Autonomous Data Collection for Surgical Procedures – Final Report

Micha Bosshart
ETH Zurich
bmicha@ethz.ch

Mischa Buob
ETH Zurich
buobm@ethz.ch

Colin Merk
ETH Zurich
cmerk@ethz.ch

Marco Trentini
ETH Zurich
marcotr@ethz.ch

Abstract

Recording surgeries is crucial for educational purposes and clinical research, particularly for training deep learning models in tasks like phase recognition, navigation, and surgical robotics. However, the operating rooms (ORs) pose challenges due to their cluttered and crowded nature, leading to occlusions that degrade the quality of the recordings. To address these issues, the proposed method aims to detect and avoid occlusions with a camera mounted on a robot arm. Autonomously collecting unoccluded data alleviates the need for additional healthcare personnel exclusively operating the camera. In this work, the fundamental algorithms for autonomous data collection, using an RGBD camera mounted on a robotic arm, are implemented. The packages and algorithms are designed in a modular manner, facilitating extensions in future work. Two main packages were developed: the obstacle interaction package, responsible for positioning the robotic arm, and the obstacle detection package, focused on detecting occlusions. All the packages are implemented in Gazebo ROS [8] [12]. The implemented pipeline successfully avoids occluding obstacles and positions the camera to maintain an unoccluded field of view. The average response time to occlusions was 7.85 s using our vision module and 4.96 s using the obstacle ground truth. Overall, this work lays the foundation for achieving fully autonomous surgical recordings and enabling efficient data collection for various applications, including surgical education and the development of advanced deep-learning models. The source code is available on GitHub: https://github.com/Divepit/mixed-reality-robot-arm-control-demo.git

1. Introduction

The integration of robotics in healthcare has opened new avenues for precision and consistency in medical procedures. This paper presents a project aimed at advancing this integration by developing an autonomous data collection pipeline for automated video documentation of surgical procedures.

This project utilizes the Kinova Jaco 2 [6] robot arm, equipped with a gripper-mounted RGBD camera, operating within a Gazebo ROS [12] [8] simulated environment. This environment mimics a surgical setup with dynamic, artificial moving obstacles, reflecting the complexities of a real-world operating room. Extensive research exists on target visibility planners as in [3] and [2], but none are employed in a comparable application as ours. Our contributions are twofold:

Firstly, we have developed an obstacle interaction package that manages the planning and movement of the robot arm, ensuring an unobstructed view of the surgical field. This package also facilitates the representation of obstacles within the simulation, enabling their detection and interaction.

Secondly, we have designed an image-processing module capable of detecting the position and size of obstacles. This module communicates with the interaction package, providing it with the necessary information to navigate the surgical field effectively.

In essence, this project represents a significant step towards automating the data collection process in surgical environments, with the potential to enhance the quality and efficiency of surgical documentation.

2. Motivation

As the field of healthcare continues to evolve, the need for technological solutions to address its complexities becomes increasingly apparent. The integration of robotics in healthcare, specifically in the context of surgical procedures, presents an opportunity to both augment the learning experience of new surgeons and ensure comprehensive documentation of surgeries. Our project, while not introducing a novel technology, combines existing ones in a unique way to address these needs.

The automation of surgical data collection could greatly aid the training of new surgeons. A comprehensive video record of surgeries provides material for instruction, allowing new surgeons to revisit procedures, dissect complex maneuvers, and learn from experienced practitioners. A more dynamic perspective, offered by an autonomous robotic system, could capture details that a stationary camera might miss due to occlusions.

In addition to educational benefits, an automated video documentation system could also have significant implications in legal contexts. Clear and comprehensive records of surgical procedures could be beneficial in cases of legal disputes or malpractice claims, providing a reliable record of events.

While the proposed system's primary function is to document surgeries, the technology has the potential for broader application. The use of computer vision algorithms for detecting and predicting obstacle positions could lead to a myriad of other uses, paving the way for further research and development in this field. As manual data collection in surgeries is time-consuming and cumbersome, data of surgeries is scarce. An automated data collection system could improve this and allow for more learning-based solutions in surgeries.

This project thus represents not only a practical solution to a specific problem but also a stepping stone toward exploring a wider range of applications.

3. Obstacle Interaction Package

The Interaction Package processes the obstacle positions and moves the camera to an unobstructed pose and position. The package receives the detected obstacle and target positions which are visualized and rendered into simulation, serving as input to the Obstacle Detection Package. Using either ground truth or the locations from the Obstacle Detection Package, it calculates possible camera positions and thereafter the optimal pose, which is fed to the MoveIt [4] planner which finally moves the camera. The package can be divided into five major components:

- Visualization and Rendering,
- Dome Generation and Occlusion Detection,
- Camera Position Cost Criterion,
- Pose Determination, and
- MoveIt Planner Integration.

3.1. Visualization and Rendering

Visualizations and renderings are a central part of our system, aiding in obstacle detection and the verification of obstacle avoidance strategies. The tools primarily employed for this purpose are RViz [5] and Gazebo [8], both of which are used to project the obstacles and the target into a simulated 3D environment. Obstacles in our simulations can be placed and manipulated in three primary ways:

- **Static Positioning**: Obstacles are manually positioned in specific locations. This method allows for controlled testing with stationary obstacles.
- **Dynamic Movement**: Obstacles are programmed to move along a predetermined path, typically in a circular pattern. This provides a more dynamic testing environment to assess the system's ability to avoid moving obstacles.
- **Periodic Occlusion**: An obstacle periodically appears at or moves to a position directly between the camera and the target.

These visualization methodologies are vital in ensuring that the simulation accurately represents the task requirements and that the robot end-effector is correctly navigating around obstacles to maintain a line of sight to the target. The simulation is running in Gazebo, which accordingly provides the true positions and shapes of all objects (Figure 1). RViz visualizes the position of the target and the robot while utilizing the obstacle detection package to detect other objects in the room (Figures 2 and 3).

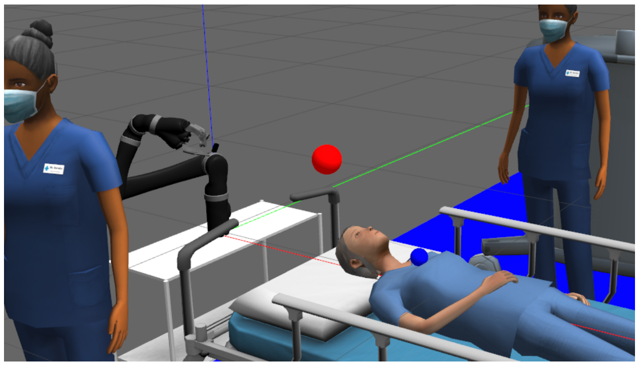

Figure 1. Snapshot of the Jaco 2 arm's end effector placed in a digital surgery room with a floating, spherical obstacle (red) and a spherical target (blue).

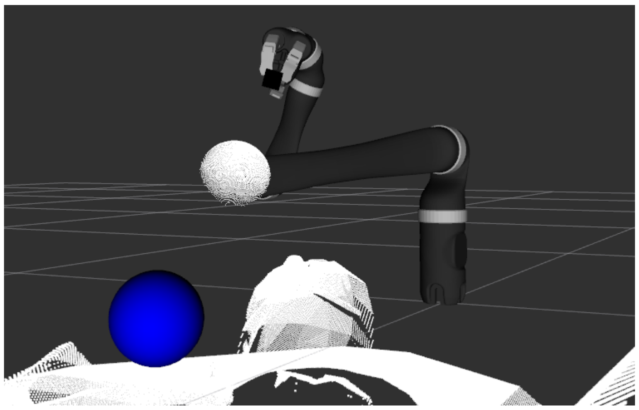

Figure 2. Snapshot of the Jaco 2 arm's end effector establishing line of sight with the target (blue sphere) by avoiding an obstacle (white sphere).

3.2. Dome Generation and Occlusion Detection

3.2.1 Dome Generation

The Dome Generation calculates camera positions by discretizing the configuration space into a dome-shaped grid. Each voxel represents a potential camera position. Note that the dome center is a design choice and not necessarily centered around the robotic arm.

The grid construction uses a radius and voxel count to determine the dome size and resolution, respectively. Each voxel is represented as a cube with a side length determined by the cube root of the dome's volume divided by the voxel count. The generator iteratively traverses the volume, checking each point for inclusion within the dome and occlusion. In RViz, occluded voxels are visualized in red, while unoccluded voxels are shown in green. By systematically following this process, the generator produces a dome-shaped distribution of voxels indicating viable camera positions.

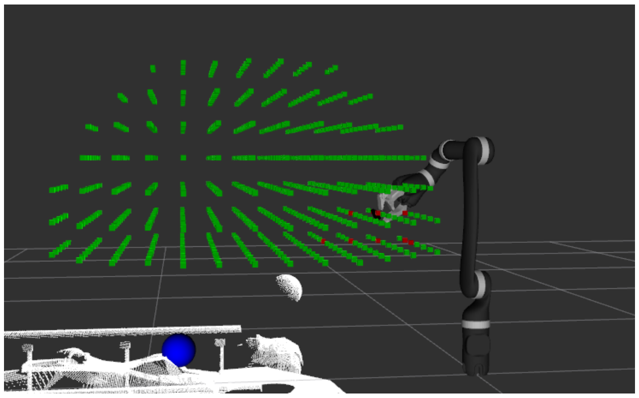

Figure 3. Snapshot of the Jaco 2 arm positioning its end-effector inside a pre defined voxel dome with occluded (red) and non-occluded (green) voxels.

3.2.2 Occlusion Detection Algorithm

The steps of the occlusion detection algorithm are as follows. For each obstacle, the vector $\vec{v}_{\text{co}}$ connecting the camera $\vec{c}$ and the obstacle center $\vec{o}$ is projected onto the direction vector $\vec{v}_{\text{ct}}$ connecting the camera $\vec{c}$ and the target position $\vec{t}$.

$$\vec{v}_{\text{co}} = \vec{o} - \vec{c} \qquad \vec{v}_{\text{ct}} = \frac{\vec{t} - \vec{c}}{\|\vec{t} - \vec{c}\|} \tag{1}$$

$$s = \vec{v}_{\text{co}} \cdot \vec{v}_{\text{ct}} \tag{2}$$

This allows us to determine the point $\vec{a}$ between the camera and the target that is closest to the obstacle in question.

$$\vec{a} = s \cdot \vec{v}_{\text{ct}} \tag{3}$$

Finally, the distance d between point $\vec{a}$ and the obstacle center can be determined as

$$d = \|\vec{a} - \vec{v}_{\text{co}}\|. \tag{4}$$

If d is smaller than the obstacle radius, the view is occluded. Otherwise we have found an unoccluded point.

These steps are repeated for every single obstacle for every voxel.

3.3. Camera Position Cost Criterion

The choice amongst all possible camera positions is made using a cost criterion. In our case we minimize the Euclidean distance between the current and the next camera position.

3.4. Pose Determination

The primary objective of pose determination is to calculate an end pose for the end-effector of the robot that allows the camera to point directly towards the target.

This calculation process employs basic concepts from vector geometry and quaternion mathematics. First, a direction vector $\vec{d}$ is computed, pointing from the current camera position $\vec{c}$ to the target $\vec{t}$.

$$\vec{d} = \frac{\vec{t} - \vec{c}}{\|\vec{t} - \vec{c}\|} \tag{5}$$

Once the direction vector is determined, a quaternion is derived from this vector. Note that the camera is oriented in positive z direction, denoted as $\vec{e}_z$.

$$\vec{q}_{\text{axis}} = \vec{e}_z \times \vec{d} \tag{6}$$

$$q_{\text{angle}} = \arccos\left(\vec{e}_z \cdot \vec{d}\right) \tag{7}$$

Quaternions represent the orientation of the camera using an axis and an angle of rotation about this axis.

By employing this approach, the end-effector is positioned in a manner that its camera directly faces the target, ensuring optimal video recording of the target.

3.5. MoveIt Planner Integration

Once we have determined the desired end-effector pose, this information is relayed to the MoveIt Planner [4] for trajectory planning. It is important to note that the MoveIt Planner is not inherently aware of any obstacles in the robot's environment; this information must be actively supplied. In a real world scenario, there would be the following options to facilitate the obstacle avoidance:

- Using the end-effector mounted camera to detect obstacles and relay this data to MoveIt.
- Implementing an additional, stationary camera system that observes the entire room, feeding information about potential obstacles to MoveIt.

4. Obstacle Detection Package

The obstacle detection package comprises vision algorithms responsible for identifying obstacles within the camera's field of view. Moreover, this package generates an abstract representation of the obstacles to facilitate communication with the interaction package. The key elements of this package consist of the segmentation algorithms discussed in Section 4.2 and the obstacle representation explained in Section 4.3. During the simulation, an RGBD camera with a resolution of 640x480 for both RGB and depth inputs was employed.

4.1. Overview

The obstacle detection package requires the RGBD images and the coordinates of the target and camera as inputs. As output, it produces a list of spheres that represent the observed obstacles. The spheres are defined by the radius and center coordinates.

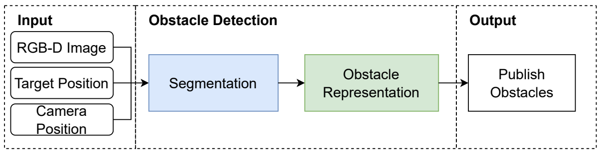

Figure 4. A top-level flowchart showing the components of the obstacle detection package.

4.2. Segmentation Algorithms

In order to identify and group together regions that are relevant to the line of sight, image segmentation is necessary. Several approaches can be employed for this purpose. Here, we discuss three different segmentation algorithms: Dynamic Depth Thresholding, learning-based depth segmentation, and edge detection-aided depth segmentation.

4.2.1 Dynamic Depth Thresholding

To facilitate Dynamic Depth Thresholding, the distance from the camera to the target can be determined using the Euclidean distance. This information is utilized to generate a binary mask from the depth image that distinguishes objects in front of and behind the target position. However, to prevent the target itself from being detected as an obstacle in the mask, the cutoff distance is adjusted by employing equation 8:

$$d_{th} = d_{target} - \max(d_{min}, \min(d_{target} \times f, d_{max})) \quad (8)$$

d_{th} denotes the adjusted threshold distance, d_{target} is the distance from the camera to the target, d_{min} is the minimum threshold distance, d_{max} is the maximum threshold distance, and f is a scaling factor. During testing, we achieved the best results with d_{min} at 0.1 m, d_{max} at 0.2 m, and an f of 0.15.

The binary image is then segmented by treating contiguous areas as individual segments, which are subsequently processed in the obstacle representation component. The main advantage of this algorithm lies in its simplicity. However, a notable drawback is that if multiple overlapping obstacles are present, they may be detected as a single entity and not separated into individual segments.

4.2.2 Learning Based Image Segmentation

Learning-based image segmentation assigns each pixel an instance which results in a segmented image. Compared to the Dynamic Depth Thresholding approach presented in 4.2.1, image segmentation can separate overlapping obstacles. Currently one of the most well-known RGB image segmentation models is Segment Anything Model (SAM) from Meta [7]. The downside of this model is that it requires 3 input channels and is computationally expensive. Accordingly, it is not viable for real-time applications. Since individual obstacles have a very distinct outline in depth images compared to their RGB counterparts, a segmentation algorithm using the depth image should yield better results. Even though this went well beyond the scope of the project and was not required, we attempted to perform knowledge distillation using the Segment Anything Model as the teacher and a UNet [9] as the student.

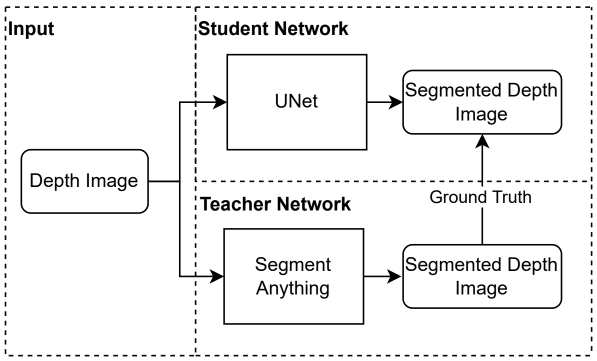

Figure 5. Flowchart of the knowledge distillation process

During the process of training the UNet, we encountered various challenges. One of these challenges was related to the order of the output channels, which proved to be irrelevant for our purposes. To address this issue, we employed a custom loss, which utilized the Sinkhorn algorithm [10] to select the appropriate channels. We subsequently applied the Binary Cross Entropy with Logistic loss as a cost function to the corresponding channels. However, this approach did not yield better results compared to using either the Dice Loss or the Binary Cross Entropy with Logistic loss without

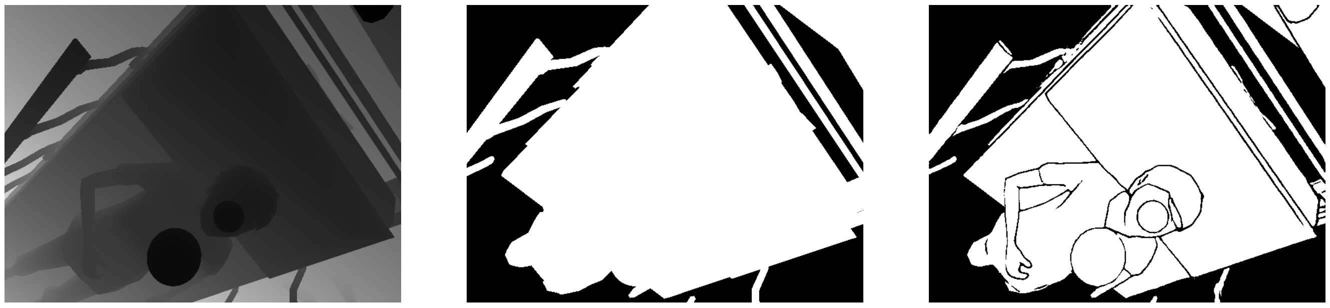

Figure 6. This figure illustrates the processing steps: original depth image (left), mask based on dynamic depth thresholding (middle), and dynamic depth threshold with DexiNed edges (right). To highligth the processing steps the target distances was increased to include parts of the background.

considering the channel order. Due to the limited duration of the project, we were unable to fully train our UNet model and integrate it into our pipeline for comprehensive testing.

4.2.3 Enhancing Depth Segmentation with Edge Detection

The drawback of the Dynamic Depth Thresholding method discussed in Section 4.2.1 was its inability to differentiate between multiple overlapping obstacles, treating them as a single large obstacle. To address this limitation, an initial attempt was made to employ the Canny edge detector to split the detected obstacles based on the edges detected. It was anticipated that this approach would be effective in depth images due to their distinctive features. However, during implementation, it was observed that the detected edges were not continuous, leading to incomplete segmentation of the images. Consequently, an alternative approach was sought.

A more sophisticated method was explored, involving the utilization of an edge detection model. For this purpose, the DexiNed model [11] was selected due to its strong edge detection performance.

The images in figure 6 demonstrate the significant improvement achieved by DexiNed when dealing with overlapping obstacles. Assessing whether the associated increase in runtime and subsequent reduction in frame rate is justified would require a final judgment based on real-world experimentation and evaluation.

4.3. Obstacle Representation

After the segmentation, each obstacle is known in the form of a binary mask and its corresponding depth values. The goal of this section is to simplify the obstacles to a representation that is consistent with what the interaction package expects. As obstacle representation, spheres are used to represent the shadow region with respect to the camera. The geometric properties of spheres enable an easy-to-understand and efficient method to identify obstructed regions as in section 3. The pseudo-code in figure 1 shows how the spheres are recursively generated. Representing the obstacles as spheres poses the problem that some objects – especially long and slender obstacles – produce large spheres. These spheres would be overly conservative, as the observed shadow region would be significantly larger than the real shadow region of the obstacle. To diminish this effect, the algorithm in figure 1 was implemented to recursively split obstacles along the second principal component if the resulting sphere was too large.

Algorithm 1 Algorithm to convert all obstacle masks to spheres.

```
1:  $spheresList \leftarrow []$
2:  for mask in obstacleMasks do
3:      procedure SPHERESGENERATION(mask)
4:          radius, center $\leftarrow$ sphereFromMask(mask)
5:          if radius $<$ max_radius then
6:              spheresList.append(radius, center)
7:          else
8:              mask1, mask2 $\leftarrow$ splitMask(mask)
9:              SpheresGeneration(mask1)
10:             SpheresGeneration(mask2)
11: return (spheresList)
```

5. Results

In order to assess the performance of our system, we conducted two evaluations: one utilizing our customized vision algorithm, and another employing the ground truth obstacle positions. Every 15 seconds, an obstacle is placed between the current camera position and the target, forcing our system to replan and move to a new un-occluded location. The parameter we measure is the absolute time the target is occluded during the entire process.

The results, as illustrated in Figure 7 and 8, demonstrate that our system achieves an average response time of 7.85 s

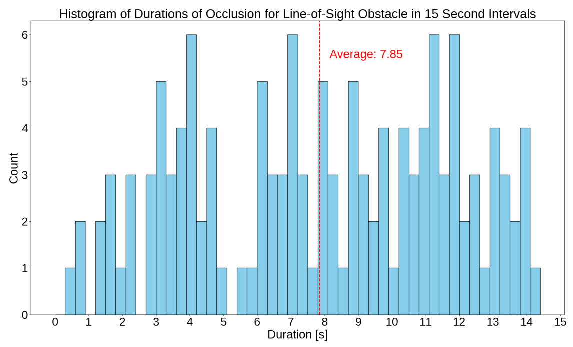

Figure 7. View obstruction during occlusion using the Obstacle detection node.

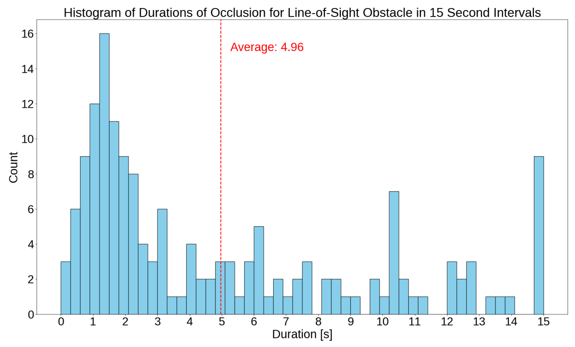

Figure 8. Results using the ground truth.

when utilizing the obstacle detection package, and 4.96 s when relying on the ground truth.

Method	Intel i5-6600K & GTX 1070	Intel i5-12600K & RTX 3070 Ti
Dynamic Depth Thresholding [ms]	0.247	0.160
Dynamic Depth Thresholding + Canny [ms]	0.873	0.461
Dynamic Depth Thresholding + DexiNed [ms]	72.027	25.484
Segment Anything [ms]	5518.937	1489.348

Table 1. Computation time of the presented algorithms averaged over 2000 runs.

In order to assess the performance of each segmentation algorithm individually, we conducted timing measurements on two separate systems. The outcomes, depicted in table 1, demonstrate differences in execution times between the newer and older hardware configurations. Notably, SAM exhibited significantly slower performance compared to other methods, rendering it unsuitable in its current form for our application.

6. Conclusion

Our project presents an approach to automating the data collection process in surgical environments by integrating existing technologies in a unique way. It opens up a plethora of possibilities for further research and development, particularly in the field of computer vision algorithms for obstacle detection and prediction.

However, it is crucial to recognize that while our project provides a solid foundation, it is not without its limitations. Currently, the system's accuracy and utility may decline when dealing with non-spherical obstacles or targets. There are also optimization opportunities in the algorithm that minimizes the displacement between the current and next camera position.

Another potential issue lies in the robot's movement. There are instances where the robot may swing around, losing sight of the target or may leave the predetermined voxel dome. Future work could focus on improving these aspects, ensuring the system's reliability in more complex and varied scenarios. Also, there are possibilities to make further use of the MoveIt planner's obstacle avoidance capabilities, like implementing an additional camera system that statically maps the room and provides information about obstacles not visible to the camera mounted on the end effector.

While these challenges present opportunities for further enhancements, the current implementation serves as a significant step towards automating the data collection process in surgical environments. The project is a testament to the potential benefits of integrating robotics into healthcare, paving the way for further advancements in this direction.

Acknowledgement

We would like to express our gratitude to our supervisors, Jonas Hein and Dr. Mahdi Rad, for their invaluable guidance, support, and suggestions throughout the project. We would also like to extend our sincere appreciation to Prof. Pollefeys, for providing us with the opportunity to undertake this project.

Work Distribution and Disclaimer

Micha Bosshart and Marco Trentini mainly worked on the obstacle interaction package while Mischa Buob and Colin Merk worked on the obstacle detection package. During development, ChatGPT [1] aided in debugging ROS, Python, and C++ code.

References

[1] ChatGPT (https://openai.com/blog/chatgpt).

[2] M. A. Baumann. *Path planning for improved target visibility: maintaining line of sight in a cluttered environment*. PhD thesis, University of British Columbia, 2009.

[3] J. Gemerek, B. Fu, Y. Chen, Z. Liu, M. Zheng, D. van Wijk, and S. Ferrari. Directional sensor planning for occlusion avoidance. *IEEE Transactions on Robotics*, 38(6):3713–3733, 2022.

[4] Ioan A. Sucan and Sachin Chitta. Moveit.

[5] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim. Rviz: A toolkit for real domain data visualization. *Telecommun. Syst.*, 60(2):337–345, oct 2015.

[6] Kinova. Jaco 2 robotic arm. https://www.kinova.com/, 2023. Accessed: 2023-05-28.

[7] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[8] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[10] R. Sinkhorn. A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *The Annals of Mathematical Statistics*, 35(2):876 – 879, 1964.

[11] X. Soria, E. Riba, and A. Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1912–1921, Los Alamitos, CA, USA, mar 2020. IEEE Computer Society.

[12] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.