DOM-BOT: A Precision Domino Placing Robot

Elijah Bell
MechE
Massachusetts Institute of Technology
Cambridge, MA, USA
elibell@mit.edu

Jonathan Zhang
EECS
Massachusetts Institute of Technology
Cambridge, MA, USA
jonzhang@mit.edu

Abstract—**Domino placing is a complex, precise and tedious task that even humans find challenging. In this paper, we explore the potential for autonomous domino placing with a KUKA iiwa 7 DoF robotic arm. Our approach integrates point cloud reconstruction, segmentation, and pose estimation to determine optimal grasps, while precision movements are executed through motion planning with Kinematic Trajectory Optimization. The system successfully generates domino grasp poses and demonstrates setup and knockdown of dominoes in a line. Future work could focus on improving runtime efficiency and expanding the range of domino configurations.**

Index Terms—**Robotic Manipulation, Stacking, Pick and Place, Inverse Kinematics, Kinematic Trajectory Optimization**

I. Introduction

The pastime of placing and knocking over dominoes has existed for hundreds of years, with dominoes being invented in China at least as early as the 12th century [1]. For each of the hundreds of years of domino stacking, people have manually placed each domino carefully, by hand, into an inherently unstable position, before knocking them down. A short duration domino toppling can require dozens of dominoes to be placed for even a couple seconds of payoff, and any mistakes will bring the attempt right back to where it started. There exists a good opportunity to update this process using some of our technological advancements since the 12th century.

If a robotic approach to stack dominoes was developed, with arbitrary paths able to be programmed in, people would be saved the hours of tedium of placing dominoes by hand. Instead, they could enjoy designing the structures and simply sit back and watch the robot build the domino run for them, getting all of the reward with far less work. In addition, if the system doesn't need a sorted set of dominoes to place from, users could simply dump out a sufficiently large pile onto a table, saving time and effort that otherwise would be spent sorting. Further, any knocked down structure could just be reassembled or reused in a new location, allowing for the domino stacking and toppling to continue perpetually.

Others have created autonomous domino machines in the past, all consisting of some sort of wheeled base, dispensement mechanism, and magazine of prearranged dominoes. These machines are widespread enough that cheap models are available on Amazon or constructable through a DIY tutorial on Instructables [2]. Each of these machines is good enough for placing down one-, maybe two-, dimensional arrays of dominoes in smooth continuous paths. However, if one wanted to avoid the tedium of placing dominoes into 3D structures as done by some advanced builders [3] [4], these machines wouldn't give you any help. A more sophisticated approach would be necessary for truly arbitrary domino construction.

We present a robotic manipulator which could achieve such arbitrary construction, albeit with a kuka iiwa robot arm likely too expensive for almost all but perhaps the most elite domino constructors. While the project is thus limited in real life implementation, we do hope it is both a cool art project and an interesting demonstration of work which can act as another block in a developing set of projects for others to build off of in the future.

II. Related Work

Not much work has been done in the specific domino stacking problem, but there have been some pick and place problems that are similar to our work that we can learn from, particularly in that of precision and stacking.

A. SimPLE, a visuotactile method learned in simulation to precisely pick, localize, regrasp, and place objects [5]

Maria et. al explored pick-and-place solutions with increased accuracy. In precise pick-and-place, also known as kitting, their robot transforms an unstructured arrangement of objects into an organized arrangement. The approach, dubbed SimPLE (Simulation to Pick Localize and placE), learns to pick, regrasp, and place objects using the object's computer-aided design (CAD) model, without any prior experience or encounters with the specific objects. Such high accuracy prior informed grasp selection would be relevant to our project.

B. Autonomous robotic stone stacking with online next best object target pose planning [6]

Fadri Furrer et al. explore autonomous stone stacking with robotic manipulation. They were able to stack towers of rocks in eleven consecutive trials. This relates to our project of stacking dominoes because of the pick and placing of objects in different orientations into specifically precalculated structures.

III. Approach

Our approach can be divided into two main stages: grasp generation and trajectory planning. Grasps are generated from depth information combined with segmented color images to calculate domino bounding boxes and return grasp pose information. We then perform precise trajectory planning with Kinematic Trajectory Optimization to transform from the desired grasp to the exact placement of the dominoes in a line without knocking any over in the process.

A. Environment

Our system consists of a Kuka iiwa robotic arm, a WSG two finger robotic gripper, two Intel Realsense D415 depth and RGB cameras, a table, and dominoes which were produced in CAD and ported in as obj files. The robotic arm is welded to the table and all dominoes are defined as floating base bodies. All setup and simulation was done in Drake, with visualizations done through Meshcat.

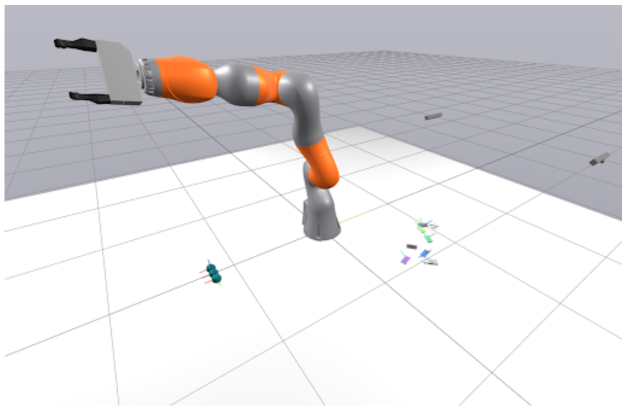


Fig. 1: An image from meshcat of the Kuka iiwa robot arm, some dominoes, three domino goal poses as spheres, two floating Realsense cameras, and a white plane for the table.

Our plants are all created with drake's built in MakeHardwareStation, and throughout the simulation, we maintain two plants: the main physics plant that we visualize in Meshcat, and a shadow copy plant without the dominoes and with a fixed gripper that we use for motion planning (further detail on the two plant structure in the trajectory planning section).

B. Grasp Generation

The domino chosen to be grasped and moved to the desired position must first be selected from the unsorted pile. This pile is created by allowing N dominoes to fall and allowing physics to determine how they land. As a result, the pile has dominoes resting on top of one another at various angles and orientations. From this state, our process of outputting the desired grasp for a domino is as follows:

1) **Take a picture with the Realsense Camera:** The Realsense combination depth and color camera is positioned above and to the side of where the dominoes fall into a pile. This orientation gives it the benefit of capturing both the top and sides of the dominoes, which allow for a more complete point cloud reconstruction.
2) **Generate masks with the Segment Anything Model (SAM):** Using the color image captured from the camera, Meta's Segment Anything Model is used to output a series of masks. Masks are generated for every object by prompting the model with samples taken from a grid all over the image.
3) **Generate point clouds and bounding box:** Now using the generated masks and depth image, we reconstruct the corresponding point cloud for each mask using the camera intrinsics and a pinhole model of the camera [7]:

$$X = \frac{z(x - c_x)}{f_x} \tag{1}$$

$$Y = \frac{z(y - c_y)}{f_y} \tag{2}$$

$$Z = z \tag{3}$$

4) **Downselect bounding boxes:** Before these bounding boxes can be used, they must first be filtered for faults since the segmentation process returns overlapping masks. As a result, some of the bounding boxes contain multiple dominoes or other undesirable parts of the scene. We know the volume and dimensions of a domino, so any points too far from the center are removed, and the bounding box is reevaluated to see if it matches the correct volume plus/minus some tolerance. Outliers are also removed based on if they have enough neighbors within a certain radius sphere.
5) **Return singular bounding box and grasp:** Using the list of orientations of dominoes output from the previous step, the best candidate is then chosen. Although multiple factors influence this idea of ”best grasp candidate”, we implemented a simple algorithm that just returns the domino which has it's largest face oriented most parallel to the floor. The condition for such a domino to exist is naturally enforced by simulating enough extra dominoes in the pile generation stage.

C. Trajectory Planning

To path plan accurate and precise enough trajectories for domino placing, we developed an approach with inverse kinematics, specifically kinematic trajectory optimization. In order to do so, we worked with Drake's KinematicTrajectoryOptimization class, finding and defining a B-spline path through an optimization of costs and constraints.

The constraints we set are as follow:

- **Position Constraints:** Initial and final position constraints at domino grasp and domino placement
- **Orientation Constraints:** Initial and final orientation constraints at domino grasp and domino placement
- **Velocity Constraints:** Velocity constraints to ensure the robot starts and ends with zero velocity
- **Collision Constraints:** Collision constraints evaluated at multiple points along the trajectory with fine-grained sampling to ensure a minimum safe distance between the robot and any obstacles

However, beyond the typical kinematic trajectory optimization setup, the main challenge for our task was the extremely

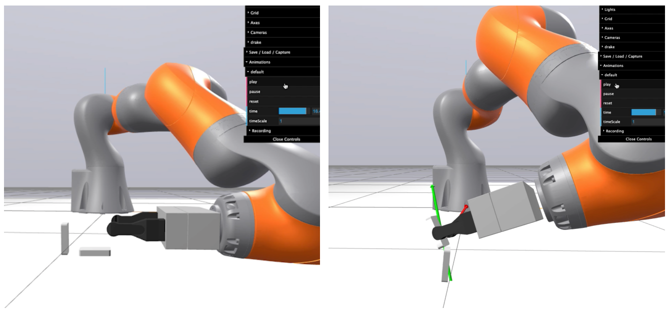

(a) Domino Misplaced (b) Domino Knocked Over

Fig. 2: Examples showing some of the difficulties encountered along the way due to the very small size of the dominoes and precise orientations we needed. On the right we see the domino failing to be placed upright with any margin of error in the constraints. On the left we see one domino knocking over an already placed one on its trajectory.

strict position and orientation constraints we needed to set. Due to the small size of the dominoes and spacing between them, we had to be able to find paths while defining no tolerance buffer between the lower and upper bounds of position and orientation constraints.

To address these challenges, we implemented two additional strategies to supplement our trajectory optimization. The first strategy involved minimizing the number of variables in optimization problem to reduce its complexity. We achieved this by defining a shadow plant in the background which mirrored the robot's initial setup, but excluded the dominoes and used a welded-finger version of the wsg gripper. By doing so, we focus only on the seven joint positions of the robot arm, simplifying the optimization process. After computing a trajectory for the shadow plant, we synchronized it with the main simulation, ensuring that the start and end poses in the trajectories would always align.

The second strategy was the introduction of an intermediate pose along the trajectory from the pile of dominoes to their placement. The strict constraints we set often caused the kinematic trajectory optimization to struggle finding viable paths. By breaking down the trajectory into two stages, we significantly improved the success rate of the optimization. To ensure seamless execution, we wrote a function to scale the two trajectory segments to align in time, enabling smooth transitions and reliable robot commands.

IV. Evaluation and Discussion

A. Grasp Generation

The intended goal of this project was to pick up a series of dominoes from an unsorted pile and place them into a user defined domino run. Beyond a basic line of dominoes, increasing complexity could be added by increasing the number of the dominoes or making the domino run more intricate. Key performance metrics we were aiming for included if the model can identify and provide a good domino grasp candidate, if it can pick up a domino, if it can place a domino, and if it can pick up and place multiple dominoes.

To validate the grasp generation stage for these metrics, each of the steps outlined in the approach section have their results shown here.

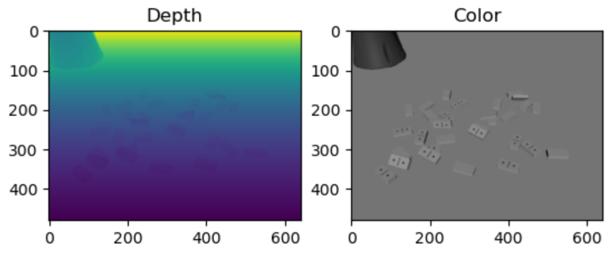


Fig. 3: **Grasp Generation Step 1.** A depth map and color image of the dominoes once they have settled into a pile. The color image is black and white simply because the world lacks color within the texture files.

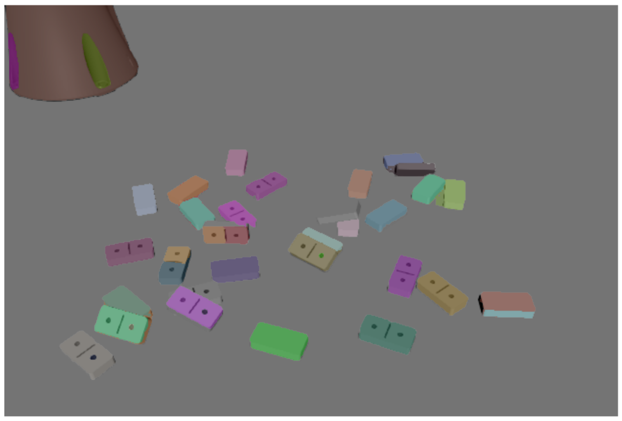


Fig. 4: **Grasp Generation Step 2.** An image showing the segmented masks of the image from the camera. Each color represents a different mask, and some masks are overlayed on top of each other. In this image there are 30 dominoes but 41 masks, so masks clearly must be filtered.

The implemented point clouds reconstruction works well, accurately overlaying the point clouds over the dominoes. However, because some of the masks generated by SAM cover more or less than just one domino, the resulting point clouds needed to be cleaned up. After the bounding boxes were generated off of the point cloud and the original masks (Fig. 5), filtering and adjustment was done.

First, the bounding boxes were filtered by volume. Since the size of the dominoes is known, bounding boxes which have a volume that is either too big or too small could be removed. While this did remove incomplete point clouds of partially obscured dominoes, this was acceptable since the domino which is doing the obscuring would still be in the list and is a better grasp candidate anyway. Volume filtering also worked to eliminate overlapping overly specific point clouds, such as can be seen in the purple and indigo domino and the

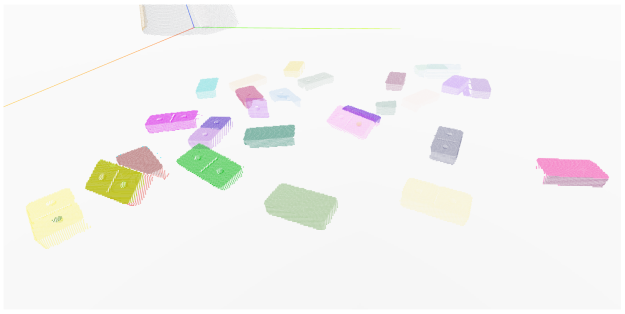


Fig. 5: **Grasp Generation Step 3.** A screenshot of meshcat showing the reconstructed point clouds using the masks from above.

yellow-green and red domino. The point clouds also include outliers, such as seen to the right of the green domino in Figure 6. Such outliers need to be removed to make the bounding box as tight as possible for good grasp generation. This is done by removing all points which miss a threshold of neighboring points within a certain radius, giving us nice bounding boxes shown in Figure 7. The yellow-green domino shown is now missing, but since it was not a prime grasp candidate, this is okay.

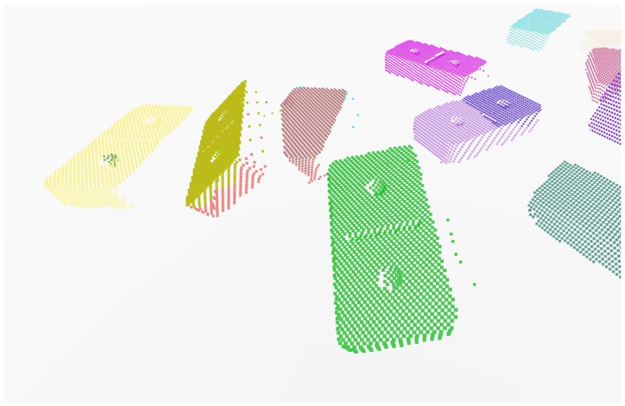


Fig. 6: A close up view of the segmented point clouds. Of note are the outlier points on the right side of the green domino.

Now that we have the bounding boxes, we want to assign an orientation to it, and one that is standardized to each of the sides of the domino. If we want to be able to grasp the domino in a particular way each time, we need to have the axes of the rigid transform be consistent. The choice of which of xyz goes to big, long, and small face is arbitrary as long as the same pattern is remembered for when we go to align the gripper to the object. They are also rotated to ensure the y(green) axis is facing upwards. Once these transforms have been generated, we are ready to select a domino for grasping. This selection process is based on a scoring process which ranks dominoes based on which dominoes have the flattest orientation. Improvements could be made to this approach by simply checking to see if there are other dominoes within a certain radius which could interfere with the grasp attempt.

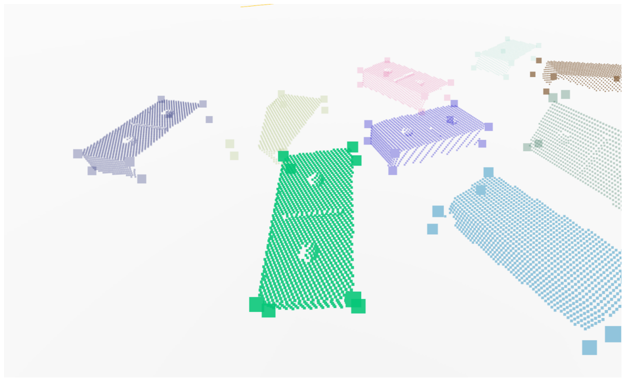


Fig. 7: **Grasp Generation Step 4.** Generated bounding boxes after filtering out outliers and point clouds which were too small.

The grasp returned is the one with the large triad as shown in Figure 8.

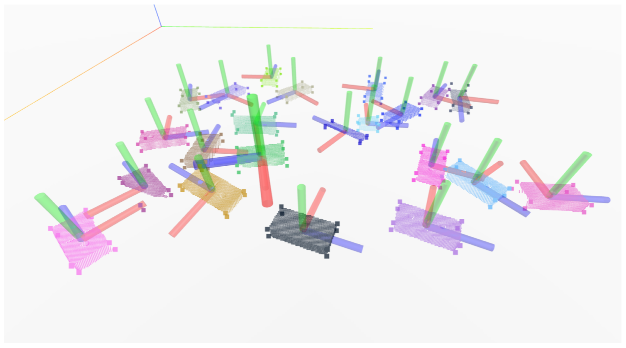


Fig. 8: **Grasp Generation Step 5.** Final bounding boxes and grasp orientations for each of the dominoes. The selected grasp to be attempted by the robot is the large triad.

B. *Trajectory Planning*

In the end, combining our grasp generation with our trajectory planning, our robot was able to successfully place a line of dominoes in the correct intervals such that they could be knocked down.

Limited in compute, we opted for a straightforward test of the robot's final capabilities: picking and placing three dominoes in a line before knocking them down. Each of the final destinations for the dominoes was entered in as a parameter, so more complex configurations should be possible with no modification to the underlying code.

The dominoes in this test all were flat on the ground before being picked up and placed in a line. The gripper was position controlled to close fully and squeeze the domino when it encountered resistance. In this demonstration, we opted for a simple orthogonal grip, but if we were doing more complex grips, this grip would be rotated about the medium axis of the domino so that when placing the arm approaches from an angle above so that it does not impact the floor. (Fig 9).

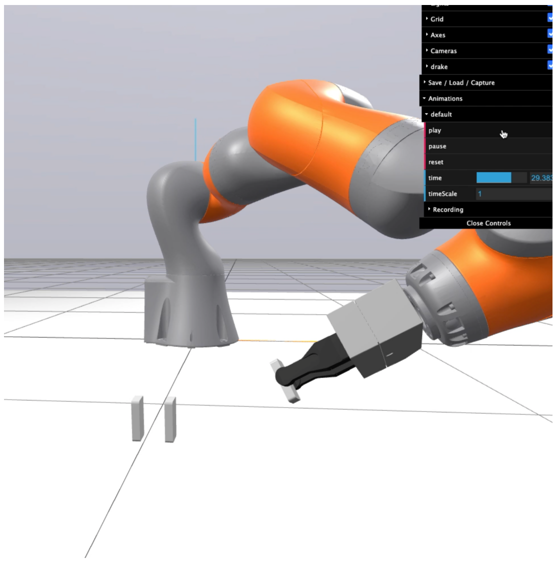


Fig. 9: **Domino Placing:** Robot successfully grabs domino and places it down.

Placing the dominoes was done once the arm reached the final position, again constrained in both position, orientation, and velocity. A simple position command was sent to open the gripper, and, as long as the domino was sufficiently close to the ground, the domino would land and stay in place (Fig 10).

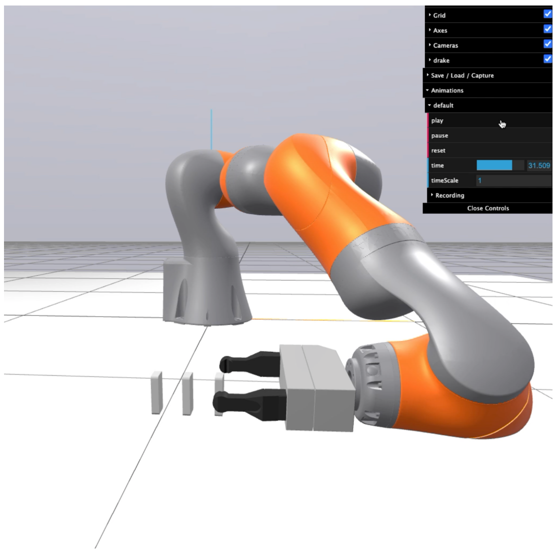


Fig. 10: **Domino Placed:** Robot successfully placed dominoes in a line.

Once all the dominoes were set up in the run, we moved the iiwa arm back and then triggered the front domino by executing a trajectory designed to strike it, causing the rest to topple over. (Fig 11).

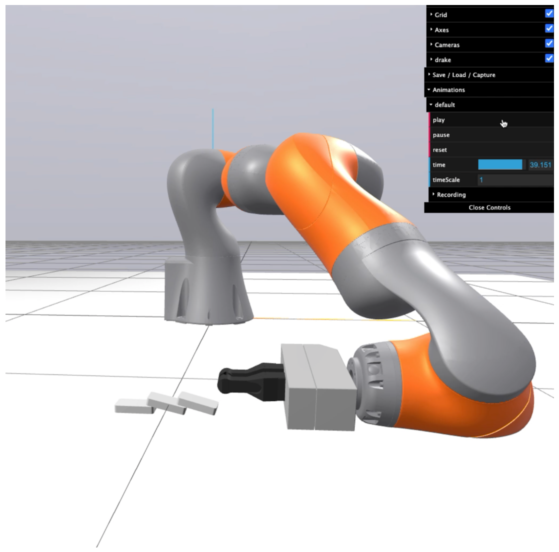
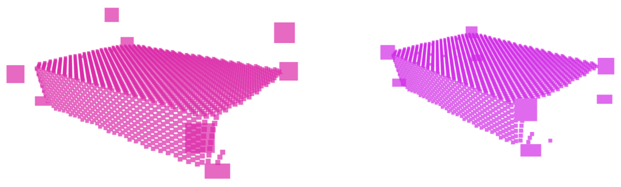


Fig. 11: **Dominoes Knocked Over:** Robot knocks over dominoes, showing the correct spacing between them.



(a) Incorrectly Rotated Bounding Box (b) Correct Bounding Box

Fig. 12: The same domino point cloud and bounding box shown in a) as correctly oriented with no outlier removal, and in b) as incorrectly oriented with outlier removal. Note the point in the lower center corner of the domino which is removed. There are also points on the other side of the domino which were removed as well.

C. *Discussion and Improvements*

The generation of grasps overall worked well: grasps were generated for at least 70% of the dominoes, and each generated grasp was almost always centered and oriented on the domino as desired. Earlier tests using statistical approaches to removing outlier points from the point clouds ended up removing too many points so the height of the bounding box would be inaccurate. Filtering by volume required manual tuning of the volume cut off limits so as not to remove good point clouds that were only too big due to outlier points.

Surprisingly, it was sometimes the few outlier points which provided a useful constraint preventing bounding boxes from being incorrectly rotated as in Fig 12. Here, the outlier point is a correct point on the side of the domino, while in the green domino shown in Fig 6, they are a set of points projected from an entirely separate domino.

Even though there were two cameras set up in the simulation, only one was being used. Instead, if the segmented point clouds were generated from both cameras, and then merged based off of a metric like distance between centers, the point clouds would be more complete, making the bounding boxes and thus grasp generation more robust. The initial versions of the simulation were slow to simulate, but 20x improvements were made once a simplified box model was used instead of the mesh for collision calculations.

An improved grasp selection algorithm would likely also take into account efforts to maximize the selected grasp's distance to other dominoes and how high in z it is (if it's on top vs bottom of a pile).

In terms of motion planning, although the robot was able to successfully pick and place dominoes in a line, the process of getting to this point was not easy, and the end solution was not as robust as we hoped. Even with the strategy listed in our approach, Kinematic Trajectory Optimization, although performing great with the position constraints, proved to not be very consistent in finding paths for the strict orientation constraints we gave it. It failed to find the correct configurations for many different poses, and was especially bottle necked by how close the target goal pose was to the floor.

In future work, we would hope that we could make the path planning algorithm more robust. We would still use kinematic trajectory optimization for the more broad movements in the process, ex: bringing the gripper above the pile of dominoes and bringing the gripper above where it needs to place them, but perhaps a more precise and consistent algorithm for the last step of bringing it from that above point to actually picking up/placing the domino. We hope that this combination of a more coarse path planning algorithm with a fine tuned one, would allow it to still have a relatively good run time while also being able to consistently handle the more precise orientation constraints.

V. Conclusion

In this paper, we have demonstrated an approach to autonomous domino placing using a KUKA iiwa 7 DoF robotic arm. Our system effectively combines point cloud reconstruction, segmentation, and precise motion planning to execute grasp generation and domino placement. The results show that the system can successfully pick dominoes from an unsorted pile and arrange them into a predefined line.

Our work advances the automation of a traditionally manual and error-prone task, showcasing the potential for robotic manipulators in fine-grained pick-and-place applications. By leveraging modern segmentation models and trajectory optimization techniques, our approach showcases a novel application but also provides a framework that can be extended for more complex domino configurations or other related work.

Future research can focus on improving the runtime efficiency of the motion planning processes, which would make real-world implementations more feasible. Additionally, while we believe the robot should be capable of constructing 3D domino structures, this capability was not tested within this project. Further research could explore the feasibility of 3D structures and enhance the system's robustness to variations in the environment, expanding the capabilities of autonomous domino manipulation systems.

VI. Contribution Statements

Elijah Bell Worked on simulation setup and domino grasp generation

Jonathan Zhang Worked on kinematic trajectory optimization and the placement of the dominos in simulation

References

[1] Lo, Andrew. ”The Game of Leaves: An Inquiry into the Origin of Chinese Playing Cards,” Bulletin of the School of Oriental and African Studies, University of London, Vol. 63, No. 3 (2000): 389-406.

[2] Gzumwalt and Instructables, “Pink and green domino machine II,” Instructables, https://www.instructables.com/Pink-and-Green-Domino-Machine-II/ (accessed Dec. 9, 2024).

[3] T. Weissker, “Building Techniques — Domino-Tim,” Domino-tim.de, 2024. https://domino-tim.de/en/tech/ (accessed Dec. 16, 2024).

[4] “Tutorials,” Hevesh5. https://www.hevesh5.com/tutorials

[5] M. Bauza, A. Bronars, Y. Hou, I. Taylor, N. Chavan-Dafle, and A. Rodriguez, “simPLE: a visuotactile method learned in simulation to precisely pick, localize, regrasp, and place objects,” arXiv.org, 2023. https://arxiv.org/abs/2307.13133 (accessed Dec. 16, 2024).

[6] “Autonomous robotic stone stacking with online next best object target pose planning — IEEE Conference Publication — IEEE Xplore,” ieeexplore.ieee.org. https://ieeexplore.ieee.org/document/7989272

[7] “Robotic Manipulation,” Mit.edu, 2024, Russ Tedrake. https://manipulation.csail.mit.edu/ (accessed Dec. 16, 2024).