Rearranging blocks into densely packed configurations

Filippos E. Sotiropoulos and Ryan J. Sandzimier

***Abstract*—In this project we implement task and motion planning for an object rearrangement problem. A collection of cube blocks are arranged in a densely packed configuration by utilising a combination of pick-and-place as well as non-prehensile pushing actions. Utilising a Probabilistic Roadmap planner to choose control actions to relocate single blocks and a recursive algorithm, called Piecewise Linear Non-Monotone Rearrangement Search, to plan the sequence of blocks to move, we are able to successfully plan a sequence of actions to rearrange handfuls of blocks from a random original configuration to the densely packed grid. The method is demonstrated on an experimental setup.**

***Index Terms*—Robot manipulation, block rearrangement, pick-and-place, pushing**

I. Introduction

IN this paper we perform a block rearrangement task. Using pick-and-place as well as pushing actions we manage to rearrange blocks from a random initial configuration to a densely packed configuration using a robot arm equipped with a two-finger gripper. Due to the dense packing, the final desired configuration cannot be achieved by only picking up and placing the blocks. Instead pushing the blocks is necessary so as to slide blocks between others.

The problem of manipulating multiple blocks to a desired configuration is a very specific example of what broadly could be classed as a combined task and motion planning problem [1]. As such various method have been applied to the problem such as evaluating good heuristics for considering the planning problem in an integrated manner [2] or methods based on discretizing the workspace [3]. In addition methods that leverage the environment to align blocks for dense packing arrangements has been proposed [4]. In this project we utilize a search based approach to solve the rearrangement problem [5] in conjunction with a low level planner.

II. Problem Statement

The problem at hand is to manipulate wooden blocks from some scattered configuration into some desired ordered configuration. The blocks rest on a flat surface and are constrained to planar motion on the table top. Fig. 1 illustrates an example of an initial configuration and final configuration of the blocks. The configuration can be represented as a list of block poses, where each block pose contains the position (x, y) and orientation θ. A planner should determine a sequence of actions for the robot to take to reach the desired configuration and the robot should execute this sequence of actions. The planner should not choose any actions for which there is a collision. Two blocks should never collide. The gripper should never collide with a block that is not purposefully being manipulated.

F. E. Sotiropoulos (fes@mit.edu) and R. Sandzimier (rsandz@mit.edu) are with the Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, USA.

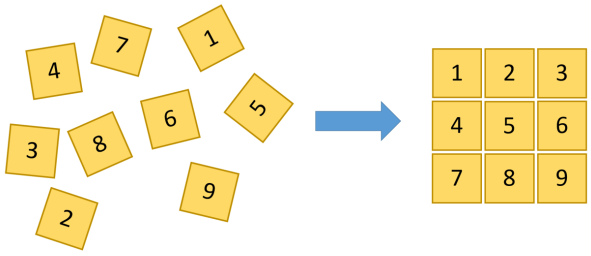

Fig. 1. The task is to manipulate the blocks from the scattered state (left) to an ordered state (right).

Using a two-finger gripper, the robot can manipulate a block by picking it up at one location and placing it down at a different location. We refer to this as a pick-and-place action. A pick-and-place action is parameterized using the initial block pose (x, y, θ), final block pose, and the grasp orientation. In general, there are two pairs of block faces the gripper can grasp and the grasp orientation specifies which one to use. Fig. 2 illustrates the pick-and-place actions that can be used to move a specific block in a given configuration. Since the final configuration requires the blocks to be densely packed, there are cases where the robot cannot move the block to its final location using a pick-and-place action. Fig. 3 illustrates such a case. In this example, moving block 6 to its final location using a pick-and-place action would result in the gripper colliding with block 3, 5, or 9. Therefore, we require another way of manipulating the block to handle these cases. By pushing on one side of a block with the gripper using two points of contact, the robot can push an object in a straight line. We refer to this as a push action. Fig. 3 illustrates the push actions that can be used to move a specific block in a given configuration. A push action is parameterized using the initial block pose (x, y, θ), the push direction, and the push distance. Push actions must move the block perpendicular to the block face that is being contacted.

III. Method

We take a hierarchical approach to the problem. A high level planner, called "piecewise linear non-monotone Rearrangement Search" (plRS) [5], which is designed to deal with complicated cases of object rearrangement, determines the sequence of blocks that must be moved to transition between different configurations. A Probabilistic Roadmap

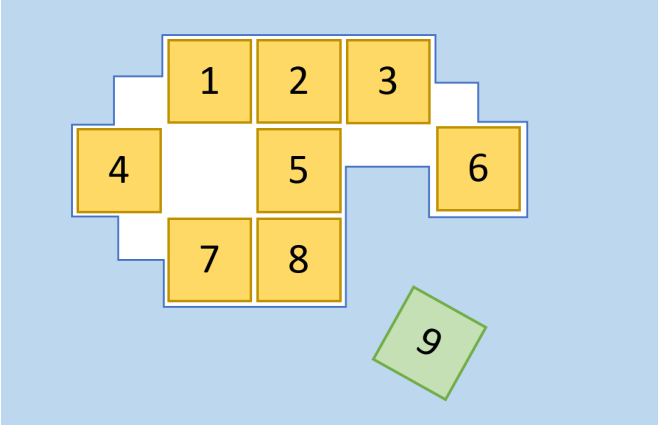

Fig. 2. Reachable region (blue) using pick-and-place action on block 9.

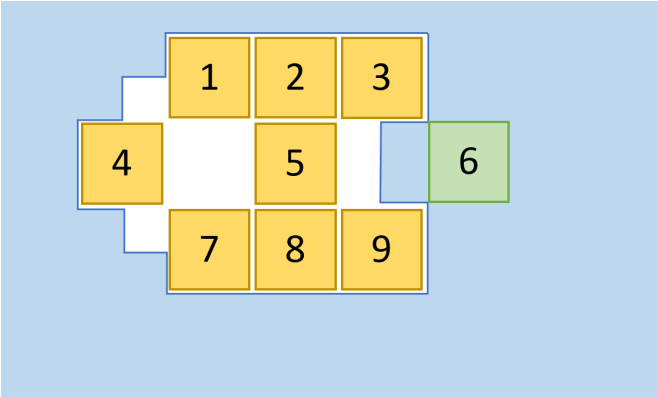

Fig. 3. Reachable region (blue) using pick-and-place action on block 6. Block 6 cannot be moved into its final location using a pick-and-place action because other blocks are in the way of the gripper fingers.

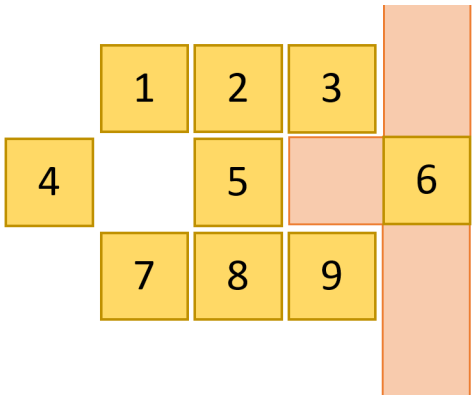

Fig. 4. Reachable region (orange) using pushing action on block 6. Block 6 can be moved into its final location without collision using a pushing action.

(PRM) path planner [6] is then used to determine the specific actions to move a specific block from one configuration to another. In the subsequent subsections, the two planners and their integration is detailed.

A. Piecewise Linear Non-Monotone Rearrangement Search

The plRS algorithm is a recursive method that attempts to move blocks into their goal configuration while allowing blocks which have not yet reached their goal location to be cleared. The full algorithm pseudocode can be seen in Algorithms 1 and 2. A single call of the plRS algorithm takes in a target object o to move to its goal, a set of objects which are still remaining to place $\mathcal{O}_R$, the current block configuration α_C and the final goal configuration α_F. It then returns a path, as described by a list of subsequent actions to perform on the system.

The first step in every call of plRS is to call the low level motion planner (details of which are provided in Section III-B) which generates a motion plan, and set of requisite actions, to move a single block from one configuration to another. This planner considers all objects that are in $\mathcal{O}_R$ as not being an obstacle to avoid. As such it takes in o, α_C, α_F and $\mathcal{O}_R$. It then returns a set of actions if found, $\pi_{\mathcal{U}}$, and also the set of blocks which are in the way but can be moved, $\mathcal{O}_b$. If an obstacle-free path is found for that block then the algorithm proceeds to recursively call plRS for the remaining blocks until all blocks have been placed while concatenating the series of actions which will place each block at each lower layer of recursion.

On the other hand, if a block is in the way the method will call the CLEAR algorithm (Algorithm 2) which recursively attempts to move blocks which are in the way to a configuration where they are no longer obstructing the path of the original block that was trying to be placed in its goal state. This method leverages the configuration sampling method outlined in Section III-B to find a valid configuration to which the block can be moved (line 1). Again, the path to move the cleared block may be obstructed and so lines 6-11 detail how the path may again be recursively cleared.

B. Probabilistic Road Map

As specified in Section III-A, the probabilistic road map (PRM) planner takes in o, α_C, α_F and $\mathcal{O}_R$ and returns a sequence of actions, $\pi_{\mathcal{U}}$, and the set of blocks that are in the way and must be moved first, $\mathcal{O}_b$. In this section, we describe our implementation of this PRM planner.

First, the PRM planner samples N intermediate configurations. The PRM planner is meant to plan a sequence of actions to move block o from its starting configuration in α_C to its final configuration in α_F. Since only one block is being moved, α_C and α_F differ only in the pose of block o. The sampled intermediate configurations should also only differ from α_C and α_F in this way. Therefore, only the pose of block o needs to be sampled for each intermediate configuration. The sampled block o poses are allowed to collide with any blocks in the set of remaining blocks, $\mathcal{O}_R$. However, any block o poses that collide with blocks that

Algorithm 1: plRS$(o, \mathcal{O}_R, \alpha_C, \alpha_F)$

$\pi_{\mathcal{U}}, \mathcal{O}_b \leftarrow \text{PRM}(o, \alpha_C, \alpha_F, \mathcal{O}_R)$;
if $\pi_{\mathcal{U}} \neq \emptyset$ & $\mathcal{O}_b == \emptyset$ **then**
  $\alpha_C[o] \leftarrow \alpha_F[o]$;
  if $\mathcal{O}_R == \emptyset$ **then**
    return $\pi_{\mathcal{U}}$
  for *each* $o_r \in \mathcal{O}_R$ **do**
    $\pi \leftarrow \text{plRS}(o_r, \mathcal{O}_R \setminus o_b, \alpha_C, \alpha_F)$;
    if $\pi \neq \emptyset$ **then return** $\pi_{\mathcal{U}}|\pi$;
else
  $o_b \leftarrow$ a block in $\mathcal{O}_b$;
  if $o_b \in \mathcal{O}_R$ **then**
    $\alpha_C, \pi' \leftarrow \text{CLEAR}(o_b, \mathcal{O}_R \setminus o_b, \alpha_C)$;
    if $\pi' \neq \emptyset$ **then**
      $\pi \leftarrow \text{plRS}(o, \mathcal{O}_R, \alpha_C, \alpha_F)$;
      if $\pi \neq \emptyset$ **then return** $\pi'|\pi$;
return $\emptyset$

Algorithm 2: CLEAR$(o, \mathcal{O}_R, \alpha_C)$

$\alpha_F \leftarrow \text{SAMPLE\_NEW\_POSE}(o, \alpha_C)$;
$\pi_{\mathcal{U}}, \mathcal{O}_b \leftarrow \text{PRM}(o, \alpha_C, \alpha_F, \mathcal{O}_R)$;
if $\pi_{\mathcal{U}} \neq \emptyset$ & $\mathcal{O}_b == \emptyset$ **then**
  $\alpha_C[o] \leftarrow \alpha_F[o]$;
  return α_C, $\pi_{\mathcal{U}}$
else
  $o_b \leftarrow$ a block in $\mathcal{O}_b$;
  if $o_b \in \mathcal{O}_R$ **then**
    α_C, $\pi' \leftarrow \text{CLEAR}(o_b, \mathcal{O}_R \setminus o_b, \alpha_C)$;
    if $\pi' \neq \emptyset$ **then**
      α_C, $\pi \leftarrow \text{CLEAR}(o, \mathcal{O}_R, \alpha_C)$;
      if $\pi \neq \emptyset$ **then return** α_C, $\pi'|\pi$;
return $\emptyset$

are not in $\mathcal{O}_R$ should be thrown out and re-sampled. In this way, we guarantee the sampled poses do not collide with any previously placed blocks. This task suffers from the narrow passage problem. In some cases, a push action is required to move a block to its final location. In order to execute this action, there must be an intermediate block o pose along a straight line perpendicular to a block face. Sampling block poses uniformly, the probability of sampling a block pose along this line is zero. Therefore, we bias the sampling so that with probability ε the block pose is sampled along one of these lines and with probability $1-\varepsilon$, the block pose is sampled uniform randomly over the workspace. N and ε are tunable design parameters.

After sampling the intermediate configurations, we construct a road map by checking each pair of configurations and connecting them with any any feasible actions that could accomplish this configuration transition. These connections are uni-directional. That is, it is possible for there to exist an action to transition from configuration β_1 to configuration β_2, but there does not an exist any action that would transition from configuration β_2 to configuration β_1. Also, there may be multiple connections between configurations. For example, it is possible that a pick-and-place action or a push action are both feasible actions to accomplish some configuration transition. In addition, it is possible that a pick-and-place action using either of the two possible grasps would accomplish a transition, in which case there would be one connection per grasp. The planner checks which actions could execute the transition and then checks if there are any collisions between the block or gripper and other blocks. If there are collisions with blocks not in $\mathcal{O}_R$, the connection is not allowed. However, collisions with blocks in $\mathcal{O}_R$ are allowed. Each connection contains information about the associated action as well as the list of blocks in collision.

Once the road map is constructed, the planner finds the shortest path from the starting configuration to the goal configuration (if one exists) using Dijkstra's algorithm. Each connection in the road map has an associated cost, which is based on which type of action is taken and the number of blocks in collision as a result of the connection. The cost function takes the form:

$$C = \alpha_1 \mathbb{1}(\text{is pick-and-place}) + \alpha_2 \mathbb{1}(\text{is push}) + \alpha_3 N_c \quad (1)$$

where N_c is the number of blocks in collision due to the connection, α_1 is the cost of a pick-and-place action, α_2 is the cost of a push action, and α_3 is the cost per collision. α_1, α_2, and α_3 are tunable design parameters. In general, it is best to choose $\alpha_2 > \alpha_1$ because push actions are less precise than pick-and-place actions. We discuss the choice of cost parameters in more detail in Section V and suggest some areas for improvement.

IV. Experiment

To test our method, we implemented the above problem on a UR10e Universal robot arm equipped with a Robotiq two-finger gripper (Fig. 5). Another important component to implementing the algorithm on a real system was being able to measure the configuration of the blocks relative to the robot system. To simplify the acquisition of configuration information for the blocks, we used fiducial markers (April-Tags), which were affixed to the top of each wooden block. These were observed from an overhead camera whose pose was calibrated with respect to the robot base.

Furthermore in our task and motion planning method we abstract away the details of the motions that the robot must take to perform either the pick-and-place or push actions. We use the on-board position control and inverse kinematics and as such must only stream position setpoints for the robot to follow. For both actions, reaching the robot pose from which to perform the actions is done by simply moving the robot in the space above the top of the blocks to a point directly above the target location. The arm then lowers the gripper to perform the required action. If a block is being picked then the gripper is centered at the block center. For a push action, the gripper is first partially closed, so that the two fingers can be used as a two-point contact end effector, and then it is lowered just behind the target block.

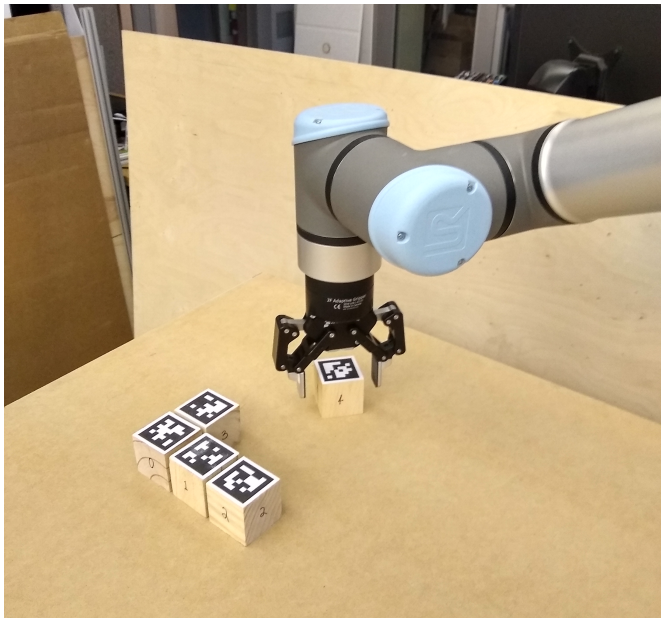

Fig. 5. The experimental setup used to test out block arrangement methods. We utilise a UR10e Universal Robots arm, a Robotiq two finger gripper. The blocks we are manipulating are 2 inch softwood blocks on a MDF surface. An overhead camera is used to measure the location of the blocks based on fiducial markers.

V. Results and Discussion

The proposed method was successful at planning the task for up to 9 blocks. The computation time for the planning was dependent on the number of blocks and the size of the workspace. For 4 blocks, the planning took less than a few seconds. For 9 blocks with a large workspace, the planning typically took around 20-30 seconds. When the workspace is reduced, the computation time increases. For 9 blocks with a small workspace, the planning took up to 2 minutes. For large workspaces, blocks can be cleared into the extra empty space, which reduced the number of times blocks need to be cleared out of the way more than once. When the workspace is smaller, it is less likely that there will be paths with no collisions. Therefore, more recursions are required to clear blocks out of the way, which leads to longer computation time. For 9 blocks and a small workspace, typically 20-25 total actions are needed to complete the task.

The cost function plays an important role in how the task is planned. Namely, the parameter α_3 that controls the cost of collisions can affect the typical paths. For example, consider a case where block 1 can only be picked up in one grasp orientation because block 2 is in the way of the other grasp orientation. At the goal location, block 1 must be placed down in the opposite grasp orientation because of a collision with block 3. One feasible plan is to clear block 2 to an intermediate location. After doing this, the robot can grasp block 1 with the correct grasp orientation. Another option is to move block 1 to an intermediate location, and then re-grasp in the opposite grasp orientation. The parameter α_3 affects this choice. For small α_3, the planner will prefer to clear block 2 out of the way and then move block 1 using only 1 action. For large α_3, the planner will prefer to move block 1 using 2 actions (switching grasp orientations in between) due to the large cost of having to clear blocks out of the way.

There are some areas for improvement with our implementation. Including some measure of distance in the cost could help to reduce the time required to complete the task. This would ensure the planner does not waste time clearing blocks very far away if it is not necessary to do so. By adding a penalty term for the distance of a push, this could help with the precision of the block manipulation. Push actions are less precise than pick-and-place actions. The push actions become less accurate the further the blocks are pushed. Therefore, it would improve accuracy if the planner preferred push actions that were short rather than long.

There is also room for improvement in the way block poses are sampled when clearing blocks. In our implementation, when a block needs to be cleared out of the way, it samples a uniform random pose for that block that does not collide with any other blocks. There are no guarantees that the sampled pose will actually be out of the way, which could lead to having to clear blocks multiple times unnecessarily. Another improvement could be to attempt to place a cleared block at its goal location if possible. When a block needs to be cleared out of the way, instead of moving it to a random intermediate position, the planner could first try to place it at its correct final pose. This could help optimize the number of actions required to keep the task.

References

[1] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013. [Online]. Available: https://doi.org/10.1177/0278364913484072

[2] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, *FFRob: An Efficient Heuristic for Task and Motion Planning*. Cham: Springer International Publishing, 2015, pp. 179–195. [Online]. Available: https://doi.org/10.1007/978-3-319-16595-0_11

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

[4] A. S. Anders, L. P. Kaelbling, and T. Lozano-Perez, "Reliably arranging objects in uncertain domains," in *IEEE Conference on Robotics and Automation (ICRA)*, 2018. [Online]. Available: http://lis.csail.mit.edu/pubs/anders-icra18.pdf

[5] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement," in *Robotics: Science and Systems (RSS)*, Rome, Italy, 07/2015 2015. [Online]. Available: http://www.cs.rutgers.edu/~kb572/pubs/Krontiris_Bekris_rearrangement_RSS2015.pdf

[6] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug 1996.