combination results in a high combinatorial complexity making the search for a geometrically feasible task plan tedious, time consuming and, in some cases, unsolvable in a reasonable amount of time. These challenges do not only come from the combinatorial complexity of the search, but also the high time cost of calling geometric planners to verify the feasibility, particularly infeasibility, of actions and plan their motions.

Recent works [3]–[6] leverage learning methods in order to tackle this shortcoming of geometric planners. They propose to learn to predict the feasibility of actions without the need for querying geometric planning. These feasibility predictions can then be used as heuristics during task and motion planning. In a previous work [1], we propose the **AGFP-Net** neural network, which predicts the feasibility of pick and place actions in 3D environments, as well as the feasibility of subsets of grasps. This model is integrated in a feasibility-informed TAMP algorithm which uses feasibility predictions as a heuristic to accelerate planning. The method proposed in [1] is however limited to single-robot TAMP problems and is not able to handle multi-robot settings. Also, it is limited to box-shaped objects and obstacles.

In this paper, we tackle these limitations by proposing a new feasibility-informed multi-robot TAMP algorithm. It takes advantage of **AGFP-Net** for efficient action and grasp feasibility predictions as a heuristic for accelerating planning time. It also introduces a method for estimating the feasibility of collaborative actions involving more than a single robot. Moreover, we propose a method for extending the use of **AGFP-Net** to mesh-shaped objects, allowing feasibility prediction on common everyday objects. To the best of our knowledge, our approach is the first to tackle multi-robot TAMP in 3D "realistic" environments (i.e with support surfaces at different elevations/overlapping, vertical and horizontal obstacles and mesh-shaped objects).



(a) $Sort$ problem

(b) $Middleman - 3$ problem

(c) $Middleman - 5$ problem

Fig. 1: Visualization of the initial and goal states of 3 of the multi-robot TAMP problems solved by our method.

II. Related Work

A. Classical TAMP

Early TAMP research [7]–[15] focused on the geometric aspect, approaching the problem as a multi-modal motion planning problem, which involves planning among multiple motion modes. Many current TAMP techniques [16]–[23] integrate a task planner with a geometric planner, relying

[1]LAAS-CNRS, Toulouse, France, {saitbouhsa, alami, simeon}@laas.fr

This work is partially supported by the EU-funded project euROBIN under grant agreement no. 101070596 (https://www.eurobin-project.eu/)

on geometric backtracking to connect the two. These approaches, nevertheless, face challenges due to the combinatorics of the hybrid discrete/continuous search. and the heavy time consumption of geometric planning, as verifying the feasibility of symbolic plans still necessitates a large number of costly calls to the geometric planner.

B. Learning for TAMP

In recent years, learning methods became increasingly popular in TAMP [3]–[6], [24]–[34], [1]. Many works aim at providing a learned heuristic to a TAMP planner, such as evaluating action affordances [29], leveraging experience to propose promising actions [26], [27], or providing fast geometric feedback to the task planner thanks to action feasibility prediction [3]–[6]. These methods are however limited to tabletop environments. In [1], [33], we propose a learning approach for predicting action and grasp feasibility in 3D environments. As most feasibility prediction methods, ours is limited to box-shaped objects. Wells et al. [3] propose to use SVMs to predict the feasibility of pick and place actions on mesh-shaped objects. These objects are represented using hand-designed features, which makes the generalizability to various shapes difficult. Park et al. [35] propose a neural network for predicting the feasibility of grasp modes in scenes containing one mesh-shaped object and a number of obstacles. However, this method is object-centric, meaning that the proposed model needs to be retrained for each new object shape. In this paper, we extend our action and grasp type feasibility prediction neural network [1] to mesh-shaped objects, while maintaining generalizability to 3D environments containing multiple objects of various shapes.

C. Multi-Robot TAMP

Multi-robot TAMP suffers from a higher combinatorial complexity due to the increased number of robots, and thus the increased dimensionality of the action space [36]. Previous works [16], [37]–[41] propose various approaches for task and motion planning in multi-robot settings such as graph-based and MDP-based techniques. These methods suffer however from a combinatorial explosion as the number of robots increases, resulting in a notable increase in the already high number of calls to the geometric planner. In this paper, we propose to accelerate multi-robot TAMP using robot-centric feasibility predictions, allowing the planner to mitigate the combinatorial explosion due to the presence of multiple robots. We focus on sequential motion planning in which objects are passed between two robots using a pair of pick-place actions, rather than coordinated motion planning where objects are exchanged between robots using handovers. Driess et al. [4], [28] train a neural network to predict action feasibility on box-shaped objects in tabletop environments using two robotic arms. This learning approach is limited, nevertheless, to the multi-robot setting it is trained on, requiring a new training if the number of robots increases, or the robots' placements change. Our proposed feasibility prediction framework does not depend on these parameters, since it queries the neural network for each robot individually. Park et al. [35] propose a similar method, but focus on environments containing one movable object only.

III. Problem Description

We tackle model-based manipulation planning problems where the goal is to rearrange a set of movable objects in a 3D environment, comprising n_R robot arms with fixed bases, n_{ss} stable support surfaces, n_{obs} fixed obstacles and n_O movable objects. The shapes, dimensions, initial poses of all objects (fixed and movable) and the initial configurations of all robots are fully known.

We define the state of the environment as the configuration of each robot together with the support surface and pose of all movable objects. The configuration of a robot r at state s is denoted $s(r)$, whilst the configuration of an object O is denoted $s(O)$. A transition between two states s and s' is defined as an action a which involves a specific robot r picking an object O from its configuration in s, and placing it at a new configuration $\mathbf{q}$, which becomes the configuration of O in s'. The action a can be explicitly represented as:

$$a = Move(r, O, s(O) \rightarrow \mathbf{q}) \tag{1}$$

The solution to the TAMP problem is a sequence of actions τ and their corresponding motions Π, which brings the environment from the initial state s_0 to a goal state s_{goal}. We define 3 types of actions at the symbolic level. The first is a $Goal$ action, which moves an object to its goal state. The second is a $Temp$ action, which places an object at a temporary placement. The third is a $Pass$ action, which aims at placing an object inside the intersection between two robots' workspaces, so that the second robot can reach it. We consider the workspace of a robot r, denoted $W(r)$, as a sphere centered at the first joint of the robot, with a radius equal to its reach[1].

IV. Background

A. Neural Network

In [1], we propose a learning approach based on the Action and Grasp Feasibility Prediction neural NETwork (**AGFP-Net**). Given a representation of the current state of the environment and the object of interest, it simultaneously predicts the probability of feasibility of a $Pick$ or $Place$ action denoted as p_F, and the feasibility of 6 classes of grasps grouped as the vector of probabilities $\mathbf{p}_G$. Each grasp class is the set of grasps related to a side (e.g. top, front, left) from which the object is approached by the robot. The 3D environment is represented using 5 depth images corresponding to different views of the scene, which show all the objects in the workspace of the robot except the object to manipulate. Also, all objects' poses are shown in the frame of the robot. The object of interest, on the other hand, is represented using a mask over each depth image, showing the object to manipulate only at the pose it should

[1]Note that this simple model of $W(r)$ is generalizable to more precise workspace representations, provided that mapping the intersection between two workspaces is possible.

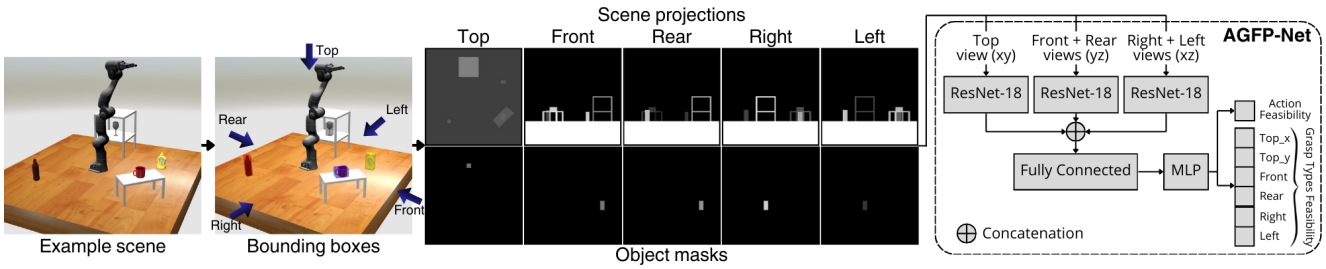


Fig. 2: Feasibility prediction pipeline for mesh-shaped objects. Scene projections and object masks are generated using the bounding boxes of objects with complex shapes, then fed to **AGFP-Net** [1] to obtain the action and grasp types feasibility.

be picked or placed at. It is important to note that these depth images are **NOT** obtained from depth cameras. Our planning approach is model-based, meaning that the shapes and poses of all objects are fully known. We use OpenCV [42] to construct orthographic projections from this known state of the 3D environment to obtain the aforementioned depth images. These are a mere scene representation method without any depth camera involved (even in practice).

AGFP-Net is trained on a fully synthetic dataset, comprised of randomly generated scenes containing a single robot, 2 box-shaped objects and up the 4 support surfaces. The dimensions and poses of these objects are sampled randomly within fixed bounds. Our results in [1] show a good generalization to new environments with a higher number of objects, obstacles and support surfaces. Also, since the input is internally generated, the sim-to-real gap is non-existent. Nonetheless, the neural network is by design limited to a single robot, and it can be used on box-shaped objects only. We refer the reader to [1] for more details.

B. Action and Grasp Feasibility

Given a state of the environment s, the tested action a and the object to manipulate O, The action a defined in (1) can be decomposed into $Pick$ and $Place$ sub-actions as follows:

$$a(r, O, s(O), \mathbf{q}) = Pick(r, O, s(O)) + Place(r, O, \mathbf{q}) \quad (2)$$

where $Pick$ corresponds to picking object O using r from $s(O)$, and $Place$ refers to placing O using robot r at $\mathbf{q}$.

For each one of the $Pick$ and $Place$ actions, we generate scene projections and object masks as explained in Section IV-A and [1]. These projections are then given as input to **AGFP-Net** to obtain the probability of feasibility of performing the action on the object of interest, as well as the feasibility of each one of the defined grasp types. Following [1], we define the probability of feasibility p_F^a of the complete action a as:

$$p_F^a = p_F^{Pick} \times p_F^{Place} \times max(\mathbf{p}_G^a) \quad (3)$$

where p_F^{Pick} and p_F^{Place} are the predicted probabilities of the $Pick$ and $Place$ sub-actions respectively, and $\mathbf{p}_G(a)$ is the vector of combined grasp type probabilities:

$$\mathbf{p}_G^a = \mathbf{p}_G^{Pick} \otimes \mathbf{p}_G^{Place} \quad (4)$$

$\otimes$ being the element-wise product.

The probability of feasibility of the action p_F^a is used at the task planning level to prioritize feasible actions during the search, while the feasibility of grasp types $\mathbf{p}_G^a$ is used during geometric planning to prioritize feasible grasps.

V. Feasibility Prediction Generalization

A. Generalization to Mesh-shaped Objects

In this work, we aim at generalizing **AGFP-Net** to objects with more complex shapes. One possibility could be to retrain the neural network on a new dataset with scenes containing mesh-shaped objects. However, since the scene projections are constructed internally, building depth images of mesh objects can be time consuming, which defeats the purpose of accelerating TAMP. Also, this method does not guarantee the generalizability to unseen shapes during training. Therefore, we propose to represent mesh-shaped objects **at the feasibility prediction level** using their bounding boxes. This allows the neural network to generalize to objects with complex shapes, without the need for any additional data generation/annotation and training effort to [1], and without an overhead in inference time.

When querying the neural network, the input scene projections are generated using the bounding boxes of mesh objects as shown in Figure 2. Just as for box-shaped objects, **AGFP-Net** outputs the probability of feasibility of performing an

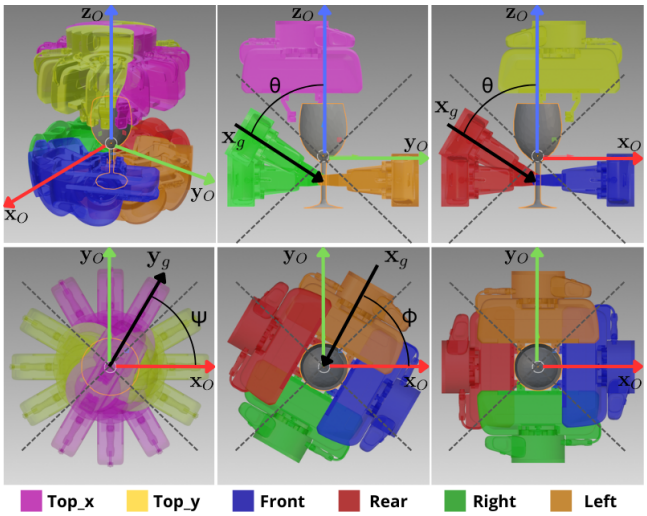


Fig. 3: Visualization of the 6 grasp types depending on the angle of approach of the robot's gripper.

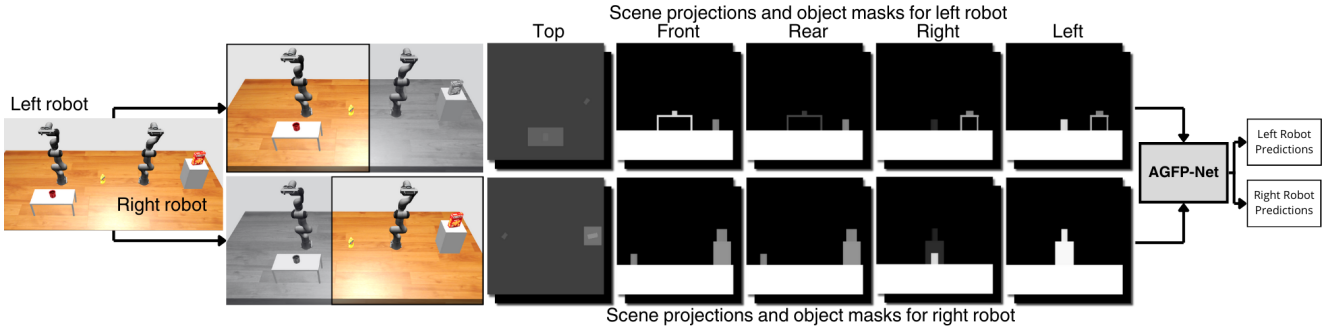


Fig. 4: Visualization of **AGFP-Net**'s input on an example 2-robot scene. To predict the feasibility of a $Pick$ or $Place$ action, the neural network is queried for each robot individually, using depth images showing the objects inside their workspaces, respectively. Objects in the intersection of both workspaces can be seen in both scene projections (i.e mustard bottle).

action on mesh-shaped objects, as well as the feasibility of each grasp type. Since these grasp types represent continuous subsets of grasps, we need to associate the mesh object's grasps (obtained using an off-the-shelf grasp planner or from a grasp database) to one of these grasp types.

Given an object O which axes are $(\mathbf{x}_O, \mathbf{y}_O, \mathbf{z}_O)$, and a grasp g of O which direction of approach is defined by the axes $(\mathbf{x}_g, \mathbf{y}_g, \mathbf{z}_g)$, we define the angles Θ, Φ, Ψ as the angles between $\mathbf{x}_g$ and $\mathbf{z}_O$, $\mathbf{x}_g$ and $\mathbf{x}_O$, $\mathbf{y}_g$ and $\mathbf{x}_O$, respectively (cf. Figure 3). The grasp type of g can be obtained as follows:

$$type(g) = \begin{cases} Front & |\Theta| > \frac{\pi}{4} \text{ and } |\Phi| > \frac{3\pi}{4} \\ Rear & |\Theta| > \frac{\pi}{4} \text{ and } |\Phi| \leq \frac{\pi}{4} \\ Right & |\Theta| > \frac{\pi}{4} \text{ and } \frac{-3\pi}{4} \leq \Phi < \frac{-\pi}{4} \\ Left & |\Theta| > \frac{\pi}{4} \text{ and } \frac{\pi}{4} < \Phi \leq \frac{3\pi}{4} \\ Top_x & |\Theta| \leq \frac{\pi}{4} \text{ and } \frac{\pi}{4} < |\Psi| \leq \frac{3\pi}{4} \\ Top_y & |\Theta| \leq \frac{\pi}{4} \text{ and } (|\Psi| \leq \frac{\pi}{4} \text{ or } |\Psi| > \frac{3\pi}{4}) \end{cases} \quad (5)$$

Using this formulation, we define the probability of feasibility of g as the probability of feasibility of its type.

B. Generalization to Multi-Robot Settings

AGFP-Net [1] is by design robot-centric, meaning that it is dependent on the kinematics of the robot only, and does not depend on any other factors such as where the robot is placed in the environment. Also, the model covers the whole workspace the robot. These properties are achieved by expressing all objects poses, fixed or movable, in the frame of the robot. The scene projections are constructed such that the center of the robot's workspace is at the center of the depth images, and each projection covers the whole workspace of the robot. As a result, the scene projections only show the objects inside the robot's workspace, hence feasibility prediction can be performed on each one of these objects. These capabilities allow a smooth generalization of **AGFP-Net** to multi-robot problems. Indeed, the neural network can be queried for each robot individually by making sure the input shows the objects (or parts of objects) in the workspace of the said robot only, and that the shown poses of these objects are in the frame of the latter, as shown in Figure 4.

The challenge, however, lies in estimating the feasibility of collaborative actions which involve two robots such as $Pass$ actions or $Handover$ actions. In this paper, we focus on $Pass$ actions (the proposed approach can nonetheless be easily extended to $Handover$ actions). A $Pass$ action aims at moving an object to the intersection region between two robots' workspaces using one robot, so that the other is able to manipulate it. Therefore, a $Pass$ action is feasible only if one robot is able to pick the object, place it at the intersection region, and if the other robot is able to pick it from this region. In order to be reliable, feasibility prediction must take into account both robots when evaluating collaborative actions. We propose a new method for estimating collaborative actions feasibility (CF).

Given an object O's pose in the world frame $\mathbf{q}_O$ and a robot r's frame transform $\mathbf{T}_r$, the pose of O expressed in the frame of the robot is simply $\mathbf{q}_O^r = \mathbf{T}_r^{-1}\mathbf{q}_O$. For a state s and a $Pass$ action $a_{pass}(r_1, O, s(O) \rightarrow \mathbf{q})$ between two robots r_1 and r_2, we transform the pick and place poses to the frame of each robot to obtain $s(O)^{r_1}$, $\mathbf{q}^{r_1}$ and $\mathbf{q}^{r_2}$ [2]. Then, we compute the probability of feasibility $p_{CF}^{a_{pass}}$ of the $Pass$ action between r_1 and r_2 as:

$$p_{CF}^{a_{pass}} = p_F^{a_{pass}}(r_1, O, s(O)^{r_1}, \mathbf{q}^{r_1}) \times p_F^{Pick}(r_2, O, \mathbf{q}^{r_2}) \quad (6)$$

For non-collaborative actions a_{goal} of type $goal$, and a_{temp} of type $Temp$ using a robot r, the probability of feasibility is computed as in (3):

$$p_F^{a_{goal}} = p_F^{a_{temp}} = p_F^a(r, O, s(O)^r, \mathbf{q}^r) \quad (7)$$

We develop a TAMP algorithm which uses these probabilities of feasibility to prioritize feasible actions and speedup the search for a geometrically feasible task plan.

VI. Task and Motion Planning

We propose a feasibility-informed multi-robot TAMP algorithm capable of solving problems involving multiple robots. The main algorithm, shown in Algorithm 1, is based on a best-first tree search, where nodes represent states and edges are actions allowing transitions between states, following the definitions of Section III. A list Q of open nodes is maintained and sorted according to a defined cost. At each iteration, the node s with the minimum cost is

[2]Note that $\mathbf{q}^{r_1}$ and $\mathbf{q}^{r_2}$ represent the placement pose expressed in the frame of robots r_1 and r_2 respectively, and are not robot configurations.

extracted from Q and compared to the goal state s_{goal}. If s is a goal state, we retrieve the complete task plan leading to s then query the geometric planner to check the feasibility of every action in the task plan and plan their corresponding motions. If the action sequence is feasible, then a solution was found. Otherwise, the search continues. During geometric planning, the probabilities of feasibility of grasp types for each action are given to the geometric planning to prioritize feasible grasps.

Algorithm 1 Task and motion planner

Input: $E, s_0, s_{goal}, \mathbf{\Gamma}$ ▷ Environment, Initial and goal states, robots graph
$Q \leftarrow \{s_0\}$ ▷ Set of nodes to expand
while Solution not found **do**
  $s \leftarrow argmin_{cost} Q$
  if $s \in s_{goal}$ **then**
    $[\tau, \Pi] \leftarrow retrieveSolution(s)]$
    if Π is feasible **then**
      return τ, Π
    end if
  else
    $Q \leftarrow Q \cup findChildren(s, E, s_{goal}, \mathbf{\Gamma})$
  end if
end while

Algorithm 2 findChildren

Input: $s, E, s_{goal}, \mathbf{\Gamma}$
$children \leftarrow \emptyset$
$A \leftarrow findPossibleActions(s, E, s_{goal}, \mathbf{\Gamma})$
for each a in A **do**
  $[p^a_F, \mathbf{p}^a_G)] \leftarrow predictFeasibility(s, E, a)$
  $child \leftarrow nextState(s, a)$
  $child.cost \leftarrow computeCost(child, s_{goal}, p_F(a), E)$
  $children \leftarrow children \cup child$
end for
return $children$

If s in not a goal state, we expand the node using the function $findChildren$, shown in Algorithm 2. It first samples applicable actions at state s, by calling the function $findPossibleActions$ which is detailed in Algorithm 3. For each movable object O in the environment, we find the set of robots capable of reaching O at its pose in s, using the function $findReachingRobots(s(O), E)$. For each robot r_i in the obtained set, we generate $Goal$, $Pass$ and $Temp$ actions. In the case where O is not already at its goal pose, if s_{goal} is reachable by r_i, we sample a set of goal placements in the workspace of r_i. Otherwise, if the goal pose of O is not reachable by r_i, we try to generate $Pass$ placements.

We first find the set of robots that are able to reach s_{goal}. The planner needs to figure out how to bring O to each one of these robot's workspace. In order to do that, we build a graph $\mathbf{\Gamma}$ which vertices are all the robots in the environment, and edges represent the existence of an intersections between two robots' workspaces. For each robot r_g reaching s_{goal}, we use a Breadth-First Graph Search to find all possible paths from r_i to r_g, and we extract the first robot from each path. These are all the robots r_i can pass object O to in order to bring it closer to its goal pose. For each robot r_j in the obtained set of robots, we sample a number of $Pass$ placements at the intersection between $W(r_i)$ and $W(r_j)$.

Finally, we sample a set of $Temp$ placements in the workspace of r_i. Then, we generate actions corresponding to each one of the $Goal$, $Pass$ and $Temp$ placements sampled[3]. Once these actions have been generated, we call $predictFeasibility$ for each action a, which queries **AGFP-Net** and uses (3), (4) and (6) to compute the probability of feasibility of the action and the feasibility of the grasp types. We construct a new child as the result of applying a at s, we compute its cost and add it to the list of open nodes. We define the cost of a node as:

$$C_{Total} = C_{SoFar} + C_{ToGoal} + C_{Feasibility} \tag{8}$$

where C_{SoFar} is the number of actions in the branch leading to the node, C_{ToGoal} is the minimum number of actions to reach the goal state, and $C_{Feasibility}$ is the feasibility cost detailed in [1]. In this work, C_{ToGoal} is constructed differently compared to [1] [33] in order to tackle multi-robot problems. Indeed, we take advantage of the previously constructed graph $\mathbf{\Gamma}$ to compute for each object the shortest path length to the goal using Breadth-First Graph Search. We then sum the obtained path lengths to obtain the minimum number of actions to reach the goal state.

Algorithm 3 findPossibleActions

Input: $s, E, s_{goal}, \mathbf{\Gamma}$
$A \leftarrow \emptyset$
for each O in movable objects **do**
  for each r_i in $findReachingRobots(s(O), E)$ **do**
    $P \leftarrow \emptyset$
    if $s(O) \notin s_{goal}(O)$ **then**
      if $s_{goal}(O)$ is reachable by r_i **then**
        $P \leftarrow P \cup sampleGoal(r_i, E, s_{goal}(O))$
      else
        for r_g in $findReachingRobots(s_{goal}(O), E)$ **do**
          $next\_robots \leftarrow BFS(\mathbf{\Gamma}, r_i, r_g)$
          for $r_j \in next\_robots$ **do**
            $P \leftarrow P \cup samplePass(r_i, r_j, E)$
          end for
        end for
      end if
    end if
    $P \leftarrow P \cup sampleTemp(r_i, E)$
    for each $\mathbf{q} \in P$ **do**
      $a \leftarrow Move(r_i, O, s(O) \rightarrow \mathbf{q})$
      $A \leftarrow A \cup a$
    end for
  end for
end for
return A

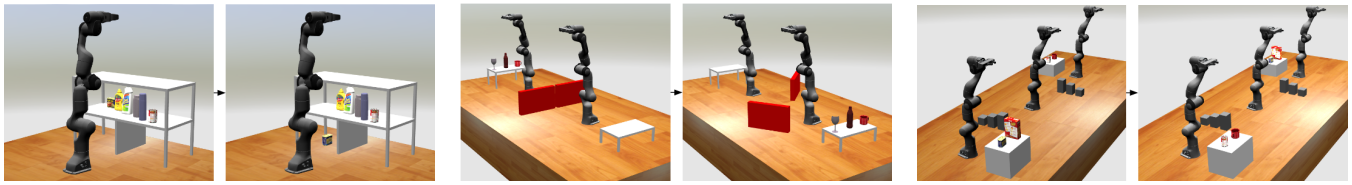
VII. Experiments

A. Test TAMP problems

In order to demonstrate the performance of our TAMP algorithm, we construct multiple TAMP problems with varying challenges in different single and multi-robot settings. Each problem contains a number of mesh-shaped objects extracted from the YCB [43] and KIT [44] objects databases. Note that none of these objects nor their bounding boxes were used during the training of **AGFP-Net**. Also note that each object is annotated with up to 200 grasps to be used by the geometric planner, that is much more than other works (eg. [22]) which generally use a single to a few grasps only.

We modify the $Access$ and $Sort$ domains defined in [1] to measure the performance of our generalization to mesh

[3]Although the number of placements sampled is a parameter fixed by the user, we give the planner the possibility to resample new placements, in case the previous ones do not lead to a feasible solution.



(a) *Access* problem (b) *Clear* problem (c) *Longswap* problem

Fig. 5: Visualization of the initial and goal states for 3 of the 6 problem domains used to test our approach.

objects and multi-robot settings. In the *Access* problem shown in Figure 5a, a single robot has to move a meat can. A number of mesh objects are however blocking access to it, which requires removing all blocking objects to access the wanted one, before returning them to their initial pose. The *Sort* problem, illustrated in Figure 1a is a dual-robot problem where the goal is to sort objects on two pre-occupied tables. Here, the algorithm has to find feasible sets of placements on narrow surfaces, although each robot can reach one of the tables and a subset of the objects only.

We also define TAMP problems in which the presence of multiple robots makes planning more challenging. In the *Clear* problem, illustrated in Figure 5b, two large objects (in red) span over the intersection region of two robots' workspaces, rendering *Pass* actions infeasible. The robots have to first clear the intersection region before moving a set of objects from one table to another. In the the *Longswap* problem (Figure 5c), three sequential robots have to collaborate to swap the placements of 4 objects. The intersection regions between robots' workspaces are, however, partially blocked by a set of obstacles. Finally, we define the $Middleman-3$ problem, shown in Figure 1b, in which three robots form a triangle such that there is an intersection between every pair of robots' workspaces. One of the intersection regions is blocked by a fixed obstacle, forcing the robots to perform two *Pass* actions via a middleman robot to move three objects from one counter to the other. In order to demonstrate the ability of our approach to handle arbitrary multi-robot settings, we define a 5-robot version of this problem named $Middleman-5$ (Figure 1c). In the latter, four symbolic action sequences allow each one of four objects to be moved from a shelf to a table, however three of them are blocked by fixed obstacles. The planner must find the only geometrically feasible sequence of robots exchanges to solve the problem.

B. Implementation details

We run our feasibility-informed TAMP algorithm on each one of these problems for 10 trials each, with a timeout set to 900 seconds per trial. We use Moveit! Task Constructor [45] for geometric planning with a BiTRRT motion planner [46]. On the feasibility prediction side, we reuse the neural network weights obtained using the training approach detailed in [1]. Experiments were conducted on an Intel i9-11950H @ 2.60GHz, with 32GB of RAM and NVIDIA RTX A3000 GPU. For comparison, we also run a baseline version of our algorithm that does not use feasibility prediction, by setting all probabilities of feasibility to 1 during the search.

VIII. Results

Table I shows the averaged results obtained for each problem. Results show that our feasibility-informed TAMP algorithm is able to solve most problems with 100% success rate, compared to the non-informed baseline which completely fails to solve 3 of the 6 defined problems. Given the high combinatorial complexity of these TAMP problems, this outcome can be expected and shows that our method is able to efficiently filter the tree search to reach a solution faster.

On the *Access* problem, our approach improves the success rate from 0% to 100%, averaging a planning time of $86.6s$. Similarly, on the dual-robot *Sort* problem, our informed planner improves the success rate from 10% to 90%, and accelerates planning time by a factor of 6.2. These results are comparable to the ones obtained in [1], which shows that our proposed feasibility prediction generalization framework is able to handle mesh-shaped objects and multi-robot systems without hurting planning time.

Results on the *Clear* problem show a 93% reduction in planning time using feasibility prediction. **AGFP-Net** is able to predict that *Pass* actions are initially infeasible due to the objects blocking the intersection region between the robots' workspaces, and allows the planner to prioritize clearing this region first. This is particularly shown by the reduction in the number of infeasible task plans generated from 21.4 to 0.4. The three-robot $Middleman-3$ problem presents a similar scenario, except the objects blocking the intersection region are fixed obstacles. Also, the added robot adds to the combinatorial complexity of the problem. Results show an improvement in success rate from 0% without feasibility prediction to 100% using **AGFP-Net**. Also, total planning time using the latter is at least 33 times faster. This shows that our feasibility-informed planner avoids spending extensive effort on trying to perform a single *Pass* action directly to the goal robot, and prefers the use of a middleman robot with two *Pass* actions for each object, leading to a reduction in the number of infeasible task plans from 146.6 to 0.4.

Results on the *Longswap* and $Middleman-5$ problems demonstrate the ability of our approach to generalize to different multi-robot settings. On the *Longswap* problem, in addition to an improvement in success rate from 40% to 100%, our feasibility-informed planner reduces the total planning time by at least 90%. Using **AGFP-Net**, our algorithm identifies which of the sampled *Pass* placements are collision-free and prioritizes them. It also recognizes occupied goal placements and includes *Temp* placements in its solution in order to free them. On the $Middleman-5$

TABLE I: Planning performances with and without using feasibility prediction, averaged over 10 trials. Speedup is computed over all the trials by considering the timeout for failed cases and the average total planning time for successful ones.

Problem	Method	Success Rate (%)	Total Planning Time (s)	Geometric Planning Time (s)		Feasibility Prediction Time (s)	Infeasible Task Plans	Speedup
				Feasible actions	Infeasible actions			
Access	Baseline	0%	$>$ 900	$>$ 458	$>$ 440.3	-	$>$ 303.3	$>$ **10.4**
	Ours	**100%**	**86.6 (+/-52.7)**	**33.9 (+/-10.3)**	**5.3 (+/-10.1)**	**44.4 (+/-49.9)**	**1.7 (+/-1.1)**	
Sort	Baseline	10%	666.1 (+/-0.0)	23.7 (+/-0.0)	639.8 (+/-0.0)	-	93.0 (+/-0.0)	$>$ **6.2**
	Ours	**90%**	**56.4 (+/-35.0)**	**27.5 (+/-11.9)**	**0.8 (+/-1.6)**	**26.5 (+/-27.2))**	**0.3 (+/-0.5)**	
Clear	Baseline	70%	299.8 (+/-117.4)	16.4 (+/-6.9)	282.8 (+/-113.7)	-	21.4 (+/-9.1)	$>$ **13.7**
	Ours	**100%**	**35.0 (+/-22.1)**	**12.2 (+/-4.5)**	**7.4 (+/-11.8)**	**15.0 (+/-9.9)**	**0.4 (+/-0.7)**	
Middleman-3	Baseline	0%	$>$ 900	$>$ 137.4	$>$ 766.7	-	$>$ 146.6	$>$ **33**
	Ours	**100%**	**27.3 (+/-8.3)**	**19.4 (+/-8.0)**	**0.2 (+/-0.4)**	**7.3 (+/-3.6)**	**0.4 (+/-0.9)**	
Longswap	Baseline	40%	438.4 (+/-144.4)	58.4 (+/-11.8)	377.9 (+/-133.9)	-	29.8 (+/-13.3)	$>$ **9.9**
	Ours	**100%**	**72.0 (+/-27.3)**	**34.1 (+/-3.1)**	**23.9 (+/-22.5)**	**12.4 (+/-5.4)**	**1.6 (+/-1.2)**	
Middleman-5	Baseline	0%	$>$ 900	$>$ 69.5	$>$ 842.1	-	$>$ 54.4	$>$ **3.4**
	Ours	90%	**198.0 (+/-173.3)**	**46.8 (+/-14.8)**	**40.2 (+/-53.1)**	**90.8 (+/-74.4)**	**1.4 (+/-2.0)**	

problem, the gain in performance is even more significant with a success rate of 90% compared to the baseline which completely fails to solve the problem. Feasibility prediction guarantees a high success rate even if the combinatorial complexity is much higher due to the increased number of robots, which is shown by an average planning time of $198s$, higher than the one obtained on the other problems, but still reasonable given the complexity of the problem.

We conduct an ablation study in order to evaluate the benefit of using the collaborative feasibility introduced in Section V-B. Table II shows a comparison of success rates, total planning time, number of expanded nodes and number of feasibility checks, with and without using collaborative feasibility (CF). Results show a clear performance improvement on all multi-robot problems when collaborative feasibility is used. Indeed, both the $Pick$ and the $Place$ actions composing a $Pass$ action can be feasible. However, the following $Pick$ action using the receiving robot might not be. Taking into account the feasibility of this second $Pick$ action allows the planner to prioritize other $Pass$ actions, or actions that aim at clearing the intersection region if necessary. This translates into less expanded nodes as well as less feasibility checks, which can be observed in Table II on all problems. Particularly, results obtained on the harder $Middleman-3$, $Longswap$ and $Middleman-5$ problems show CF not only reduces the number of expanded nodes and feasibility checks, it also improves success rate and planning time. On the $Middleman-3$ problem, in addition to a 50% improvement in success rate, planning time is 3.9 times faster. Moreover, the number of expanded nodes is 68 times lower, while the number of feasibility checks is reduced by 96%. Table II shows a 90% improvement in success rate for the $Longswap$ problem thanks to the use of CF, with 58% faster planning time, 59 times less expanded nodes and 23 times less feasibility checks. On the $Middleman-5$ problem, results show that the planner completely fails to solve the problem without collaborative feasibility, compared to a 90% success rate using CF. This demonstrates the necessity of using CF for the more complex problems.

These results show that our proposed approach is able to tackle single and multi-robot TAMP problems involving mesh-shaped objects. Feasibility prediction not only guarantees almost 100% success rate on all problems, but it also reduces considerably the planning time. Also, the overhead due to feasibility prediction is largely compensated by the time saved in geometric planning time.

TABLE II: Comparison of the planning performances obtained with (+) and without (-) collaborative feasibility (CF).

Problem	CF	Success Rate	Planning Time (s)	Expanded Nodes	Feasibility Checks
Sort	-	70%	120	444	6166
	+	**90%**	**56**	**144**	**2363**
Clear	-	90%	**31**	60	1635
	+	**100%**	35	**40**	**1025**
Middleman-3	-	50%	106	1093	14417
	+	**100%**	**27**	**16**	**598**
Longswap	-	10%	172	1771	22330
	+	**100%**	**72**	**30**	**961**
Middleman-5	-	0%	$>$ 900	$>$ 4616	$>$ 104433
	+	**90%**	**198**	**797**	**6271**

IX. Conclusion

In this paper, we present a feasibility-informed multi-robot TAMP algorithm, capable of solving complex problems involving multiple robots in 3D environments. We introduce a method for extending the use of **AGFP-Net** [1] to mesh-shaped objects, allowing action and grasp feasibility prediction on realistic objects. We also propose a framework for predicting the feasibility of actions in arbitrary multi-robot settings, taking advantage of the robot-centric nature of the neural network, and using a new approach for estimating the feasibility of collaborative actions such as $Pass$ actions. We demonstrate the performance of our method on six TAMP problems containing multiple mesh-shaped objects, and different single and multi-robot settings. Results show a notable gain in success rate and planning time using feasibility prediction as a heuristic. Future work involves extending the approach to more collaborative actions such as handovers, and to coordinated motion planning problems, allowing a parallel execution of tasks. We also plan to experimentally demonstrate our TAMP planner in real settings.

REFERENCES

[1] S. A. Bouhsain, R. Alami, and T. Simeon, "Simultaneous action and grasp feasibility prediction for task and motion planning through multi-task learning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[2] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, 2021.

[3] A. M. Wells, N. T. Dantam, A. Shrivastava, and L. E. Kavraki, "Learning feasibility for task and motion planning in tabletop environments," *IEEE robotics and automation letters*, 2019.

[4] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint, "Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*.

[5] L. Xu, T. Ren, G. Chalvatzaki, and J. Peters, "Accelerating integrated task and motion planning with neural feasibility checking," *arXiv preprint arXiv:2203.10568*, 2022.

[6] Z. Yang, C. R. Garrett, and D. Fox, "Sequence-based plan feasibility prediction for efficient task and motion planning," *arXiv preprint arXiv:2211.01576*, 2022.

[7] R. Alami, T. Simeon, and J.-P. Laumond, "A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps," in *The fifth international symposium on Robotics research*, 1990.

[8] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," in *1994 IEEE International Conference on Robotics and Automation*.

[9] J. M. Ahuactzin, K. Gupta, and E. Mazer, "Manipulation planning for redundant robots: a practical approach," *The International Journal of Robotics Research*, 1998.

[10] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, 2004.

[11] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, 2010.

[12] K. Hauser, "Randomized belief-space replanning in partially-observable continuous spaces," in *Algorithmic Foundations of Robotics IX*, 2010.

[13] 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.

[14] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation with multiple action types," in *Experimental Robotics*, 2013.

[15] J. Barry, L. P. Kaelbling, and T. Lozano-Pérez, "A hierarchical approach to manipulation with diverse actions," in *2013 IEEE International Conference on Robotics and Automation*.

[16] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, 2009.

[17] 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 *2014 IEEE international conference on robotics and automation (ICRA)*.

[18] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, and L. Karlsson, "Efficiently combining task and motion planning using geometric constraints," *The International Journal of Robotics Research*, 2014.

[19] L. De Silva, M. Gharbi, A. K. Pandey, and R. Alami, "A new approach to combined symbolic-geometric backtracking in the context of human-robot interaction," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*.

[20] M. Gharbi, R. Lallement, and R. Alami, "Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[21] J. Bidot, L. Karlsson, F. Lagriffoul, and A. Saffiotti, "Geometric backtracking for combined task and motion planning in robotic systems," *Artificial Intelligence*, 2017.

[22] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, 2018.

[23] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Sampling-based methods for factored task and motion planning," *The International Journal of Robotics Research*, 2018.

[24] R. Chitnis, D. Hadfield-Menell, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel, "Guided search for task and motion plans using learned heuristics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*.

[25] J. Carpentier, R. Budhiraja, and N. Mansard, "Learning feasibility constraints for multi-contact locomotion of legged robots," in *Robotics: Science and Systems*, 2017.

[26] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, "Learning to guide task and motion planning using score-space representation," *The International Journal of Robotics Research*, 2019.

[27] B. Kim and L. Shimanuki, "Learning value functions with relational state representations for guiding task-and-motion planning," in *Conference on Robot Learning*, 2020.

[28] D. Driess, J.-S. Ha, and M. Toussaint, "Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image," in *Robotics: Science and Systems 2020*.

[29] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*.

[30] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Planning with learned object importance in large problem instances using graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, 2021.

[31] B. Kim, L. Shimanuki, L. P. Kaelbling, and T. Lozano-Pérez, "Representation, learning, and planning algorithms for geometric task and motion planning," *The International Journal of Robotics Research*, 2022.

[32] M. J. McDonald and D. Hadfield-Menell, "Guided imitation of task and motion planning," in *Conference on Robot Learning*, 2022.

[33] S. Ait Bouhsain, R. Alami, and T. Simeon, "Learning to predict action feasibility for task and motion planning in 3d environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*.

[34] M. Khodeir, B. Agro, and F. Shkurti, "Learning to search in task and motion planning with streams," *IEEE Robotics and Automation Letters*, 2023.

[35] S. Park, H. C. Kim, J. Baek, and J. Park, "Scalable learned geometric feasibility for cooperative grasp and motion planning," *IEEE Robotics and Automation Letters*, 2022.

[36] L. Antonyshyn, J. Silveira, S. Givigi, and J. Marshall, "Multiple mobile robot task and motion planning: A survey," *ACM Computing Surveys*, 2023.

[37] I. Umay, B. Fidan, and W. Melek, "An integrated task and motion planning technique for multi-robot-systems," in *2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*.

[38] H. Karami, A. Thomas, and F. Mastrogiovanni, "Task allocation for multi-robot task and motion planning: A case for object picking in cluttered workspaces," in *International Conference of the Italian Association for Artificial Intelligence*, 2021.

[39] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Transactions on Robotics*, 2022.

[40] H. Zhang, S.-H. Chan, J. Zhong, J. Li, S. Koenig, and S. Nikolaidis, "A mip-based approach for multi-robot geometric task-and-motion planning," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*.

[41] J. Motes, T. Chen, T. Bretl, M. M. Aguirre, and N. M. Amato, "Hypergraph-based multi-robot task and motion planning," *IEEE Transactions on Robotics*, 2023.

[42] G. Bradski, "The opencv library." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 2000.

[43] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*.

[44] A. Kasper, Z. Xue, and R. Dillmann, "The kit object models database: An object model database for object recognition, localization and manipulation in service robotics," *The International Journal of Robotics Research*, 2012.

[45] M. Görner, R. Haschke, H. Ritter, and J. Zhang, "Moveit! task constructor for task-level motion planning," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*.

[46] D. Devaurs, T. Siméon, and J. Cortés, "Enhancing the transition-based rrt to deal with complex cost spaces," in *2013 IEEE International Conference on Robotics and Automation*.