The Robot Librarian: Mobile Manipulation in a Library

Shiva Sreeram & Peter Holderrieth
Department of Electrical Engineering and Computer Science (EECS)
Massachusetts Institute of Technology (MIT)
Cambridge, MA, USA

***Abstract*—Libraries around the globe store the world's most critical human knowledge, but require laborious work to keep the books organized and sorted. The ability to use robots to do the manual labor of sorting these books would allow librarians to focus on research and core archival work. In this project, we aim to build a mobile manipulator that supports a librarian in their work and sorts books from a table into the correct shelf. Working in PyDrake, we created a virtual library environment containing shelves and a table with books on it. Using geometric perception combining ICP and graph-based clustering, a perception system recognizes the books positions on the table. Antipodal grasps are sampled, scored and ranked. Using RRT planning and inverse kinematics, actuator trajectories are found. In addition, we included further optimizations defining waypoints (e.g. right in front of the shelf) to allow planning to be more efficient. Limitations of our work are discussed. Overall, this work shows how a simple mobile manipulator can achieve accurate sorting of books using traditional robotic manipulation methods.**[1]

***Index Terms*—mobile manipulation, simulator, library, sorting**

I. Introduction

Libraries have long been revered as sanctuaries of knowledge and culture, fostering education and preserving human history. However, the maintenance and organization of these extensive collections can be a labor-intensive task, often consuming significant time and resources that could otherwise be devoted to more specialized archival and research work. By employing robots for the meticulous, yet essential, task of sorting and shelving books, librarians can be liberated from routine manual labor, enabling them to concentrate more on their core responsibilities such as curatorial work and engaging in scholarly research. This project, therefore, explores the feasibility and efficacy of a mobile manipulator designed to support librarians by autonomously sorting books. Beyond its apparent utility, a key motivation for picking this task was that it is complex but still simple enough to build a full-stack robotic pipeline from the ground up. Our goal was to get as far as possible with classical robotic methods while applying a diverse set of techniques learnt in the course *Robotic Manipulation* at MIT by Russ Tedrake [13].

II. Related Work

There has been analogous prior work in this space involving a manipulator attached to a mobile base. One example is the use of robots in the Amazon warehouses [3]. These robots are capable of manipulating the packages/bins in the warehouse to their desired locations, similar to how we intend to manipulate books to their shelf-space destinations. Similarly, the work from TRI in using a robot to grab groceries from shelves involves a related process with the order of events flipped (we are placing items on shelves whereas this picks items from shelves) [10]. The closest equivalent to our Robot Librarian project formulation is the Joe and Rika Mansueto Library at the University of Chicago [2]. In this setup, the library is entirely underground with high density storage of books. Their approach also uses perception but instead with book barcodes to identify books. While their setup is different in that the shelves are extremely tall and the gripper needs to be raised to super high bin locations rather than low shelf spaces, the Joe and Mansueto Library provides a unique inspiration to our project.

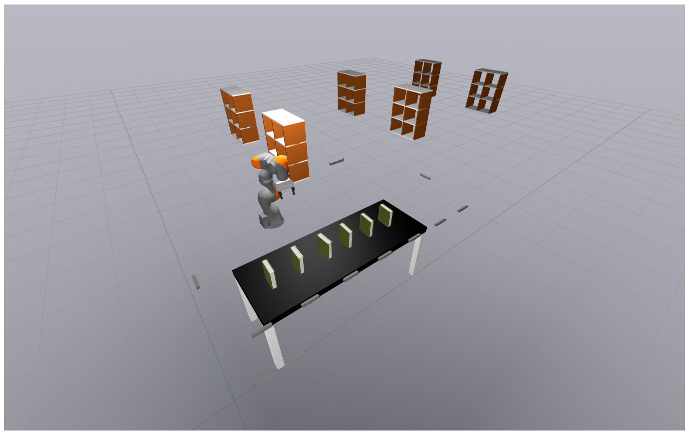
Fig. 1: Our simulation setup with the mobile iiwa arm, a table (black) with books (green), surrounding depth cameras, and 6 shelves.

III. Methods

A. Simulator Set-Up

As a first step in our project, we used PyDrake [14] to setup our simulation as it provides a flexible and high quality robotic simulation engine (see fig. 1). We created SDF files for the shelves and placed them in rows and columns with variable distances. Similarly, we created the table SDF file but with

[1]The code for this project is in the repo https://https://github.com/PeterHolderrieth/RobotLibrarian

hydroelastic collisions, as this is where we'll be grasping from. The locations of the shelves and the table are fixed throughout ("welded"). In particular, their coordinate frames are assumed to be known by the robot (as this could be measured once setting up the robot in the library). In addition, we initialized books (also with hydroelastic collisions) to stand upright on the table where both the number of books and their location is unknown to the robot. We included a set of 10 depth cameras surrounding the library table allowing perception of the books.

As a manipulation station, we use the Kuka IIWA robotic arm and a simple two-finger Schunk WSG 50 for our gripper equipped with hydroelastic collisions. We placed this station on a mobile base ("mobile iiwa") with minimal ability to change height ("z-axis"). It is important to note that this mobile base has no graphical representation in our simulation, so the robot appears to be sliding on the ground. Our setup can be seen in Figure 1. In order to execute a trajectory, we first take said trajectory for the mobile IIWA, which is in positional space, connect it to a discrete interpolator ("StateInterpolatorWithDiscreteDerivative" in Pydrake) to obtain the velocities and use an inverse dynamics PID controller ("InverseDynamicsDriver" in PyDrake) to control the IIWA. We then also use the Schunk position controller (including a discrete interpolator for velocities). The 3 main inputs to the system are therefore:

1) The position of (fingers of) the gripper.
2) The joint positions.
3) The positions of the x-,y-, and z-base.

B. Perception

To perceive the locations of the books, we built a perception pipeline as illustrated in fig. 2. We first extract point clouds from the depth images ("DepthImagetoPointCloud"). Then, we crop the point clouds to only contain points that are within a bounding box on the table (which is possible as we assume we know the geometry of the table). The point clouds are then merged and downsampled.

To segment the point clouds, we decided to use a clustering algorithm. As we assume that the robot has no access to the number of books, a standard clustering algorithm such as K-means [8] would be inappropriate as they require to specify K, i.e. the number of clusters. Therefore, we decided to use community detection algorithms on graphs such as Leiden clustering [15], where we use the 5-nearest neighbors graph on points in the point cloud. If the books are not in contact, these algorithms essentially found the connectivity components of the graphs that perfectly corresponded to books as the books are not in contact. In this case, we simply ended up using the connectivity components of the graph as a segmentation method.

For each individual book point cloud, we then run iterative closest point (ICP) [1] to perform point cloud registration. To increase accuracy and avoid being stuck in local minima, it was beneficial to initialize the ICP algorithm with its translation to be the mean locations of all members of the point cloud. Due to the symmetry of the book, ICP could result in frames rotated 180 degrees around the z-axis. While not strictly necessary, for grasping it makes sense to align the frames to point into the same direction. To fix this, we keep restarting ICP with random initial rotations until we find a frame whose x-axis (in red in fig. 2) is pointing away from the robot.

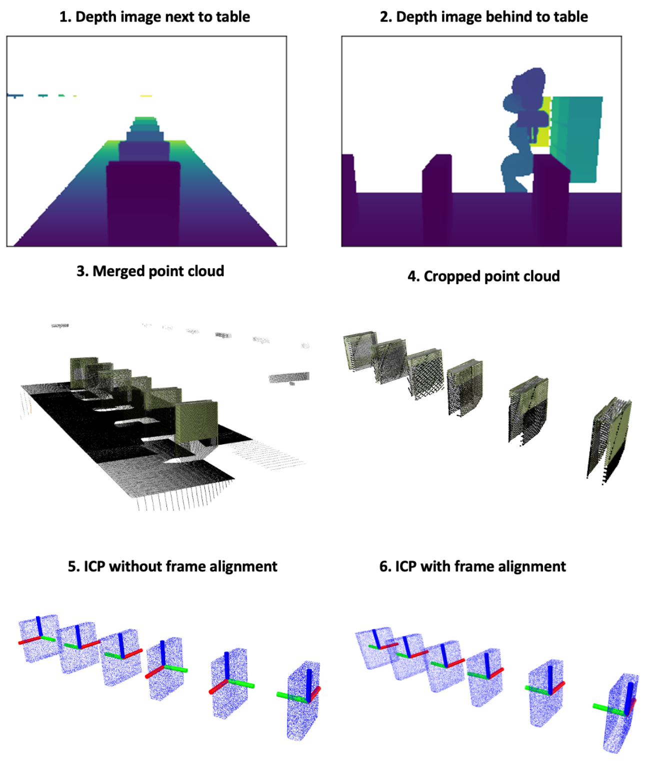
1. Depth image next to table
2. Depth image behind to table
3. Merged point cloud
4. Cropped point cloud
5. ICP without frame alignment
6. ICP with frame alignment

Fig. 2: Our perception pipeline. Depth images (1. and 2.) are merged to point cloud (3.) that is cropped (4.) and clustered into separate objects. After ICP, a frame for each object is found (5.) and then aligned (6.).

C. Antipodal grasping

To find a grasping position, we find antipodal grasping via sampling and a custom cost evaluation method (see fig. 3). Random points on a book point cloud are sampled and the finger of a free-floating gripper is aligned with the normal of that point. Random rotations around the normal axis are drawn and evaluated. If a grasp would result in a collision, we assign its cost $= \infty$. If there is no collision, we penalize deviations from vertical grasping position (i.e. we aim for grasps coming from above). We then rank all grasps found over 100 samples and select the top 5 as candidates which are selected to be the ones closest to the book but not in collision. To select the best grasp out of the 5 candidates, it turned out to be beneficial to compute the distance of the grasping frame position to the current gripper frame position and to select the one with minimal distance. The main reason for this is to simplify planning by avoiding that the robot picks a grasp that requires it to go around the table or to reach far over the table.

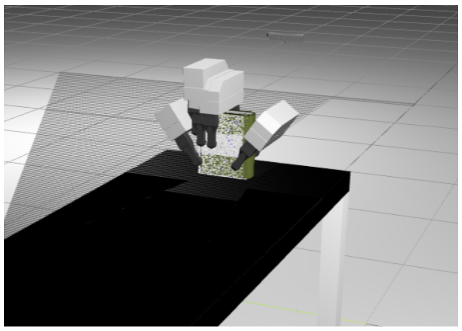
Fig. 3: Sampling and evaluation of grasp candidates (free-floating grippers) for a single book. Book in green. ICP point cloud in white.

D. Inverse Kinematics

To translate the antipodal grasp frames into joint space, we use inverse kinematics using Drake's optimization framework (e.g. using the SNOPT solver [4]). The position and orientation constraints in coordinates are the desired gripper positions up to a marginal error. Constraints in joints are the joint limits of the IIWA arm. To accelerate optimization, the joint limits of the mobile base were set such that impossible solutions, i.e. solutions from the mobile base coordinates to the desired gripper frame is higher than the distance of the fully extended iiwa, are avoided.

E. Planning book grasp trajectory

To plan the trajectory from the initial position ("home position", see fig. 1) to the book grasp, we use rapidly exploring random trees (RRT) [12]. Our reasoning to pick RRT as opposed to probabilistic roadmap [5] was as follows: we wanted to simulate a dynamic environment where books can be placed on the table while the robot is sorting the books into the shelf. In such a scenario, the robot should plan at each sorting iteration a new trajectory to a book. Therefore, we considered a single-query solution using RRT as more appropriate than a multi-query solution. To accelerate RRT planning, we made two modifications: 1) we reduced the size of the sampling space as much as possible to have the base not plan out of the vicinity of the table and 2) first planning to a pre-pick pose a distance away from the book and then planning into the antipodal grasp pose. This was helpful as our plans should now never collide into other books on the table, and also reduces planning time as the trajectory is able to be obtained in a far fewer number of max iterations of running RRT. Finally, we implemented a check for our RRT plan whether the length of the path from the initial position is longer than a threshold. This helped us avoid paths that go around the table and are suboptimal (see fig. 4).

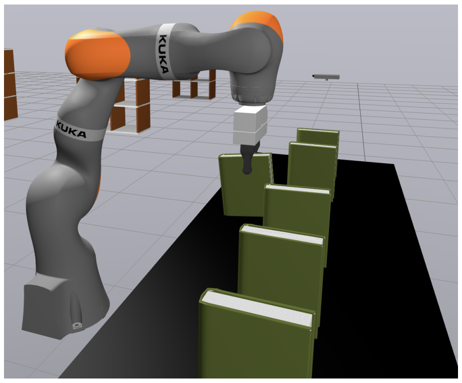

Fig. 4: Successful grasp of a book.

Once we have an RRT path list to the book defining the 10 dimensional array for the mobile IIWA, we need to define the entire trajectory for this grasp: one for the IIWA states and one for the WSG. We do this as follows: 1) move to grasp pose 2) wait for WSG to close 3) move in the reverse direction of our grasp pose move. The reason our step 3 is able to have such success is due to only selecting grasp poses above the book, so our robot minimizes the amount of dragging the book along the table.

F. Planning shelf placement

After we have a book in our WSG gripper and we have returned to our home position, we want to plan to a specific shelf and shelf position for which we also employ RRT with the following optimizations (see fig. 6):

1) **Pre-placement pose:** The first optimization is similar to the one we did for the book, where we plan to a pre-placement pose that is right in front of the shelf.
2) **Fixing joints:** To speed up planning, we reduce the dimension of the configuration space for planning to go to the pre-placement pose by only allowing to vary "x", "y", "z" of the mobile base and the first joint of the IIWA (allowing the iiwa to rotate easily). Beyond efficiency, this approach has the advantage that the motion it takes to the shelf is less unstable and as such the book will no longer fall out of the WSG (which we observed repeatable).
3) **Waypoints for distant shelves:** For the the shelves on the far back beyond the first row, we do a simple linear interpolation in the x-axis from the home position to a waypoint lined up with the pre-placement pose to save on compute time. We note that we managed to compute trajectories to each shelf even without waypoints, i.e. there are introduced for speed-up not for feasibility.
4) **"RRT-jump":** Another addition we made to the algorithm comes how we check a connection to the goal. In the off the shelf implementation we worked with, the only time it would check if we can connect to the goal is if we sample the goal. Instead, we now attempt to connect to the goal if the L2 norm of the

sample we're adding to our trajectory is within a certain distance (being 1.0). This way planning is faster for this large space we are in and we aren't limited by the probability of sampling the goal.

As for generating the trajectories for the mobile IIWA and WSG, we follow a similar process as we did for the book grasping: 1) plan into the shelf, 2) wait for the gripper to open, and 3) run the reverse of the plan to return to the home position. The amount of time we dedicate to these trajectories increases with the distance away each shelf is in order to have a smooth path that won't drop the book due to changes in momentum. Finally we note that we aren't planning with the book (i.e. don't explicitly check for collisions with the book) but plan as there was "nothing" in our gripper. The reason for this is that the book will need to collide the shelf anyway when it is placed and we not explicitly positions for positions within a shelf box.

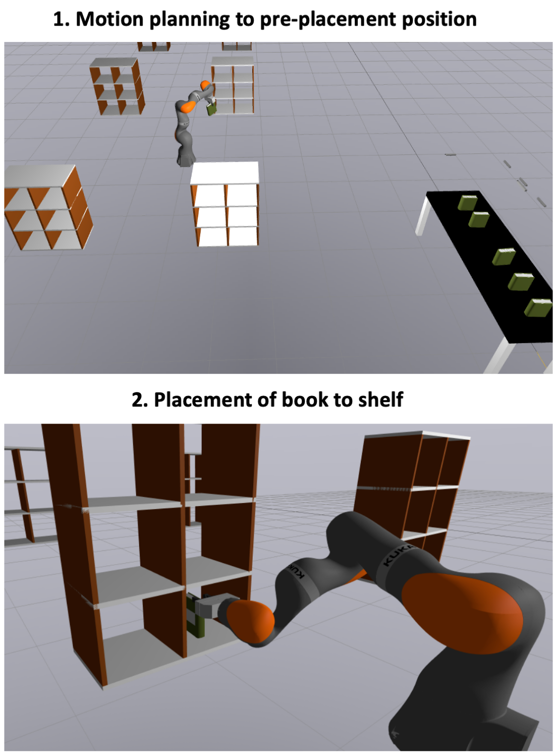
1. Motion planning to pre-placement position
2. Placement of book to shelf

Fig. 5: Successful placement of a book in the bottom-right position of a distant shelf.

IV. Results

With our pipeline, the robot could place an arbitrary number of books into the shelves. We managed to find iterations where all 6-10 books on a table have been placed correctly in various shelf boxes in various shelves (see presentation).

A. Evaluating Book Grasping Success

As we made additional optimizations to our grasping pipeline (see previous sections), our success rate of grasping a book improved significantly (see table I). Our solution is quite robust regardless which of the books we grasp from the table and with the hydroelastic collisions used, the book was less likely to slip out of the WSG once the grasp is complete.

Initial	Limit plan	Pre-grasp
0.2	0.3	0.9

TABLE I: We see the success rate of running ten grasps on a random book. With the initial approach, the robot tended to wrap around the table often and try to grasp the book between the cameras which tended not to work. Limiting the plan to only plan in front of the book added some success but needed more optimization (note initially this was done by only checking if the length of the path was short enough to not plan behind the table but this was later improved by limiting the joints). Performing a pre-grasp substantially improved the process as lining up with the book prevented any chance with knocking it over when coming at an angle. The only failures we observed in this case were usally an overshoot.

B. Evaluating Shelf Placing Success by Shelf

Next, we discuss the success rate of the shelf placing process for each given shelf. With the optimizations we made, we see that our approach is quite robust (as can be seen in table II).

Mode	Shelf A	Shelf B	Shelf C	Shelf D	Shelf E	Shelf F
No wp.	0.9	0.6	0.9	0.2	0.8	0.2
With wp.	-	-	-	0.7	-	0.6

TABLE II: We see the success rate for ten tries of placing a book in one of the random 6 positions on a given shelf. The naming conventions of a shelf are A-F from right to left, front to back (shelf A is closest to the home position of the IIWA where shelf F is the furthest). Note that for shelves D and F we generate a waypoint in the x direction from the home position the interpolate to first (the other shelves do not use such a waypoint).

C. Evaluating Improvements by RRT jump

Finally, we study improvements based on our modification of the RRT algorithm ("RRT jump") outlined above (see table III). We can see for the closer shelves, A and B, the speedup is negligible. However, for the much further shelves C and especially E the speedup is significant. Therefore, we can see how this optimization was able to speedup planning time.

V. Limitations and Future Work

a) Perception.: The first major limitation of our current pipeline is perception, in particular the assumption that the books stand up-right on the table and in minimal contact. We have tried to use large-scale vision models such as Segment

Mode	Shelf A	Shelf B	Shelf C	Shelf E
RRT	$< 0.2s$	$0.4s$	$> 5min$	$> 10min$
RRT-jump	$< 0.2s$	$0.2s$	$0.6s$	$8.4s$

TABLE III: Comparison time passed to plan various shelves with various algorithms. Note that we didn't evaluate D and F as the waypoint for those already "solves" the planning time issue (or is equivalent to planning to B).

Anything [6] to segment the point clouds. We combined it with the CLIP model [11] to retrieve the segmented part of the image that best described a specific description (e.g. "the book that is on the right of the table"). However, while approach turned out to be accurate for shelves, the model was not accurate for our books (see fig. 6). We also explored using novel open-set object detection algorithms such as Grounding DINO [7]. However, we could not get such models to run in the Docker container due to version issues and the fact that the model was prohibitively large. Therefore, deep learning offered no advantages over our naive clustering approach with geometric perception. However, further work could explore fine-tuning a deep learning model (e.g. a Mask R-CNN) to have a single model that not only segment point clouds but then also recognizes book labels that could indicate the shelf.

b) Failure modes and finite-state machine.: While we have built a finite-state machine as a Drake LeafSystem that is responsible for high-level planning and selection of trajectories, the final code of our project did not use that but runs with a constant trajectory source. The reason for that was mainly computational constraints: Our simulation crashed mid-way when the planning was performed after a full book-sorting iteration had already been executed. As there was no current limitation with a constant trajectory source, we ended up using this solution as it easier to debug and optimize hyperparameters with. However, including a finite-state machine has other advantages that future work should explore, e.g. allowing for failure modes. For example, when the book falls during delivery to the shelf, a finite-state machine could switch into recovery mode and try to pick up the book. To allow for that, we would need to integrate a camera into the mobile manipulator as the robot is currently "blind" outside a close surrounding of the table. Such a camera would also to assume that it is unknown whether shelf boxes are empty as this could be detected on the fly in this case.

c) Planning inefficiencies.: Our current planning strategy is inefficient for placing a book on the shelf as it is a single-query algorithm. The library environment (i.e. the shelves and the table) are static and a single pre-computation at the start to explore that environment would significantly increase efficiency. Therefore, methods like graph of convex sets (GCS) [9] or traditional methods such as probabilistic roadmap (PRM) [5] would offer a more concise solution for the shelf placement than RRT. In addition, the shelf geometries are the same across shelves. Therefore, once could leverage the planning for one shelf for the planning of the other shelf. Overall, this would lead a much more efficient planning algorithm once the pre-computation has finished.

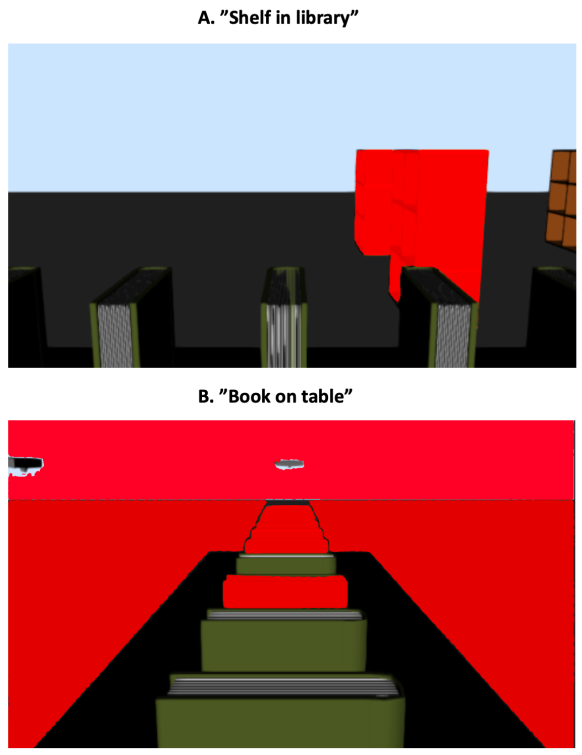

Fig. 6: Semantic segmentation pipeline using Segment Anything and CLIP. For shelves, the pipeline worked well (top), while for books the model keeps retrieving the background (bottom).

VI. Conclusion

Our Robot Librarian project has demonstrated the feasibility and effectiveness of using a mobile manipulator to autonomously sort books in a library environment. This system not only streamlines the sorting process but also alleviates the manual labor required from librarians. The use of PyDrake for simulation, combined with precise geometric perception, antipodal grasping, and RRT planning, has resulted in accurate solutions capable of identifying, grasping, and placing books on shelves with high success rates achieved through various optimizations, such as pre-grasp positioning and trajectory adjustments.

However, the project also highlights areas for future development. The limitations in perception, especially in recognizing and handling books in close contact, suggests a potential avenue for integrating more advanced deep learning models for object recognition and segmentation. Moreover, the incorporation of cameras into mobile manipulator would allow to program failure modes such as dropped books. Finally, planning inefficiencies, particularly for shelf placement, indicate a need to incorporate techniques, like GCS or PRM, that offer speed-ups in static environments like libraries.

We are convinced that with more compute resources and time, such limitations can be addressed with existing techniques. In conclusion, the Robot Librarian project therefore showcases the promising potential of robotic automation in library settings.

VII. Contributions

In regards to our joint efforts, there was considerable overlap in our roles. Consequently, it's challenging to clearly demarcate the boundaries of our individual contributions. Peter was predominantly responsible for the environment and simulation set-up and perception (including ICP), while Shiva was predominantly concerned with set-up of the robot and antipodal grasping. All other steps including RRT planning, planning optimizations, report writing, and preparing the presentation were done together.

References

[1] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes". In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606.

[2] University of Chicago. *The Joe and Rika Mansueto Library: How It Works*. 2011. URL: https://www.youtube.com/watch?v=ESCxYchCaWI.

[3] Scott Dresser. "Amazon announces 2 new ways it's using robots to assist employees and deliver for customers". In: (2023).

[4] Philip E. Gill, Walter Murray, and Michael A. Saunders. "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization". In: *SIAM Review* 47.1 (2005), pp. 99–131. DOI: 10.1137/S0036144504446096.

[5] Lydia E Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[6] Alexander Kirillov et al. "Segment anything". In: *arXiv preprint arXiv:2304.02643* (2023).

[7] Shilong Liu et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection". In: *arXiv preprint arXiv:2303.05499* (2023).

[8] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[9] Tobia Marcucci et al. "Shortest paths in graphs of convex sets". In: *arXiv preprint arXiv:2101.11565* (2021).

[10] Max Bajracharya, et al. "Demonstrating Mobile Manipulation in the Wild: A Metrics-Driven Approach". In: *Robotics: Science and Systems* (2023).

[11] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[12] "Rapidly-exploring random trees: A new tool for path planning". In: *Research Report 9811* (1998).

[13] Russ Tedrake. *Robotic Manipulation, Lecture Notes*. 2023. URL: https://manipulation.csail.mit.edu/index.html.

[14] Russ Tedrake and the Drake Development Team. *Drake: Model-based design and verification for robotics*. 2019. URL: https://drake.mit.edu.

[15] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. "From Louvain to Leiden: guaranteeing well-connected communities". In: *Scientific reports* 9.1 (2019), p. 5233.