HAL open science

Monte-Carlo Tree Search for Efficient Visually Guided Rearrangement Planning

Yann Labbé, Sergey Zagoruyko, Igor Kalevatykh, Ivan Laptev, Justin Carpentier, Mathieu Aubry, Josef Sivic

To cite this version:

Yann Labbé, Sergey Zagoruyko, Igor Kalevatykh, Ivan Laptev, Justin Carpentier, et al.. Monte-Carlo Tree Search for Efficient Visually Guided Rearrangement Planning. IEEE Robotics and Automation Letters, 2020, 5 (2), pp.3715-3722. 10.1109/LRA.2020.2980984 . hal-02108930

HAL Id: hal-02108930

https://hal.science/hal-02108930v1

Submitted on 11 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

arXiv:1904.10348v2 [cs.RO] 1 Apr 2020

Monte-Carlo Tree Search for Efficient Visually Guided Rearrangement Planning

Yann Labbé [a,b], Sergey Zagoruyko [a,b], Igor Kalevatykh [a,b], Ivan Laptev [a,b], Justin Carpentier [a,b], Mathieu Aubry [c] and Josef Sivic [a,b,d]

***Abstract*— We address the problem of visually guided rearrangement planning with many movable objects, i.e., finding a sequence of actions to move a set of objects from an initial arrangement to a desired one, while relying on visual inputs coming from an RGB camera. To do so, we introduce a complete pipeline relying on two key contributions. First, we introduce an efficient and scalable rearrangement planning method, based on a Monte-Carlo Tree Search exploration strategy. We demonstrate that because of its good trade-off between exploration and exploitation our method (i) scales well with the number of objects while (ii) finding solutions which require a smaller number of moves compared to the other state-of-the-art approaches. Note that on the contrary to many approaches, we do not require any buffer space to be available. Second, to precisely localize movable objects in the scene, we develop an integrated approach for robust multi-object workspace state estimation from a single uncalibrated RGB camera using a deep neural network trained only with synthetic data. We validate our multi-object visually guided manipulation pipeline with several experiments on a real UR-5 robotic arm by solving various rearrangement planning instances, requiring only 60 ms to compute the plan to rearrange 25 objects. In addition, we show that our system is insensitive to camera movements and can successfully recover from external perturbations. Supplementary video, source code and pre-trained models are available at https://ylabbe.github.io/rearrangement-planning.**

I. Introduction

Using a robot to clean up a room is a dream shared far beyond the robotics community. This would require a robot to both localize and re-arrange many objects. Other industrial scenarios, such as sorting and packing objects on a production line or car assembly tasks, share similar objectives and properties. This paper presents an integrated approach that makes a step towards the efficiency, scalability and robustness required for solving such rearrangement planning tasks. Fig. 1 shows an example of the problem we consider, where objects have to be moved from an initial position to a target one. The current and target states are described only by a single image taken from an uncalibrated RGB camera.

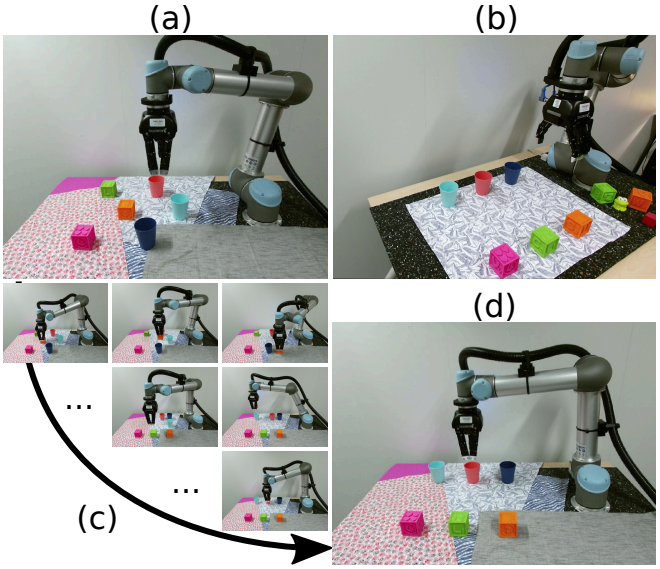

Fig. 1: **Visually guided rearrangement planning.** Given a source (a) and target (b) RGB images depicting a robot and multiple movable objects, our approach estimates the positions of objects in the scene without the need for explicit camera calibration and efficiently finds a sequence of robot actions (c) to re-arrange the scene into the target scene. Final object configuration after re-arrangement by the robot is shown in (d).

Rearrangement planning has a long history in robotics [1]–[6] and remains an active research topic [7]–[10] in the motion planing community. The goal is to find a sequence of *transit* and *transfer* motions [1,5] to move a set of objects from an initial arrangement to a target arrangement, while avoiding collisions with the environment. This leads to a complex sequential decision process, whose complexity depends on the number of objects to move, on the free-space available around the objects, and the robot kinematics.

Several solutions have been proposed in the literature which can be roughly classified into two groups. Methods in the first group [6,8]–[12] rely on the *task* and *motion* planning hierarchy where a high-level task planner is combined with a local motion planner [3]. Methods in the second group [2,4,13,14] aim at solving a single unified formulation of the problem by using classic sample-based algorithms such as Probabilistic RoadMap (PRM) or Rapidly-Exploring Random Tree (RRT) [3] or use advanced optimization strategies to solve a unique optimization instance [15].

While methods from both groups have been shown to work well in practice with few objects, existing methods do

This work was partially supported by the DGA RAPID projects DRAAF and TABASCO, the MSR-Inria joint lab, the Louis Vuitton - ENS Chair on Artificial Intelligence, the HPC resources from GENCI-IDRIS (Grant 011011181), the ERC grant LEAP (No. 336845), the CIFAR Learning in Machines&Brains program, the European Regional Development Fund under the project IMPACT (reg. no. CZ.02.1.01/0.0/0.0/15 003/0000468) and the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

[a] Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France, [b] INRIA, France,[c] LIGM (UMR 8049), École des Ponts, UPE, France,[d] Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague.

Corresponding author: yann.labbe@inria.fr

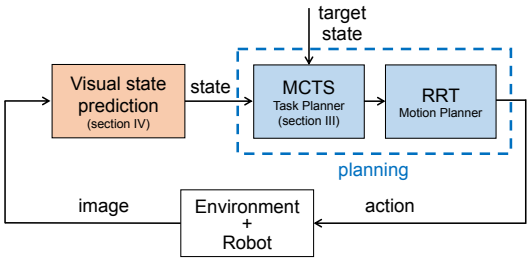

Fig. 2: **Approach overview.** Given an image of the scene, the visual state prediction module outputs a list of objects and their coordinates in the robot coordinate frame. Together with a target state, these serve as inputs to the task and motion planning module which combines Monte-Carlo Tree Search with a standard robot motion planning algorithm.

not scale to a large set of objects, because the number of possible action sequences increases exponentially with the number of objects to move. Some recent methods [8,10,13] scale better with the number of objects but these methods either only focus on feasibility, producing solutions with sub-optimal number of grasps [8], or are limited to specific constrained scenarios, for example, with explicitly available buffer space [10] or strict constraints of monotony (i.e. an object can be moved only once during the plan).

In this work we describe an efficient and generic approach for rearrangement planning that overcomes these limitations: (i) it scales well with the number of objects, by taking only 60 ms to plan complex rearrangement scenarios for multiple objects, and (ii) it can be applied to the most challenging table-top re-arrangement scenarios, not requiring explicit buffer space. Our approach is based on Monte-Carlo Tree Search [16], which allows us to lower the combinatorial complexity of the decision process and to find an optimistic number of steps to solve the problem, by making a compromise between exploration (using random sampling) and exploitation (biasing the search towards the promising already sampled action sequences to cut off some of the search directions).

To demonstrate the benefits of our planning approach in real scenarios we also introduced a multi-object calibration-free deep neural network architecture for object position estimation. It is trained entirely from synthetic images and, compared to other currently available methods [17]–[20], does not use markers or require known CAD models of the specific observed object instances. To the best of our knowledge, the approach we present is the first one able to locate multiple objects in such difficult and generic conditions. This is of high practical interest, since it allows to perform the full task using only a single non-calibrated, or even hand-held, RGB camera looking at the robot.

Our complete pipeline for visually guided rearrangement planning, illustrated in Fig. 2, is composed of three main stages. The goal of the first stage, visual state prediction (section III), is to estimate the positions of multiple objects relative to a robot given a single non-calibrated image of the scene. The second stage (Sec. IV), is our MCTS-based task planner: at each planning step, this module chooses which object to move and computes its new desired position in the workspace. The last stage is a standard RRT-based local motion planner which plans robot movements given the high-level plan computed by the MCTS planner.

II. Related Work

We build our framework on results in robotics, search algorithms and computer vision, which we review below.

Rearrangement planning is NP-hard [21]. As a result, standard hierarchical [6,8,9,12] and randomized methods [2,4,14] for solving general manipulation planning problems do not scale well with the number of objects. The most efficient and scalable high-level planners only address specific constrained set-ups leveraging the structure of the rearrangement problem [8,10,13]. In addition, they often focus on feasibility but do not attempt to find high-quality solutions with a low number of object moves [8,13]. For instance, some methods [13] only consider the monotone problem instances, where each object can be grasped at most once. In contrast, our method finds high-quality plans but also addresses the more general cases of non-monotone re-arrangement problems, which are known to be significantly harder [4]. Others works have looked at finding optimal plans [10] but address only constrained set-ups that have available buffer space (i.e. space that does not overlap with the union of the initial and target configurations), noting that solving the general case without available buffer space is significantly harder [10]. In this work, we address this more general case and describe an approach that efficiently finds high-quality re-arrangement solutions without requiring any available buffer space. In addition and unlike previous works [8,10,13], we also propose a complete system able to operate from real images in closed loop.

Search algorithms. The problem of rearrangement planning can be posed as a tree search. Blind tree search algorithms such as Breadth-First search (BFS) [22] can be used to iteratively expand nodes of a tree until a goal node is found, but these methods do not exploit information about the problem (e.g. a cost or reward) to select which nodes to expand first, and typically scale exponentially with the tree depth. Algorithms such as greedy BFS [22] allow to exploit a reward function to drive the exploration of the tree directly towards nodes with high reward, but might get stuck in a local mimima. Other algorithms such as $A^\star$ [23] can better estimate the promising branches using additional hand-crafted heuristic evaluation function. We choose Monte-Carlo Tree Search over others, because it only relies on a reward and iteratively learns a heuristic (the value function) which allows to efficiently balance between exploration and exploitation. It has been used in related areas to solve planning and routing for ground transportation [24] and to guide the tree-search in cooperative manipulation [25]. MCTS is also at the core of AlphaGo, the first system able to achieve human performance in the game of Go [26], where it was combined with neural networks to speed-up the search. These works directly address problems whose action space is discrete by nature. In contrast, the space of possible object arrangements is infinite. We propose a novel discrete action

parameterization of the rearrangement planning problem which allows us to efficiently apply MCTS.

Vision-based object localization. In robotics, fiducial markers are commonly used for detecting the objects and predicting their pose relative to the camera [17] but their use limits the type of environments the robot can operate in. This constraint can be removed by using a trainable object detector architecture [18]–[20,27]–[29]. However, these methods often require gathering training data for the target objects at hand, which is often time consuming and requires the knowledge of the object (e.g. in the form its 3D model) beforehand. In addition, these methods estimate the pose of the objects in the frame of the camera and using these predictions for robotic manipulation requires calibration of the camera system with respect to the robot. The calibration procedure [30,31] is time-consuming and must be redone each time the camera is moved. More recently, [32] proposed to directly predict the position of a single object in the robot coordinate frame by training a deep network on hundreds of thousands of synthetically generated images using domain randomization [32]–[35]. We build on the work [32] and extend it for predicting the 2D positions of *multiple objects with unknown dimensions* relative to the robot.

III. Visual scene state prediction with multiple objects

In this section, we detail the visual state prediction stage. Our visual system takes as input a single photograph of a scene taken from an uncalibrated camera and predicts a workspace state that can then be used for rearrangement-planning. More precisely it outputs the 2D positions of a variable number of objects expressed in the coordinate system of the robot. This problem is difficult because the scene can contain a variable number of objects, placed on different backgrounds, in variable illumination conditions, and observed from different viewpoints, as illustrated in Fig. 1. In contrast to [33], we do not assume that the different types of objects are known at training time. In contrast to state-of-the-art pose estimation techniques in RGB images [17,18], we do not use markers and do not assume the CAD models of the objects are known at test time.

To address these challenges, we design a visual recognition system that does not require explicit camera calibration and outputs accurate 2D positions. Moreover, even if we deploy our system on a real robot, we show that it can be trained entirely from synthetic data using domain randomization [33], avoiding the need for real training images. Also, our system does not require any tedious camera calibration because it is trained to predict positions of objects directly in the coordinate frame of the robot, effectively using the robot itself, which is visible in the image, as an (known) implicit calibration object. This feature is important for applications in unstructured environments such as construction sites containing multiple unknown objects and moving cameras for instance. Our recognition system is summarized in Fig. 3 and in the rest of this section, we present in more details the different components.

A. Position prediction network

In this section, we give details of the network for predicting a dense 2D position field and an object segmentation mask. The 2D position field maps each input pixel to a 2D coordinate frame of the robot acting as implicit calibration.

Architecture. Our architecture is based on ResNet-34 [36]. We remove the average pooling and fully connected layers and replace them by two independent decoders. Both decoders use the same architecture: four transposed convolution layers with batch normalization and leaky ReLU activations in all but the last layer. The resolution of the input image is 320×240 and the spatial resolution of the output of each head is 85×69. We add the 6D joint configuration vector of the robot as input to the network by copying it into a tensor of size $320 \times 240 \times 6$, and simply concatenating it with the three channels of the input image. The two heads predict an object mask and a 2D position field which are visualized in Fig. 3. In addition, we found that predicting depth and semantic segmentation during training increased the localization accuracy at test time. These modalities are predicted using two additionnal decoders with the same architecture.

Synthetic training data. Following [32]–[35], we use domain randomization for training our network without requiring any real data. We generate two million images displaying the robot and a variable number of objects with various shapes (cubes, cylinders and triangles) in its workspace. In each scene, we randomly sample from 1 up to 12 objects, with various dimensions between 2.5 and 8 cm. Examples of training images are shown in Fig. 4. Randomization parameters include the textures of the robot and objects, the position of the gripper, the position, orientation and field of view of the camera, the positions and intensities of the light sources and their diffuse/ambient/specular coefficients.

Training procedure. We train our network by minimizing the following loss: $\mathcal{L} = \mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{segm}} + \mathcal{L}_{\text{depth}}$, where the individual terms are explained next. For the position field loss we use $\mathcal{L}_{\text{pos}} = \sum_{i,j} \delta_{i,j} \left[(\hat{x}_{i,j} - x_{i,j})^2 + (\hat{y}_{i,j} - y_{i,j})^2 \right]$ where (i, j) are the pixel coordinates in the output; $\delta_{i,j}$ is the binary object mask; $x_{i,j}$, $y_{i,j}$ are the ground truth 2D coordinates of the center of the object (that appears at pixel (i, j)) and $\hat{x}_{i,j}, \hat{y}_{i,j}$ are the components of the predicted position field. For $\mathcal{L}_{\text{mask}}$, $\mathcal{L}_{\text{segm}}$ and $\mathcal{L}_{\text{depth}}$ we respectively use the following standard losses: binary cross entropy loss, cross entropy loss and mean squared error. These losses are computed pixel-wise. Note that depth is not used to estimate object positions at test time. L_{segm} and L_{depth} are auxiliary losses used only for training, similar to [35]. We use the Adam optimizer [37] and train the network for 20 epochs, starting with a learning rate of 10^{-3} and decreasing it to 10^{-4} after 10 epochs.

B. Identifying individual objects

The model described above predicts a dense 2D position field and an object mask but does not distinguish individual objects in the scene. Hence, we use the following procedure to group pixels belonging to each individual object. Applying

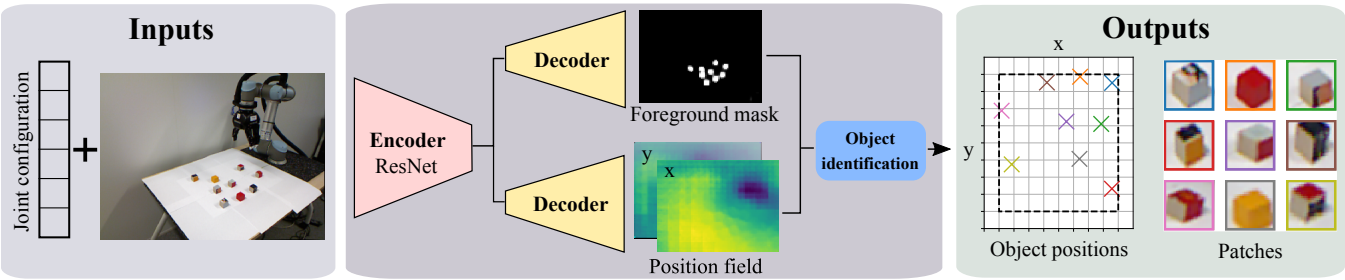

Fig. 3: **The visual recognition system.** The input is an image of the scene captured by an uncalibrated camera together with the joint configuration vector of the depicted robot. Given this input a convolutional neural network (CNN) predicts the foreground-background object segmentation mask and a dense position field that maps each pixel of the (downsampled) input image to a 2D coordinate in a frame centered on the robot. The estimated masks and position fields are then combined by the object identification module to identify individual object instances. The output is a set of image patches associated with the 2D position of the object in the workspace.

Fig. 4: Examples of synthetic training images. We generate images displaying the robot and a variable number of objects in its workspace. The images are taken by cameras with different viewpoints and depicting large scene appearance variations.

a threshold to the predicted mask yields a binary object segmentation. The corresponding pixels of the 2D position field provide a point set in the robot coordinate frame. We use the mean-shift algorithm [38] to cluster the 2D points corresponding to the different objects and obtain an estimate of the position of each object. The resulting clusters then identify pixels belonging to each individual object providing instance segmentation of the input image. We use the resulting instance segmentation to extract patches that describe the appearance of each object in the scene.

C. Source-Target matching

To perform rearrangement, we need to associate each object in the current image to an object in the target configuration. To do so, we use the extracted image patches. We designed a simple procedure to obtain matches robust to the exact position of the object within the patches, their background and some amount of viewpoint variations. We rescale patches to 64×64 pixels and extract conv3 features of an AlexNet network trained for ImageNet classification. We finally run the Hungarian algorithm to find the one-to-one matching between source and target patches maximizing the sum of cosine similarities between the extracted features. We have tried using features from different layers, or from the more recent network ResNet. We found the conv3 features of AlexNet to be the most suited for the task, based on a qualitative evaluation of matches in images coming from the dataset presented in Sec. V-B . Note that our patch matching strategy assumes all objects are visible in the source and target images.

IV. Rearrangement Planning with Monte-Carlo Tree Search (MCTS)

Given the current and target arrangements, the MCTS task planner has to compute a sequence of pick-and-place actions that transform the current arrangement into the target one. In this section, we first review Monte-Carlo Tree Search and then explain how we adapt it for rearrangement planning.

A. Review of Monte-Carlo Tree Search

The MCTS decision process is represented by a tree, where each node is associated to a state s, and each edge represents a discrete action $a = \{1, ..., N\}$. For each node in the tree, a reward function $r(s)$ gives a scalar representing the level of accomplishment of the task to solve. Each node stores the number of times it has been visited $n(s)$ and a cumulative reward $w(s)$. The goal of MCTS is to find the most optimistic path, i.e. the path that maximizes the expected reward, starting from the root node and leading to a leaf node solving the task. MCTS is an iterative algorithm where each iteration is composed of three stages.

During the *selection* stage, an action is selected using the Upper Confidence Bound (UCB) formula:

$$U(s,a) = Q(s,a) + c\sqrt{\frac{2\log n(s)}{n(f(s,a))}}, \tag{1}$$

where $f(s,a)$ is the child node of s corresponding to the edge (s,a) and $Q(s,a) = \frac{w(f(s,a))}{n(f(s,a))}$ is the expected value at state s when choosing action a. The parameter c controls the trade-off between exploiting states with high expected value (first term in (1)) and exploring states with low visit count (second term in (1)). We found this trade-off is crucial for finding good solutions in a limited number of iterations as we show in Sec. V-A. The optimistic action selected a_{selected} is the action that maximizes $U(s,a)$ given by (1). Starting from the root node, this stage is repeated until an expandable node (a node that has unvisited children) is visited. Then, a random unvisited child node s' is added to the tree in the *expansion stage*. The reward signal $r(s')$ is then back-propagated towards the root node in the *back-propagation stage*, where the cumulative rewards w and visit counts n of all the parent nodes are updated. The search process is run

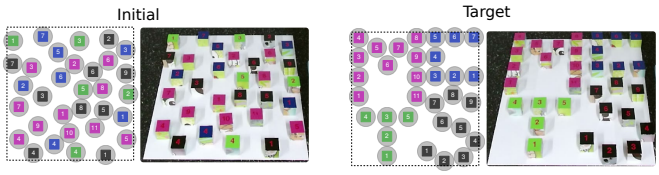

Fig. 5: Examples of source and target object configurations. The planner has to find a sequence of actions (which object to move and where to displace it inside the workspace), while avoiding collisions with other objects. Workspace limits are shown as dashed lines. Grey circles depict the collision radius for each object. We demonstrate our method solving this complex rearrangement problem in the **supplementary video.** Here source and target states are known (state estimation network is not used).

iteratively until the problem is solved or a maximal number of iterations is reached.

B. Monte-Carlo Tree Search task planner

We limit the scope to tabletop rearrangement planning problems with overhand grasps but our solution may be applied to other contexts. We assume that the complete state of any object is given by its 2D position in the workspace, and this information is sufficient to grasp it. The movements are constrained by the limited workspace, and actions should not result in collisions between objects. The planner has to compute a sequence of actions which transform the source arrangement into the target arrangement while satisfying these constraints. Each object can be moved multiple times and we do not assume that explicit buffer space is available. An example of source and target configurations is depicted in Fig. 5. We now detail the action parametrization and the reward.

Let $\{C_i\}_{i=1,..,N}$ denote the list of 2D positions that define the current arrangement with N objects, $\{I_i\}_i$ and $\{T_i\}_i$ the initial and target arrangements respectively, which are fixed for a given rearrangement problem. MCTS state corresponds to an arrangement $s = \{C_i\}_i$.

As a reward $r(s)$ we use the number of objects located within a small distance of their target position:

$$r(s) = \sum_{i=1}^{N} R_i \text{ with } R_i = \begin{cases} 1 & \text{if } ||C_i - T_i||_2 \leq \epsilon \\ 0 & \text{otherwise.} \end{cases}, \quad (2)$$

where N is the number of objects, C_i is the current location of object i, T_i is the target location of object i and ϵ is a small constant.

We define a discrete action space with N actions where each action corresponds to one pick-and-place motion moving one of the N objects. The action is hence pararametrized by 2D picking and placing positions defined by the function GET_MOTION outlined in detail in Algo. 1. The input to that function is the current state $\{C_i\}_i$, target state $\{T_i\}_i$ and the chosen object k that should be moved. The function proceeds as follows (please see also Algo. 1). First, if possible, the object k is moved directly to it's target T_k (lines 3-4 in Algo. 1), otherwise the obstructing object j which is the closest to T_k (line 7 in Algo. 1) is moved to a position inside the workspace that does not overlap with T_k (lines 8-10 in Algo. 1). The position P where to place j is found using random sampling (line 8). For collision checks, we consider simple object collision models with fixed radiuses as depicted in Figure 5. If no suitable position is found, no objects are moved (line 11). Note that additional heuristics could be added to the action parametrization to further improve the quality of the resulting solutions and to reduce the number of iterations required. Examples include (i) avoiding to place j at target positions of other objects and (ii) restricting the search for position P in a neighborhood of C_j. The parameters of the pick-and-place motion for a given state and MCTS action are computed only once during the expansion stage and then cached in the tree and recovered once a solution is found.

Algorithm 1: Action Parametrization

1 **function** GET_MOTION ($\{C_i\}_i$, $\{T_i\}_i$, k) **:**
2 /* Check if object k can be moved to T_k */
3 **if** IS_MOVE_VALID ($\{C_i\}_{i \neq k}$, C_k, T_k) **then**
4 **return** (C_k, T_k)
5 **else**
6 /* Move obstructing object j to position P*/
7 j = FIND_CLOSEST_OBJECT ($\{C_i\}_i$, T_k)
8 found, P = FIND_POSITION ($\{C_i\}_{i \neq j} \cup \{T_k\}$)
9 **if** *found* **then**
10 **return** (C_j, P)
11 **return** (C_k, C_k)

As opposed to games where a complete game must be played before having access to the outcome (win or lose), the reward in our problem is defined in every state. Therefore, we found that using a MCTS simulation stage is not necessary.

The number of MCTS iterations to build the tree is typically a hyper-parameter. In order to have a generic method that works regardless of the number of objects, we adopt the following strategy: we run MCTS until a maximum (large) number of iterations is reached or until a solution is found. We indeed noticed that the first solution provided by MCTS is already sufficiently good compared to the next ones when letting the algorithm run for longer.

The presented approach only considers tabletop rearrangement planning with overhand grasps. The main limitation is that we assume the motion planning algorithm can successfully plan all the pick-and-place motions computed in Algo. 1. This assumption does not hold for more complex environment where some objects are not reachable at any time (e.g. moving objects in a constrained space such as inside a refrigerator). In this case, the function IS_MOVE_VALID can be adapted to check whether the movement can be executed on the robot. Note that we consider simple collision models in FIND_POSITION but meshes of the objects and environment could be used if available.

V. Experiments

We start by evaluating planning (section V-A) and visual scene state estimation (section V-B) separately, demonstrating that: (i) our MCTS task planner scales well with the

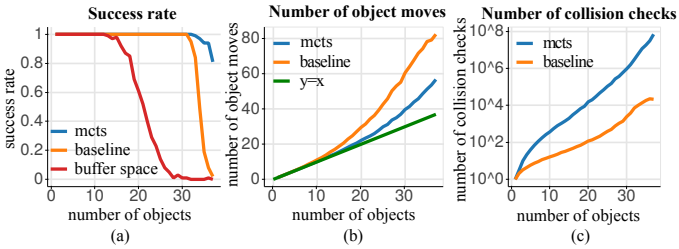

Fig. 6: Comparison of the proposed MCTS planning approach against the strong baseline heuristic. MCTS is able to solve more complex scenarios (with more objects) in a significantly lower number of steps. MCTS does not require free space to be available or that the problems are monotone.

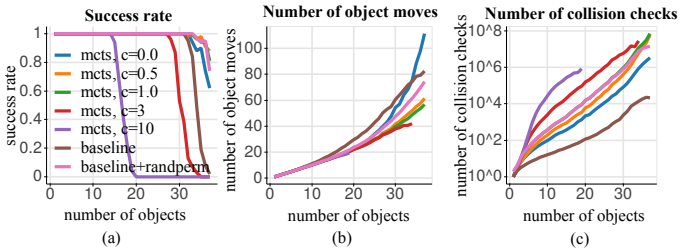

Fig. 7: Exploration-exploitation tradeoff in MCTS. MCTS performs better than a random baseline heuristic search. Balancing the exploration term of UCB with the parameter c is crucial for finding good solutions while limiting the number of collision checks.

number of objects and is efficient enough so that it can be used online (i.e. able to recompute a plan after each movement); (ii) the visual system detects and localizes the objects with an accuracy sufficient for grasping. Finally, in section V-C we evaluate our full pipeline in challenging setups and demonstrate that it can efficiently perform the task and can recover from errors and perturbations as also shown in the supplementary video.

A. MCTS planning

Experimental setup. To evaluate planning capabilities and efficiency we randomly sample 3700 initial and target configurations for 1 up to 37 objects in the workspace. It is difficult to go beyond 37 objects as it becomes hard to find valid configurations for more due to the workspace constraints.

Planning performance. We first want to demonstrate the interest of MCTS-based exploration compared to a simpler solution in term of efficiency and performances. As a baseline, we propose a fast heuristic search method, which simply iterates over all actions once in a random order, trying to complete the rearrangement using the same action space as our MCTS approach, until completion or a time limit is reached. Instead of moving only the closest object that is obstructing a target position T_k, we move all the objects that overlap with T_k to their target positions or to a free position inside the workspace that do not overlap with T_k. Our MCTS approach is compared with this strong baseline heuristic in Fig. 6. Unless specified otherwise, we use $c = 1$ for MCTS and set the maximum number of MCTS iterations to 100000.

As shown in Fig. 6(a), the baseline is able to solve complex instances but its success rate starts dropping after 33 objects whereas MCTS is able to find plans for 37 objects with 80% success. More importantly, as shown in Fig. 6(b), the number of object movements in the plans found by MCTS is significantly lower. For example, rearranging 30 objects takes only 40 object moves with MCTS compared to 60 with the baseline. This difference corresponds to more than 4 minutes of real operation in our robotics setup. The baseline and MCTS have the same action parametrization but MCTS produces higher quality plans because it is able to take into consideration the future effects of picking-up an object and placing it at a specific location. On a laptop with a single CPU core, MCTS finds plans for 25 objects in 60 ms. This high efficiency allows to replan after each pick-and-place action and perform rearrangement in a closed loop.

Exploration-exploitation trade-off. We now want to demonstrate that the benefits of our planning method are due to the quality of the exploration/exploitation trade-off in MCTS. An important metric is the total number of collision checks that the method requires for finding a plan. The collision check (checking whether an object can be placed to a certain location) is indeed one of the most costly operation when planning. Fig. 6(c) shows that MCTS uses more collision checks compared to the baseline because MCTS explores many possible optimistic action sequences while the baseline is only able to find a solution and does not optimize any objective. We propose another method that we refer to as *baseline+randperm* which solves the problem with the baseline augmented with a random search over action sequences: the baseline is run with different random object orders until a number of collision checks similar to MCTS with $c = 1$ is reached and we keep the solution which has the smallest number of object moves. As can be see in Fig. 7, baseline+randperm has a higher success rate and produces higher quality plans with lower number of object moves compared to the baseline (Fig. 7(b)). However, MCTS with $c = 1$, still produces higher quality plans given the same amount of collision checks. The reason is that baseline+randperm only relies on a random exploration of action sequences while MCTS allows to balance the exploration of new actions with the exploitation of promising already sampled partial sequences through the exploration term of UCB (equation 1). In Fig. 7, we also study the impact of the exploration parameter c. MCTS with no exploration ($c = 0$) finds plans using fewer collision checks compared to $c > 0$ but the plans have high number of object moves. Increasing c leads to higher quality plans while also increasing the number of collision checks. Setting c too high also decreases the success rate (c=3, c=10 in Fig 7(a)) because too many nodes are added to the tree and the limit on the number of MCTS iterations is reached before finding a solution.

Generality of our set-up. Previous work for finding high-quality solutions to rearrangement problems has been limited to either monotone instances [13] or instances where buffer space is available [10]. The red curve in Fig. 6(a) clearly shows that in our set-up the number of problems where some buffer space is available for at least one object quickly decreases with the number of objects in the workspace. In other words, the red curve is an upper bound on the success

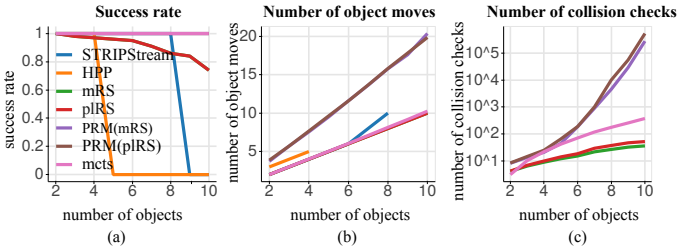

Fig. 8: Comparison of our MCTS planning approach with several other state-of-the-art methods. MCTS performs better than other methods applied to the rearrangement planning problem. MCTS finds high quality plans (b) using few collisions checks (c) with 100% success rate for up to 10 objects.

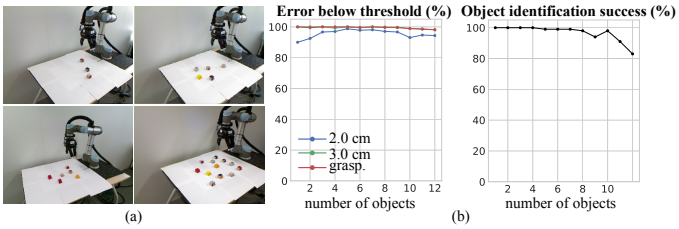

Fig. 9: (a) Example images from the dataset that we use to evaluate the accuracy of object position estimation as a function of number of objects. (b) Evaluation of our visual system for a variable number of objects. We report the localization accuracy (left) and the percentage of images where all objects are correctly identified (right). Please see the supplementary video for examples where our system is used in more challenging visual conditions.

rate of [10], which requires available buffer space. In order to evaluate the performance of our approach on monotone problems, we generated the same number of problems but the target configuration was designed by moving object from the initial configuration one by one, in a random order into free space. This ensures that the instances are monotone and can be solved by moving each object once. Our MCTS-based approach was able to solve 100% of these instances optimally in N steps. Our method can therefore solve the problems considered in [13] while also being able to handle significantly more challenging non-monotonic instances, where objects need to be moved multiple times.

Comparisons with other methods. To demonstrate that other planning methods do not scale well when used in a challenging scenario similar to ours, we compared our planner with three other methods of the state of the art: STRIPStream [12], the Humanoid Path-Planner [14], mRS and plRS [8]. Results are presented in Fig. 8 for 900 random rearrangement instances, we limit the experiments to problems with up to 10 objects as evaluating these methods for more complex problems is difficult given a reasonable amount of time (few hours). HPP [14] is the slowest method and could not handle more than 4 objects, taking more than 45 minutes of computation for solving the task with 4 objects. HPP fails to scale because it attempts to solve the combined task and motion planning problem at once using RRT without explicit task/motion planning hierarchy thus computing many unnecessary robot/environment collision checks. The other methods adopt a task/motion planning hierarchy and we compare results for the task planners only. The state-of-the-art task and motion planner for general problems, STRIPSStream [12], is able to solve problems with up to 8 objects in few seconds but do not scale (Fig. 8(a)) when target locations are specified for all objects in the scene as it is the case for rearrangement planning problems. The success rate of specialized rearrangement methods, mRS and plRS, drops when increasing number of objects because these methods cannot handle situations where objects are permuted, i.e. placing an object at its target position requires moving another objects first thus requiring longer term planning capability. When used in combination with a PRM, more problems can be addressed but the main drawback is that these methods are slow as they perform a high number of collision checks (Fig. 8(c)). Overall, the main limitation of STRIPSTream, PRM(mRS) and PRM(plRS) comes from the fact that the graph of possible states is sampled randomly whereas MCTS will prioritize the most optimistic branches (exploitation). MCTS and it's tree structure also allows to build the action sequence progressively (moving one object at once) compared to PRM-based approaches that sample entire arrangements and then try to solve them.

B. Visual scene state estimation

Experimental setup. To evaluate our approach, we created a dataset of real images with known object configurations. We used 1 to 12 small 3.5 cm cubes, 50 configurations for each number of cubes, and captured images using two cameras for each configuration, leading to a total of 1200 images depicting 600 different configurations. Example images from this evaluation set are shown in Fig. 9(a).

Single object accuracy. When a single object is present in the image, the mean-shift algorithm always succeeds and the precision of our object position prediction is 1.1 ± 0.6 cm. This is comparable to the results reported in [33] for positioning of a known object with respect to a known table without occlusions and distractors, 1.3 ± 0.6 cm, and to results reported in [32] for the coarse alignment of a single object with respect to a robot, 1.7 ± 3.4 cm. The strength of our method, however, is that this accuracy remains constant for up to 10 previously unknown objects, a situation that neither [33] nor [32] can deal with.

Accuracy for multiple objects. In Fig. 9(b), we report the performance of the object localization and object identification modules as a function of the number of objects. For localization, we report the percentage of objects localized with errors below 2 cm and 3 cm respectively. For 10 objects, the accuracy is 1.1 ± 0.6 cm. The 3 cm accuracy approximately corresponds to the success of grasping, that we evaluate using a simple geometric model of the gripper. Note that grasping success rates are close to 100% for up to 10 objects. As the number of objects increases, the object identification accuracy decreases slightly because the objects start to occlude one each other in the image. This situation is very challenging when the objects are unknown because two objects that are too close can be perceived as a single larger object. Note that we are not aware of another method

that would be able to directly predict the workspace position of multiple unseen objects with unknown dimensions using only (uncalibrated) RGB images as input.

Discussion. Our experiments demonstrate that our visual predictor is able to scale well up to 10 objects with a constant precision that is sufficient for grasping. We have also observed that our method is able to generalize to objects with shapes not seen during training, such as cups or plastic toys. While we apply our visual predictor to visually guided rearrangement planning, it could be easily extended to other contexts using additional task-specific synthetic training data. Accuracy could be further improved using a refinement similar to [32]. Our approach is limited to 2D predictions for table-top rearrangement planning. Predicting 6DoF pose of unseen objects precise enough for robotic manipulation remains an open problem.

C. Real robot experiments using full pipeline

We evaluated our full pipeline, performing both online visual scene estimation and rearrangement planning by performing 20 rearrangement tasks, each of them composed with 10 objects. In each case, the target configuration was described by an image of a configuration captured from a different viewpoint, with a different table texture and a different type of camera. Despite the very challenging nature of the task, our system succeeded in correctly solving $17/20$ of the experiments. In case of success, our system used on average 12.2 steps. The three failures were due to errors in the visual recognition system (incorrectly estimated number of objects, mismatch of source and target objects). Interestingly, the successful cases were not always perfect runs, in the sense that the re-arrangement strategy was not optimal or that the visual estimation confused two objects at one step of the matching process. However, our system was able to recover robustly from these failures because it is applied in a closed-loop fashion, where then plan is recomputed at each object move.

The supplementary video shows additional experiments including objects other than cubes, different backgrounds, a moving hand-held camera and external perturbations, where an object is moved during the rearrangement. These results demonstrate the robustness of our system. To the best of our knowledge, rearranging a priory unknown number of unseen objects with a robotic arm while relying only on images captured by a moving hand-held camera and dealing with object perturbations has not been demonstrated in prior work.

VI. Conclusion

We have introduced a robust and efficient system for online rearrangement planning, that scales to many objects and recovers from perturbations, without requiring calibrated camera or fiducial markers on objects. To our best knowledge, such a system has not been shown in previous work. At the core of our approach is the idea of applying MCTS to rearrangement planning, which leads to better plans, significant speed-ups and ability to address more general set-ups compared to prior work. While in this work we focus on table-top re-arrangement, the proposed MCTS approach is general and opens-up the possibility for efficient re-arrangement planning in 3D or non-prehensile set-ups.

References

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

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

[3] S. M. LaValle, *Planning algorithms*, 2006.

[4] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *ICRA*, 2007.

[5] J.-C. Latombe, *Robot motion planning*, 2012.

[6] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now." 2011.

[7] G. Havur *et al.*, "Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach," in *ICRA*, 2014.

[8] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement." in *RSS*, 2015.

[9] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach." in *Robotics: Science and systems*, vol. 12. Ann Arbor, MI, USA, 2016, p. 00052.

[10] H. Shuai, N. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "High-quality tabletop rearrangement with overhand grasps: Hardness results and fast methods," in *Robotics: Science and Systems (RSS)*, Cambridge, MA, 07/2017 2017.

[11] 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 *ICRA*, 2014.

[12] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," 2020.

[13] A. Krontiris and K. E. Bekris, "Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner," in *ICRA*, 2016.

[14] J. Mirabel, S. Tonneau, P. Fernbach, A. Seppälä, M. Campana, N. Mansard, and F. Lamiraux, "HPP: A new software for constrained motion planning," in *IROS*, 2016.

[15] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[16] R. Munos *et al.*, "From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning," *Foundations and Trends® in Machine Learning*, vol. 7, no. 1, pp. 1–129, 2014.

[17] S. Garrido-Jurado *et al.*, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit.*, 2014.

[18] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 683–698.

[19] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *RSS*, 2018.

[20] J. Tremblay *et al.*, "Deep object pose estimation for semantic robotic grasping of household objects," in *CoRL*, 2018.

[21] G. Wilfong, "Motion planning in the presence of movable obstacles," *Annals of Mathematics and Artificial Intelligence*, 1991.

[22] S. Russel and P. Norvig, "Artificial intelligence: A modern approach," 2003.

[23] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, 1968.

[24] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov, "Combining neural networks and tree search for task and motion planning in challenging environments," in *IROS*, 2017.

[25] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *ICRA*, 2017.

[26] D. Silver and A. H. et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, 2016.

[27] J. Wu, B. Zhou, R. Russell, V. Kee, S. Wagner, M. Hebert, A. Torralba, and D. M. S. Johnson, "Real-time object pose estimation with pose interpreter networks," in *IROS*, 2018.

[28] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker Jr, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge," in *ICRA*, 2016.

[29] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6D object pose estimation," in *ECCV Workshops*, 2016.

[30] R. Horaud and F. Dornaika, "Hand-eye calibration," *The international journal of robotics research*, vol. 14, no. 3, pp. 195–210, 1995.

[31] J. Heller *et al.*, "Structure-from-motion based hand-eye calibration using L-∞ minimization," in *CVPR*, 2011.

[32] V. Loing, R. Marlet, and M. Aubry, "Virtual training for a real application: Accurate object-robot relative localization without calibration," *IJCV*, 2018.

[33] J. Tobin *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," *IROS*, 2017.

[34] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real viewpoint invariant visual servoing by recurrent control," in *CVPR 2018*, 2018, p. 8.

[35] S. James and P. e. a. Wohlhart, "Sim-to-Real via Sim-to-Sim: Data-efficient robotic grasping via Randomized-to-Canonical adaptation networks," *CVPR*, 2019.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CVPR*, 2016.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[38] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002.