Complete Visual and Geometric Object Reconstruction via Autonomous Robotic Manipulation

by

Evelyn Fu

S.B. Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2024)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2025

Authored by:	Evelyn Fu Department of Electrical Engineering and Computer Science May 16, 2025
Certified by:	Russ Tedrake Toyota Professor of EECS, Aero/Astro, MechE, Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

Complete Visual and Geometric Object Reconstruction via Autonomous Robotic Manipulation

by

Evelyn Fu

Submitted to the Department of Electrical Engineering and Computer Science
on May 16, 2025 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Accurately simulating object dynamics based on real-world perception inputs has wide applications in digital twins and robotic manipulation. Yet, doing so requires practitioners to carefully measure and reconstruct the dynamic and geometric properties of the objects, which is time-consuming and requires domain expertise. This project proposes an automatic pipeline to construct 3D representations from a collection of real objects, which can further be used to generate assets with accurate visual texture and collision geometry to be used in simulation. This pipeline will be designed to have minimal hardware requirements and aim to be efficient in time for physical actuation to maximize data collection on minimal hardware.

Thesis supervisor: Russ Tedrake
Title: Toyota Professor of EECS, Aero/Astro, MechE

Acknowledgments

This thesis is part of a larger collaborative effort that was recently released [1]. I am proud to have contributed to this broader project, and I gratefully acknowledge the collective work of the team involved. Please refer to the full project for additional context and results: https://scalable-real2sim.github.io/.

* * *

I would like to express my deepest gratitude to my thesis supervisor, Professor Russ Tedrake, for his invaluable insight, guidance, and expertise throughout this project. His mentorship and thoughtful input into this project were instrumental in shaping the direction and depth of this work.

I am especially grateful to Nicholas Pfaff, my mentor and collaborator, whose unwavering support, encouragement, and technical insight were critical at every stage of this research. Working alongside him has been a formative and enriching experience, and he has taught me so much as both a researcher and an engineer.

Additionally, I want to recognize my other collaborators on this project, Jeremy Binagia and Professor Phillip Isola for their insightful discussions and input throughout the project. I would also like to thank the entire Robot Locomotion Group for creating an intellectually stimulating and supportive research environment. I'm so grateful to have had the chance to work alongside so many creative and kind people, who made all the late night working sessions a fun and memorable experience.

I am thankful to the staff of 6.3800 for both funding my first semester and for providing me the opportunity to explore foundational ideas and skills that contributed to my development as a researcher.

Finally, I gratefully acknowledge the support of the Office of Naval Research (ONR) under Grant No. N000142412603, which funded my second semester and made this research possible.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Simulation is widely used in robotics research and development to speed up the setup, testing, and iteration of new algorithms and provide easily resettable training environments, among many other purposes [2–5]. For example, physics simulation has fueled many recent advances in robotics by enabling learning skills in simulation and executing them in the real world [6–8]. This approach, termed *Sim2Real*, is desirable as many machine learning-based approaches require large quantities of interaction data that are much easier to obtain in simulation than in the real world. An example of such a learning-based approach is reinforcement learning as seen in works such [6, 7]. Additionally, this approach is becoming increasingly valuable as robotics shifts toward foundation models, which require even larger training datasets [9].

The Sim-to-Real approach requires the simulated environment to be dynamically accurate to the real-world scene for algorithms developed in simulation to translate well into the real world. If the physical and visual properties of the simulated environment lack sufficient fidelity, policies trained in simulation may fail when transferred into the real world [8, 10, 11].

Currently, researchers must construct a replica of the real-world scene in which the robots will be trained. Building such a replica involves manually curating object geometries, which could involve the tedious approximation of object collisions using geometric primitives or creating more complex meshes using 3D art software, and tuning their dynamic parameters. This manual process requires domain expertise and is time-consuming. As scenes become more complicated with more types of objects, the burden to create simulations of all objects of interest increases. Consequently, it is difficult to scale this process to a large variety of real-world scenes, creating a Sim2Real gap. Therefore, we aim to develop a fast, reliable method for replicating real-life objects in simulation.

1.2 Contributions

This project applies aims to rapidly and autonomously construct a complete simulation asset description, such as a Unified Robot Description Format (URDF), for an object of interest that exists in the real world. These assets specify the object's appearance/visual geometry, the collision geometry used for collision checking during simulation, and the physical properties such as mass and moment of inertia. In particular, this project focuses on gathering visual and 3D geometric data of real-world objects, which includes visual texture maps to define the appearance of the surface of the object and mesh geometries to define the shape of the object in 3D, to obtain detailed collision geometries.

This data is gathered through autonomous interaction with the objects of interest, prioritizing collecting a complete observation of the object surface while maintaining efficiency and minimizing the complexity of the setup. In particular, we develop our method on a simple pick and place setup with a 7-DOF robotic arm, which can be easily modified to different hardware and is a common setup used by researchers and practitioners. This way, our method remains accessible and does not require specific hardware or unrealistic constraints. Additionally, pick and place is a standard factory task, so our automated pipeline that integrates into this task is a step towards obtaining data at scale from these factory setups.

There are two main components to this project.

1. A trajectory planner that commands a robot to manipulate a set of objects from a bin in front of an RGBD camera such that data of all surfaces is collected for each object

2. A data processing pipeline that ingests the collected RGBD images and learns a 3D representation of the object from arbitrary photometric reconstruction methods, so that our method can be applied to new photometric reconstruction methods as more powerful methods are developed in the future. From this representation, the pipeline further extracts a mesh geometry and texture map of the object, completing the visual and geometric reconstruction.

Chapter 2

Literature Review

2.1 Prior Approaches For Addressing The Sim2Real Gap

As Sim2Real gap is a widespread issue in reinforcement learning, various approaches to address the problem have been studied [12–20]. Given the diversity of simulation environments and the complex nature of real environments in which these policies act, these approaches vary in generality, assumptions, and intended applications.

2.1.1 Parameter Tuning with Existing Models

Many works have approached the problem of poor Sim2Real transfer by adjusting the parameters that dictate dynamics in the simulated environment.

One such established approach to training policies that are robust to the Sim2Real gap is domain randomization [12]. Domain randomization addresses this issue by randomizing environment parameters in the simulation over a distribution of visuals and/or dynamics. In this way, the trained policy can transfer into the real world as long as the real dynamics are within the distribution on which the policy was trained. This technique performs best with a good initial distribution approximation and is not very efficient otherwise.

Other works have aimed to estimate the correct parameters of the real world rather than train a policy generalized to a distribution of parameters [13, 14]. Like domain randomization, the need for a known model of the environment to estimate its parameters means although these approaches address the issue of poor Sim2Real transfer, for each training setting and object within the scene, a human still must manually construct a parameterizable representation in simulation.

2.1.2 Real2Sim in Robotics

Real2Sim [15] is the problem of automatically generating a digital replica of a real-world scene to reduce the simulation's Sim2Real gap. Works such as [16][17] focus on reconstructing the visual geometry of an entire static scene. Dynamic objects in the simulation of these scenes are assumed to be previously known. Le Cleac'h et al. [18] propose a Real2Sim system for reconstructing an object's geometry from observations and identifying its dynamic parameters, such as mass and moment of inertia. They first obtained an implicit model of the object's geometry from RGB images and then proposed a novel simulation approach for directly simulating their implicit model. Their method achieves dynamically accurate object-level Real2Sim from only RGB observations. However, most current simulations [21–23] and model-based approaches still use explicit representations of geometry. Wang et al. [19] proposed a Real2Sim system for constructing explicit object mesh representations from depth images. Torne et al. [20] presented a Real2Sim system that reconstructs a 3D scene from video stream and provided a GUI to add joints to create articulated environments for RL training. Most importantly, all of these pipelines do not include any method of automatically obtaining the RGB images to get a full view of the object. We additionally construct an automatic pipeline that interacts with objects to get data of the entire surface.

2.2 3D Reconstruction

3D reconstruction methods focus on reconstructing geometry from real-world observations such as RGB and depth images. These methods produce either implicit [24–28] or explicit [29–31] representations. Explicit methods represent geometry directly, such as with a triangle mesh. Implicit methods indirectly represent geometry using a continuous function and consequently can represent geometry in greater detail. Neural Radiance Fields (NeRF) [25, 32] and Gaussian Splatting [28] are implicit methods we apply our object-centric, occlusion robust 3D reconstruction formula to due to their popularity and the numerous investigations into variations of these methods, some of which may be better suited for future users of this method. These methods and their variations fall under the category of *neural rendering*, which is a class of techniques where rendering of the represented scene is differentiable, enabling end-to-end training with gradient descent. From these implicit representations, mesh extraction algorithms such as Marching Cubes [33] and Poisson reconstruction [34] can be used to produce a mesh geometry.

2.2.1 NeRFs

NeRFs were introduced by Mildenhall et al. [25] and represent a 3D scene with a learned continuous function $F_\theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$ that maps a spatial location $\mathbf{x} = (x, y, z)$ and

viewing direction $\mathbf{d} = (\theta, \phi)$ to a view-dependent RGB color $\mathbf{c} = (r, g, b)$ and volume density σ.

NeRFs are trained from images of a static scene where the camera pose within the scene is known for each image. Given the camera pose, a synthetic image of the scene from the viewpoint of the camera pose can be rendered from the NeRF by marching along the rays from the camera's origin for each pixel, compiling the learned colors at the spatial locations along the ray using the accumulated learned densities.

2.2.2 Gaussian Splatting

Gaussian Splatting is a neural rendering method introduced by Kerbl et al. [35] with the key advantage of much faster training and rendering times than the previous state of the art, NeRFs, while preserving quality and high-resolution renderings. Gaussian Splatting takes the same input as NeRFs but instead represents the scene using a set of *3D Gaussians*, defined by a position (mean) μ, covariance matrix Σ, opacity α, and spectral harmonics which represent color. Similar to NeRFs, the optimization process compares images rendered from the set of Gaussians to the ground truth images from the captured dataset.

2.3 Next Best View Selection

The quality of 3D reconstructions produced by the above-mentioned methods is highly dependent on the quality and completeness of the training data. Active 3D reconstruction dynamically acquires 3D data by intelligently choosing new viewpoints to observe areas of lower confidence, based on previously collected data. The process of choosing future observations based on the completeness of information collected from previous observations has generally been referred to as *Next Best View Selection* (or Next Best Configuration) [36–38]. This approach is particularly beneficial for scenarios with occlusions, textureless surfaces, or ambiguous geometry as it ensures the dataset of collected observations is fully informative of the scene.

Multiple approaches have been explored regarding active reconstruction and next best view (NBV) selection. ActiveNeRF [39] and ActiveGS [40] implicitly model uncertainty at spatial locations with the learned color and density representations. These approaches construct an information gain metric based on the mapped uncertainty to evaluate and select future views to gather data from. Alternatively, Jin et al. [41] proposed a mapless planning framework for NBV selection where a neural network is trained to predict uncertainty at new view candidates given previously collected images from nearby views. This method does not require an uncertainty map of the scene or an implicit neural reconstruction with uncertainty to be updated online while taking more samples.

2.3.1 Next Best View Manipulation Planning

Previous investigations have also investigated manipulation planning to build a complete 3D reconstruction of the surface of the manipulated object. Krainin et al. [36] developed a mobile robot system that grasps and manipulates objects in front of a depth camera to create a full 3D surface reconstruction, optimizing to display less-seen surfaces in trajectory planning and considering occlusions from previous grasps when planning new grasps. Kobayashi et al. [37] developed a similar system, but with a dual-arm setup instead.

2.4 3D Scanners

Custom hardware setups have also been used to reconstruct real-world objects with high fidelity, rather than relying on planning methods. In 2022, Google Research released a dataset of 1030 high-quality simulatable reconstructions of everyday objects, collected with a custom scanning rig where images are taken in a highly calibrated setting [42]. This dataset has proved to be immensely helpful for training Sim2Real policies on objects that are seen in the real world [43–45].

2.5 Research Gap

The Google Scanned Objects dataset method of collection provides a direct, mostly autonomous method of reconstructing real-life objects into high-fidelity simulatable assets, but their custom, highly-calibrated scanning rig is inaccessible to most practitioners and researchers. When users want to train a policy on unique or specialized objects that fall outside the distribution of everyday objects in the Google Scanned Objects dataset or similar datasets, they still must manually recreate those objects in simulation. Additionally, the Google Scanned Objects dataset scanner setup requires a human-in-the-loop to manually insert objects. Therefore, it cannot be run at all times and requires non-negligible human effort to scale way beyond the 1,000 objects in the current dataset.

[36] and [37] developed methods to create 3D reconstructions of objects through manipulation for data collection using more common robot setups, but do not complete the pipeline of creating a fully simulatable asset with collision geometry and physical properties. Additionally, the hardware setups in these works require multiple robots, and can be simplified to be made even more accessible to users.

Our project tackles the goal of autonomously manipulating objects for 3D reconstruction with only a fixed 7-DOF arm. Without a mobile base or second arm, this setup has additional constraints on feasible grasps and display trajectories, but requires less hardware, making it more accessible to future users. Furthermore, we create a full pipeline from object manipulation to a ready to be used simulatable asset file.

Chapter 3

System Overview

3.1 Full Pipeline Outline

The investigation presented in this work is part of a larger project to construct a fully automatic pipeline to extract a physics simulatable asset from a standard pick and place setup [1]. For each object, the pipeline produces a visual geometry $\mathcal{V}$ (a textured visual mesh), collision geometry $\mathcal{C}$ (a union of convex meshes), and physical properties $\mathcal{P}$ (mass, center of mass, rotational inertia).

Objects are placed in the first bin, where the robot picks them up and reconstructs their visual and collision geometries by moving them in front of a static camera while re-grasping to reduce occlusions (Section 4.2 & 5.2 & 5.2.1). Next, the robot identifies the object's physical parameters by following a trajectory designed to be informative for the inertial parameters. Finally, it places the object into the second bin and repeats the process with the next object until the bin is empty. The extracted geometric and physical parameters are combined to generate a complete, simulatable object description. Figure 3.1 illustrates the asset generation workflow.

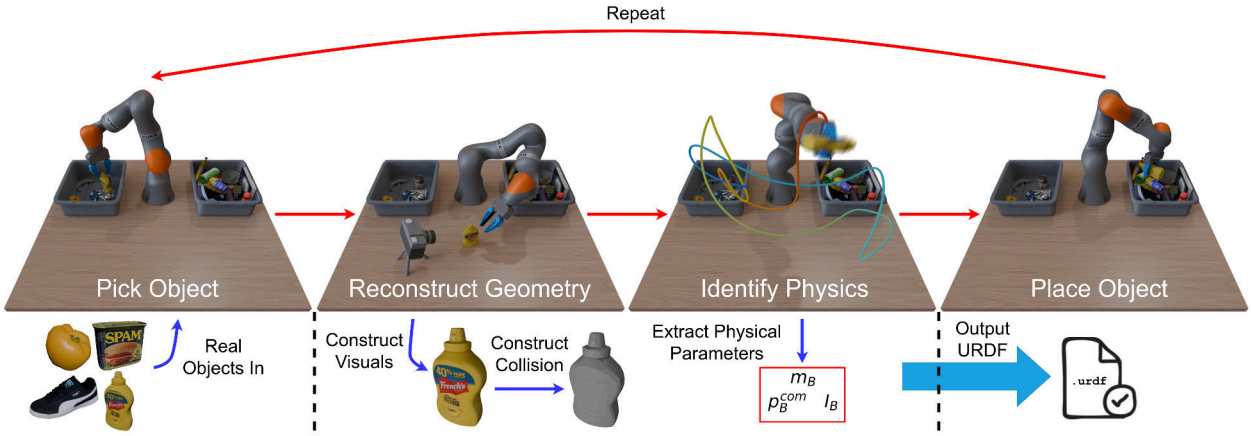


Figure 3.1: **An overview of the full automatic asset reconstruction pipeline**

This work focuses on the implementation of the pick and place setup and the extraction of visual textures and 3D collision geometries for generating simulatable assets; the first, second, and fourth stage of the pipeline in Figure 3.1.

Zooming into the second stage of the full pipeline diagram, which describes the data collection and 3D reconstruction for extracting visual and geometric properties of the object, a more detailed diagram of this step is shown in the figure below 3.2.

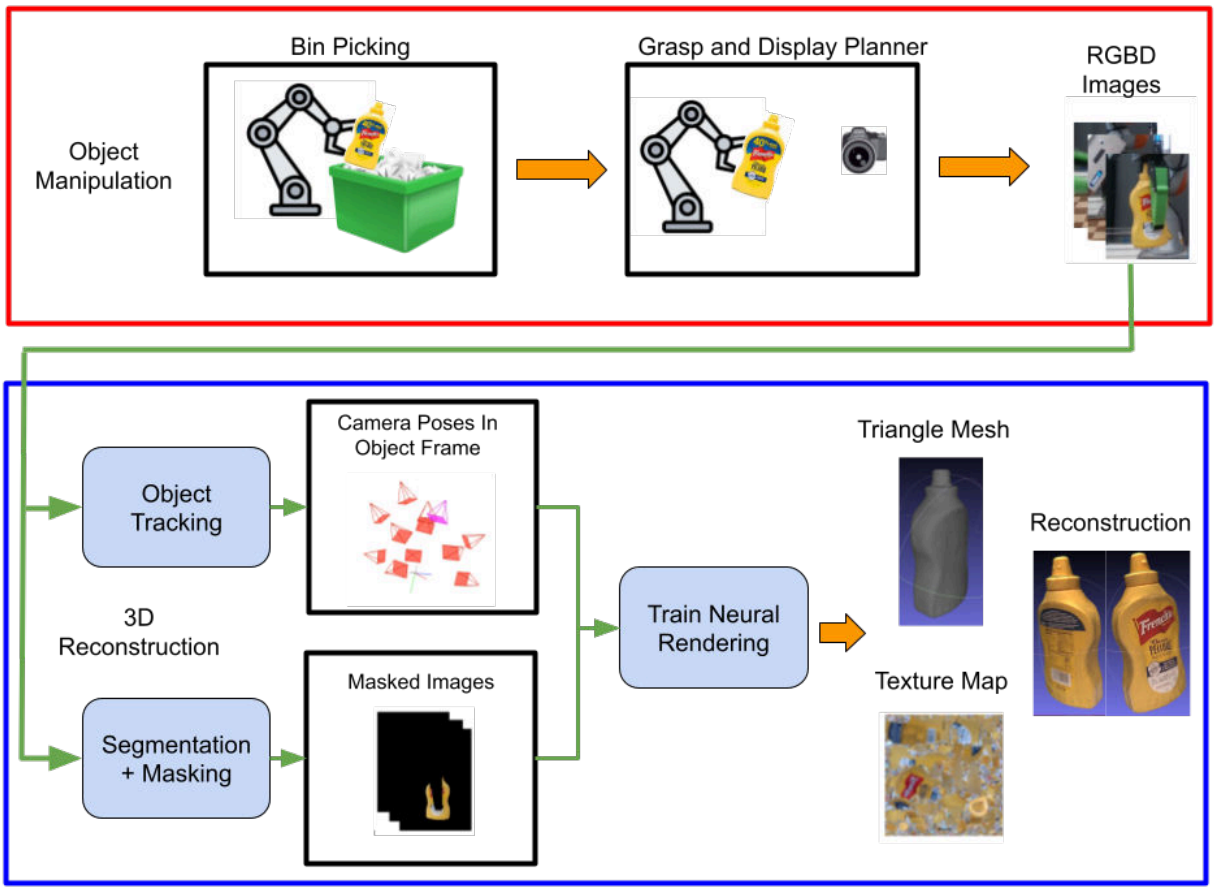


Figure 3.2: **A detailed diagram of our visual and geometric reconstruction pipeline**

For extracting the texture maps and collision geometries, our system collects RGBD images while the robot manipulates the object in front of the camera. Once the RGBD images of an object are collected, they are sent to be asynchronously processed for reconstruction. At this step, the pose of the object in the camera frame is determined using 6D object tracking (Section 5.1.2). Then, segmentation is used to produce masks of both the object of interest and the robot gripper. The combined masks and relative poses of the object to camera are both used as input into photometric reconstruction methods to train a 3D representation of the object. The texture map and collision geometry, represented as a triangle mesh, are then extracted out of the trained representation using existing methods.

3.2 Physical Setup

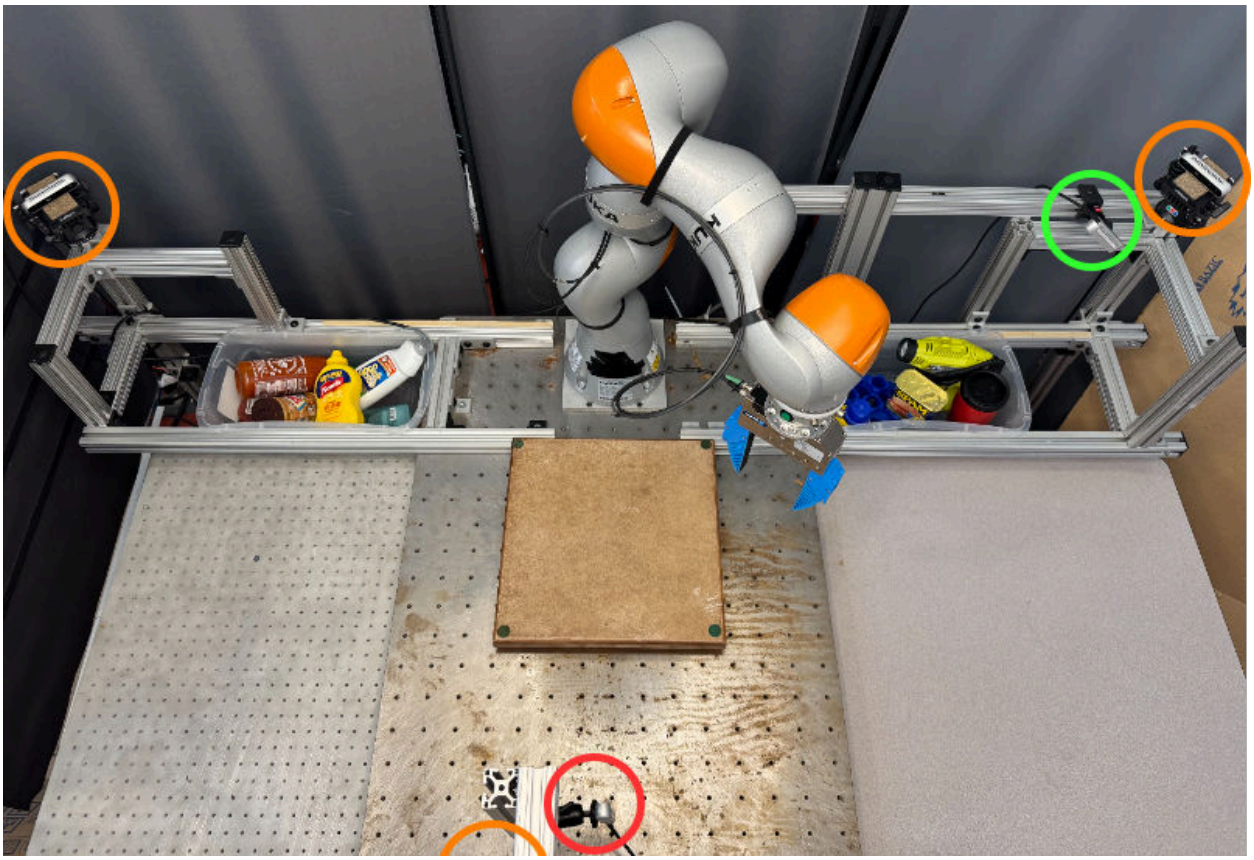


Figure 3.3: **Our pick-and-place setup.**

The physical pick and place setup we used for the implementation and testing of our system features a Kuka LBR iiwa 7 arm with a Schunk WSG-50 gripper and Toyota Research Institute Finray fingers. Workspace observations rely on three static RealSense D415 cameras (orange circles), while bin picking uses a RealSense D435 (green circle), and object scanning is performed with another D415 (red circle). All cameras capture 640×480 resolution RGBD images. Objects are picked from the right bin and placed into the left bin. A platform is used in the scanning workspace to enhance the iiwa's kinematic range during re-grasping.

It is important to note that the pipeline introduced in this project is extendable to any other pick and place setup. This method can be applied to different robot hardware, such as different arms, grippers, and motion planners for such, different positioning of pick and place bins, or different perception systems. One target application of this project would be to integrate it into an existing pick and place setup, such as sorting the contents of a starting bin into multiple ending bins, so that simulatable assets of the objects can be collected simultaneously to completing another task.

Chapter 4

Autonomous Pick and Display

In this section, we go over the physical autonomous pick and place pipeline, with focus on the grasp and display planner, which manipulates the object in front of the data collection camera with regrasps in order to observe the entire unoccluded surface of the object.

4.1 Background

4.1.1 Grasp Candidate Selection

We roughly follow the sampling method from ten Pas et al. [46] to generate grasp candidates, with modifications. The sampling method (given in Algorithm 2 of [46]), is as summarized below:

1. Sample N points from the point cloud
2. For each point, compute the orthogonal reference frame (Darboux frame, $\mathcal{D}(p)$) at that point. This is the frame such that the gripper's palm is pointing in the opposite direction of the outward-pointing unit surface normal at the point. Described in relation to the gripper, the x-axis of $\mathcal{D}(p)$ is the axis pointing away from the gripper's palm, which is parallel to the fingers of the gripper. The y-axis is the axis connecting the two parallel fingers, and the z-axis is the axis orthogonal to the two other axes.
3. For each sample reference frame, search a two-dimensional grid $G = Y \times \Phi$ where Y is a set of translations along the y-axis and a set Φ of additional rotations about the z-axis
4. For each $(y, \phi) \in G$, apply the corresponding translation and rotation to $\mathcal{D}(p)$ and further translate the grasp along the $-x$ direction until right before the gripper makes contact with the object. Check that this final grasp frame satisfies the following conditions:

(a) The gripper is not in collision with the object

(b) The closing region of the gripper contains at least one point from the object's point cloud

5. All grasp frames that pass the conditions above are added to a list of grasp candidates, which are considered for grasp selection.

In addition to this sampling algorithm, we make a few modifications in our implementation. In particular, we found that, in practice, searching along the remaining translations and rotations from the original Darboux frame (along the z axis and about the x and y axes) allowed us to find grasp candidates that were more stable or better aligned with our desired grasp properties. This approach also took less sampling time than simply increasing the number of points sampled from the point cloud.

4.1.2 Antipodal Grasping

We use *antipodal grasping* for determining stable grasps to pick up the object at each stage of the pipeline. Antipodal grasping involves a two-finger, parallel jaw gripper, where the fingers make contact at two points with collinear surface normals in the opposite direction.

We use the method provided in [47] to calculate the antipodal score of a grasp using the alignment of the surface normals within the gripper closing range to the axis connecting the fingers of the gripper:

Let:

- $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$ be the full set of point cloud points,
- $\mathcal{C} \subseteq \mathcal{P}$ be the subset of points within the gripper's closing region,
- $\mathbf{n}^i \in \mathbb{R}^3$ be the surface normal at point $\mathbf{p}_i \in \mathcal{C}$,
- $n^i_{G_y}$ be the component of $\mathbf{n}^i$ along the gripper's horizontal (closing) axis.

Then the antipodal cost is defined as:

$$\text{antipodal\_cost} = -\sum_{\mathbf{p}_i \in \mathcal{C}} \left(\mathbf{n}^i_{G_y}\right)^2 \tag{4.1}$$

We use this score along with other factors to grade sampled grasp candidates based on the desired properties of the grasp. Depending on the stage of the pipeline, the components of the scoring function change depending on the goal at that stage. This is described in more detail in 4.2. Since other scores are factored in for grasp selection, we modify the antipodal cost to control its weight among the different terms in the score. In particular, we exponentiate the per point terms in the antipodal cost to bias towards grasps where the points of contact are

more antipodal, rather than grasps with more enclosed points. Additionally, we normalize by the total number of points and scale the cost by a weight, $w_{\text{antipodal}}$, so the proportion of the antipodal cost in relation to the full grasp score is not dependent on the total size of the point cloud. Without normalization, it is natural that a larger object with more points would have more points in the grasping range, and therefore likely have a more substantial antipodal score.

$$\text{antipodal\_cost\_weighted} = -w_{\text{antipodal}} \cdot \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}_i \in \mathcal{C}} \exp\left(\left(n^i_{G_y}\right)^2\right) \tag{4.2}$$

Grasps are ranked in ascending order of score, with lower costs indicating better grasp quality.

4.2 Display Strategy

We aim to employ a display strategy with the following goals:

- Collect observations of all locations on the surface of the desired object
- Minimize time spent on data collection

We propose the following method. We grasp the object along two perpendicular axes and move the gripper in front of the viewing camera, then we rotate the object 360 deg along the grasp axis by rotating along the final joint (wrist) of the robot arm. This method minimizes the number of regrasps, which wastes data collection time since as the robot repositions itself for a regrasp, the object stays still, resulting in no new observations being collected in that time. Additionally, rotating the object along two perpendicular axes allows all surfaces of an object to be observed from the rays of a camera. This result is visualized in Figure 4.1. During the time that the robot is manipulating the object, we take RGBD images of the object from a stationary camera for data collection.

It is also important that we select stable grasps, in which objects are unlikely to slip out of the initial grasp position or fall out during the display trajectory. This is because we want to ensure the object stays within the working space of the robot to prevent unrecoverable failures, and that the object continues to be rotated along the planned axis for the method above to collect the desired data.

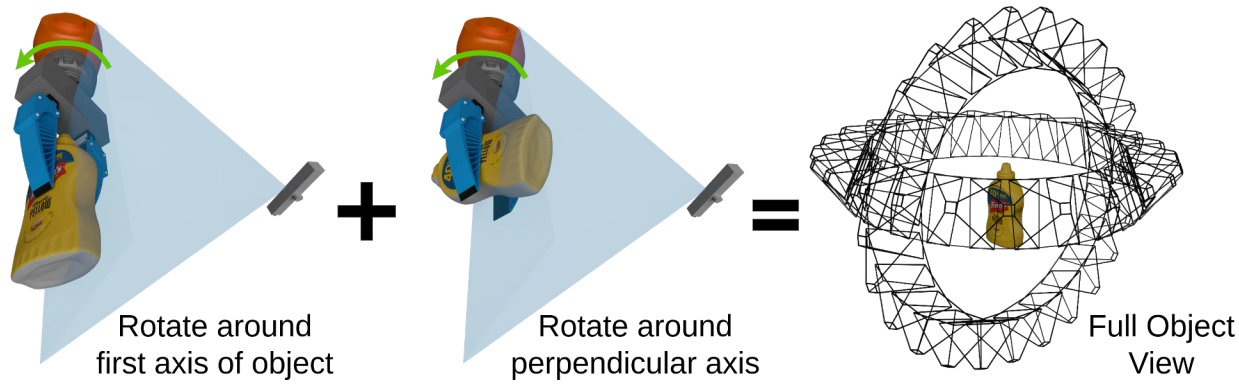


Figure 4.1: **Our object scanning method.** We use re-grasps to display the object along two perpendicular axes, providing the camera with a complete view of the object.

One immediate pitfall that can be noticed about this method is if the gripper occludes the same areas of the object between both the first and second rotation. If this happens, then the surface in those areas cannot be seen during the display trajectory. We employ a few methods to prevent this issue. The first is by selecting grasps that are farther apart and less likely to occlude the same surfaces, which is described in Section 4.3.2. The second is by keeping an uncertainty map of the surface voxels, and using additional regrasps if necessary based on the uncertainty map, covered in Section 4.3.3. Finally, we place the object down after executing the display trajectory facing the opposite direction from the camera as it had started. This ensures that two faces of the object are viewed completely unoccluded while the robot is not grasping the object and is reviewed in further detail in 4.5.2.

4.3 Display Grasp Selection

In this section, we go over the methods we investigated in order to implement the above described display strategy, such as how we find two perpendicular axes to rotate between and how we prevent occlusion of the same surfaces from multiple grasps. We first review one method we attempted for finding two perpendicular, non-overlapping grasps, that was not robust enough for our desired automated pipeline. Then, we review an updated method that was effective and an additional method to cover any gaps in collected observations.

4.3.1 Method 1 - Sequential Grasp Selection

Our first method involves using principal component analysis (PCA) to identify three orthogonal axes of the object's point cloud. PCA is a statistical method for identifying the orthogonal directions (principal components) along which data varies the most. The ith principal axis is the axis which best fits to the points in the set of data, while being orthogonal

to the first $i-1$ vectors, with the first principal axis being the axis which best aligns to the data without further constraints.

Given the object's point cloud, we use PCA to find the principal axes. We then align grasps to the first two of these axes, such that the gripper's fingers would close around the smallest axis. In this method, the first grasp is solved for by optimizing for alignment with the object's principal axis and the second grasp is solved for independently of the first. The first grasp is sampled from the point cloud of the object directly before the first grasp, and the second grasp is sampled from the point cloud of the object after the first grasp, display, and place has happened, directly before the second grasp.

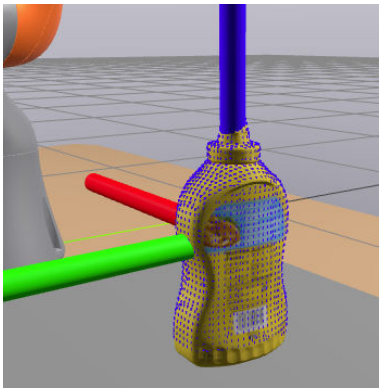


Figure 4.2: Principal components of the point cloud of a mustard bottle visualized in a Drake simulation. The blue, green, and red axes here correspond to the first, second, and third principal axis, respectively.

Given an axis to align the grasp to, we compute the axis alignment cost of a candidate grasp as follows, where $\mathbf{G_y}$ is the axis parallel to the gripper fingers and $\mathbf{a}_{\text{desired}}$ is the desired axis to align to

$$\text{axis\_alignment\_cost} = -w_{\text{alignment}} \cdot |\mathbf{G_y} \cdot \mathbf{a}_{\text{desired}}| \tag{4.3}$$

where the lower the cost, the better.

To ensure stability of the grasp, we additionally calculated the split ratios of the points at which grasp candidates are sampled. The split ratio measures how close a point is to the center of the bounding box, computed along the principal axes, and is a heuristic for center of mass for grasp stability. Grasping closer to the center of mass keeps torque low, which prevents the object from rotating out of the grasp, a common issue for parallel-yaw force-closure grasps. The split ratio is individually computed for each axis of the bounding box. A value of 1 means the point is exactly at the center along that axis, while a value of 0 means it is at one of the edges.

The split ratio at a point p for the ith axis is computed as:

$$\text{split\_ratio}_i = 1 - \left| \frac{2(p_i - c_i)}{\max(\{p_i^{(1)}, ..., p_i^{(n)}\}) - \min(\{p_i^{(1)}, ..., p_i^{(n)}\})} \right| \tag{4.4}$$

where:

- p_i: The coordinate of the point along the ith axis.
- $c_i = \frac{\max(\{p_i^{(1)}, ..., p_i^{(n)}\}) + \min(\{p_i^{(1)}, ..., p_i^{(n)}\})}{2}$: The center of the ith axis.
- $\{p_i^{(1)}, ..., p_i^{(n)}\}$: The set of coordinates along the ith axis for all n points in the point cloud

Then, we compute the split ratio cost as the negative sum of the split ratios along the most principal axis apart from the desired alignment axis:

$$\text{split\_ratio\_cost\_axis\_aligned} = -w_{\text{split ratio}} \cdot \sum_{i \neq a} \text{split\_ratio}_i \tag{4.5}$$

where a is the index of the principal axis we want to align the grasp to.

Finally, we also include the antipodal cost from Equation 4.2 for stability. In total, the axis-aligned grasps for this method are selected with the following cost function:

$$\begin{aligned} \text{sequential\_grasp\_cost} = \ & \text{axis\_alignment\_cost} \\ & + \ \text{split\_ratio\_cost\_axis\_aligned} \\ & + \ \text{antipodal\_cost\_weighted} \end{aligned} \tag{4.6}$$

This cost function is used to rank and select grasp candidates for the first and second grasps independently of and sequential to each other, therefore we refer to this method as "Sequential Grasp Selection".

Limitations

Although this method provides a simple formula for determining orthogonal grasps, there are shortcomings with this method in practice. Namely, if the first and second principal axes are in a direction such that all grasps that align are outside of the robot's joint limits, this method automatically fails. In particular, if one of these axes faces in the direction of the robot from the object and parallel to the table, we saw that there was often no joint configuration that could be found for any aligning grasps, since the object would likely be too close to the robot to grasp from the front, but too far to grasp from the back.

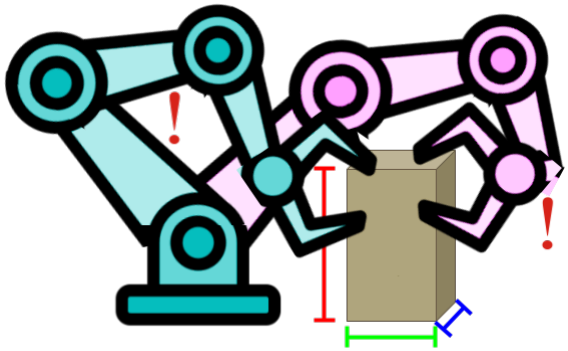
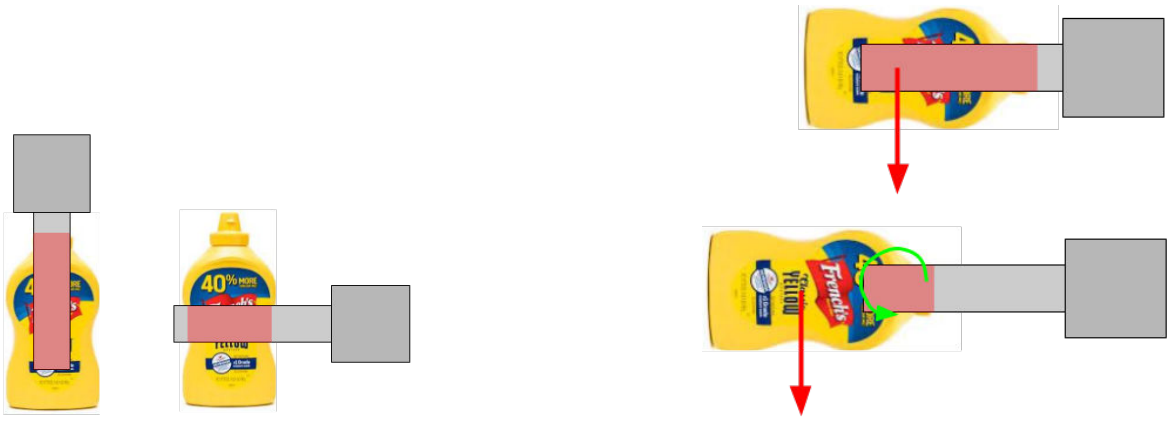


Figure 4.3: **A visualization of a case where all grasps which align to a principal axis are out of the robot's joint limits.** In this example, the red, green, and blue axes are the first, second and third principal axes, respectively. In order to grasp along the second axis, the robot must come either directly from the front or back, which goes out of its joint limits since it must fold up too tight in at least one joint, denoted by the red exclaimation marks. But, it would be possible to grasp this object from other axes. Therefore, we do not want to restrict our search.

One initial consideration for addressing the infeasibility of grasps aligning to one of the first two principal axes is to align to the third principal axis if either of the first two fail. Unfortunately, this method fails for objects that are particularly long across both the first and second axes, but short along a third, such as a cereal box. In this case, all feasible grasps must be such that the fingers are orthogonal to the third axis since the gripper cannot open wide enough to surround the item along any other axis.

Additionally, aligning one of the grasps to the longest axis of the object results in the gripper occluding much of the object in order to ensure a stable, centralized grasp along that long axis. This large occlusion makes it much more likely that the second grasp will have overlapping occluded regions, preventing complete observation of the object's surface.



(a) (b)

Figure 4.4: A visualization of issues with aligning grasps with the longest axis of the object. The image (a) shows how a close grasp occludes much of the object, as opposed to a close grasp along a shorter axis. Image (b) shows how a farther grasp becomes less stable and more vulnerable to slippage as the assumed center of mass moves out of the gripper's contact points. The red highlighted areas represent the surfaces occluded by the gripper.

4.3.2 Method 2 - Pair Grasp Selection

To address the limitations with Method 1, we introduce a second, more robust method. Now, we solve for both the first and second grasp jointly, such that they are optimized to be near perpendicular and translationally different, while both being stable grasps individually and kinematically feasible.

We first sample a set of stable grasp candidates and calculate their individual grasp scores, and choose a pair of stable grasp candidates out of this set. For the individual grasps, we want to ensure that all grasp candidates are sufficiently secure, since it is not guaranteed that the grasps with the highest scores will be in the highest-scoring pair, depending on how well they pair with other candidates. We review the components of the individual grasp score below.

Costs

Similarly to the single grasp selection method, the costs involved for scoring an individual grasp in this pair grasp selection method are the antipodal cost (Equation 4.2) and the split ratio cost. Here, we no longer consider aligning grasps to known perpendicular axes, so the axis alignment cost is not considered. Additionally, since we do not have a particular axis to align to, there is also no particular axis we particularly care to have a high split ratio for. Due to this, we modify the split ratio cost. Instead, we use the following formula

$$\text{split\_ratio\_cost} = -w_{\text{split ratio}} \cdot \Big(\text{split\_ratios}_{(1)} + \text{split\_ratios}_{(2)}\Big) \tag{4.7}$$

where $\text{split\_ratios}_{(i)}$ denotes the ith smallest value from the sorted set. We use the two largest split ratios amongst the three axes since the origin point of a grasp will always be at the surface of the object, which is likely to be at the far end of at least one axis for all grasps, so the lowest split ratio would provide little useful information.

Additionally, an interesting observation made during the implementation of this grasp scoring function was that using $|\mathcal{P}|$, the total number of points in the point cloud, versus $|\mathcal{C}|$, the number of points in the gripper closing range, as the the normalizing value in the weighted antipodal cost from Equation 4.2 affected the quality of the grasps. We found that a linear weighted combination of the two produced the best results and ranked intuitively stable grasp candidates higher. This is likely due to a tradeoff between making contact with more surface area that is not fully normal versus making contact with less surface area that is closer to normal to the contact point on the gripper fingers (higher quantity, decent quality versus lower quantity, high quality). Therefore, our antipodal cost was given by

$$\begin{aligned}&\text{antipodal\_cost\_hybrid} = \\ &-w_{\text{antipodal}} \cdot \left(w_{\text{all}} \cdot \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}_i \in \mathcal{C}} \exp\left(\left(n^i_{G_y}\right)^2\right) + \; w_{\text{closing}} \cdot \frac{1}{|\mathcal{C}|} \sum_{\mathbf{p}_i \in \mathcal{C}} \exp\left(\left(n^i_{G_y}\right)^2\right)\right)\end{aligned} \tag{4.8}$$

In total, the individual grasp cost is given by

$$\begin{aligned}\text{individual\_grasp\_cost} = \; &\text{split\_ratio\_cost} \\ &+ \; \text{antipodal\_cost\_hybrid}\end{aligned} \tag{4.9}$$

Conditions

We additionally filter out any grasps that do not satisfy the following conditions to ensure all grasps in plausible pairs are sufficiently secure:

Proportion Enclosed The proportion of all points in the gripper closing range is over some threshold τ_{enclosed}. This ensures that enough volume of the object is supported by the grasp, reducing the risk of slippage from an unbalanced center of mass.

$$\frac{|\mathcal{C}|}{|\mathcal{P}|} \geq \tau_{\text{enclosed}} \tag{4.10}$$

Proportion Enclosed Aligned We define the set of points within the closing range of the gripper $\mathcal{G} \subseteq \mathcal{C}$ as the set of points at which the surface normals are greater than some threshold τ_{normals} along the gripper's closing axis:

$$\mathcal{G} = \{\mathbf{p_i} \in \mathcal{C} : \mathbf{n}_{G_y}^i \geq \tau_{\text{normals}}\} \tag{4.11}$$

Then, we check that the proportion of all points which are in $\mathcal{G}$ is over some threshold $\tau_{\text{all\_aligned}}$ and the proportion of points in the gripper closing range which are in $\mathcal{G}$ is over some threshold $\tau_{\text{closing\_aligned}}$

$$\frac{|\mathcal{G}|}{|\mathcal{P}|} \geq \tau_{\text{all\_aligned}} \tag{4.12}$$

$$\frac{|\mathcal{G}|}{|\mathcal{C}|} \geq \tau_{\text{closing\_aligned}} \tag{4.13}$$

All grasps that pass these conditions are considered for grasp pair candidates.

Pair Scores

After we have our set of n stable grasp candidates, we compute the pair grasp scores of the $n(n-1)$ pairs. The pair scores are a combination of the two individual scores, plus a translational difference cost and a rotational difference cost.

The translational difference cost is proportional to the translational difference between the origin points of the grasps, which are the points in the point cloud that the Darboux frames from which the grasps were found were sampled from. Mathematically,

$$\text{translation\_cost}_{ij} = -\exp\left(\frac{\|\mathbf{p}_i - \mathbf{p}_j\|}{\|\mathbf{p}_{\max} - \mathbf{p}_{\min}\|}\right) \tag{4.14}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the origin points for the two grasps we want to find the pair grasp score for and $\mathbf{p}_{\max}$ and $\mathbf{p}_{\min}$ are the max and min coordinates in the point cloud. The denominator acts as a scaling factor to keep the translational cost invariant to the size of the object. Since we prefer a lower cost, this translation cost results in grasp pairs with farther origin points ranking higher.

Then, the rotational difference cost is found through the cosine difference of the x axis of the gripper, or the direction parallel to fingers ($\mathbf{x}_i^\top \mathbf{x}_j = \cos(\theta)$ where θ is the angle between the two x axes).

$$\text{rotation\_cost}_{ij} = w_{\text{rotation}} \cdot \frac{\exp\left(\left|\text{clip}\left(\mathbf{x}_i^\top \mathbf{x}_j, -1, 1\right)\right|\right) - 1}{\exp(1) - 1} \tag{4.15}$$

This gives a scalar value in $[0, 1]$ that measures how aligned the x axes of the two grasps are. The rotation cost is close to 1 if they point in the same or opposite directions and close to 0 if they are orthogonal. Since we prefer a lower cost, this results in grasp pairs which are orthogonal ranking higher.

In total, the cost for a pair of grasps is given by

$$\begin{aligned} \text{pair\_grasp\_cost}_{ij} = \ & \text{individual\_grasp\_cost}_i \\ & + \ \text{individual\_grasp\_cost}_j \\ & + \ \text{translation\_cost}_{ij} \\ & + \ \text{rotation\_cost}_{ij} \end{aligned} \tag{4.16}$$

The pair with the lowest cost and is kinematically feasible is chosen for pick and display.

Shifting Second Grasp

The location of the object may change between before and after the first grasp pick, display, and place. Since we calculate both the first and second grasp at once from the point cloud of the object before the first grasp, the second grasp may no longer align as desired to the object if the object has moved. We use ICP [48] to determine the relative translation between the point cloud of the object before and after the first grasp and apply this translation to the second grasp in order to shift it to the new location of the object.

4.3.3 Method 2.5 - Additional Uncertainty Based Regrasps

In order to certify and ensure completeness of the collected data of the object's surface, we develop an addition to the previous method. During data collection, we keep a lightweight map of our certainty of the object's surface. This map represents the object surface as a set of voxels each with an associated confidence value in $[0, 1]$, where voxels that have been completely unobserved have confidence 0 and voxels that have been fully observed have confidence 1. This method is loosely based off of the method presented in [49], which was also used in [36], and is simplified in our approach to reduce complexity given other priors we have.

Uncertainty Map Initialization

The map is initialized using the point cloud of the object taken before the first grasp, which is downsized into a set voxel size. This way, the points in the point cloud act as the voxels for the uncertainty map. The bottom surface of the object is filled in with voxels by filling in the shape traced by the bottommost points in the point cloud. Since the initial point

cloud measured is in an unoccluded space, we make the assumption that it is an accurate representation of the shape of the surface of the object. Then, for each voxel, we initialize the confidence value to 0. We also save the initial object pose in the world frame at this point, since the world frame is arbitrarily defined from the object tracking output, so this must be known to update the map once the object moves.

Map Update

As we collect images of the object, we extract the object pose and gripper masks from the images are use these and the data collection camera's pose to update the map. The object pose and gripper mask extraction is explained in further detail in Sections 5.1.2 and 5.2.2. Given 1) the object pose X_{WO_i} and 2) gripper mask M_G which specifies pixels in the image that belong to the robot gripper, and therefore may occlude the object, where the pose and gripper mask correspond to the same data frame, 3) the camera's stationary pose X_{WC}, and 4) original object pose W_{WO_0}, we make the following update to the uncertainty map.

Algorithm 1 Update Uncertainty Voxel Map with Observation

Require: New object pose $\mathbf{X}_{WO_i}$, original object pose $\mathbf{X}_{WO_0}$, camera pose $\mathbf{X}_{WC}$, occlusion mask $\mathbf{M}_G$, camera intrinsics K, image width in pixels W, image height inpixels H, hit radius threshold r

Ensure: Updated voxel confidences

$\mathbf{X}_{W'W} \leftarrow \mathbf{X}_{WO_0} \cdot \mathbf{X}_{WO_i}^{-1}$ *(Transform to new voxel map frame)*

$\mathbf{X}_{W'C} \leftarrow \mathbf{X}_{W'W} \cdot \mathbf{X}_{WC}$ *(Camera pose in new voxel map frame)*

$\mathbf{R} \leftarrow \text{CreateRays}(K, \mathbf{X}_{W'C}, W, H)$

$\mathbf{t}_{\text{hit}} \leftarrow \text{RaycastOnVoxelMap}(\mathbf{R})$

$\mathbf{m}_{\text{hit}} \leftarrow \text{IsFinite}(\mathbf{t}_{\text{hit}}) \wedge (\mathbf{M}_G == 0)$ *(Ray indices at which the object was hit and not occluded)*

$\mathbf{o}, \mathbf{d} \leftarrow \text{ExtractOriginsAndDirections}(\mathbf{R})$

$\mathbf{p}_{\text{hit}} \leftarrow \mathbf{o}[\mathbf{m}_{\text{hit}}] + \mathbf{t}_{\text{hit}}[\mathbf{m}_{\text{hit}}] \cdot \mathbf{d}[\mathbf{m}_{\text{hit}}]$ *(Hit points)*

MaxConfidenceImprovement $\leftarrow$ empty map

for all $(\mathbf{p}, \mathbf{r})$ in $(\mathbf{p}_{\text{hit}}, \mathbf{d}[\mathbf{m}_{\text{hit}}])$ **do**

  $\mathcal{N} \leftarrow \text{FindNeighborsWithinRadius}(\mathbf{p}, r)$

  if $\mathcal{N} = \emptyset$ **then**

    continue

  end if

  $\mathbf{n} \leftarrow \text{SurfaceNormals}[\mathcal{N}]$

  $\mathbf{w} \leftarrow \max(-\mathbf{n} \cdot \mathbf{r}, 0)$

  for all $(i, \text{voxel}) \in \text{enumerate}(\mathcal{N})$ **do**

    MaxConfidenceImprovement[voxel] $\leftarrow \max($*MaxConfidenceImprovement.get(voxel)*$, \mathbf{w}[i])$

  end for

end for

for all $(\text{voxel}, w) \in$ *MaxConfidenceImprovement* **do**

  Confidences[voxel] $+= w$

end for

Confidences $\leftarrow \text{Clip}(\text{Confidences}, 0, 1)$

The algorithm above does the following. First, since the object has moved, the camera is in a new location in relation to the voxel map. Therefore, we want to get the camera's pose in the voxel map's frame. To do this, we get the transformation from the initial voxel map frame to the current voxel map frame. Then, we get the pose of the camera in the current voxel map frame from the camera pose in the world frame (same as the initial voxel map frame, since it was initialized from points in the world frame) and the previously found transformation. Then, we perform ray casting on the voxel grid using the camera's pose in the new voxel map frame and known intrinsics. We note all rays that hit the grid at a finite distance. For rays extending from any pixels that are in the gripper mask, we treat the pixel

as occluded and do not count it as hit. Then, for each ray that hits, we calculate the location on the voxel map it hits using the origin, direction, and distance traveled of the ray. Then, we calculate the potential confidence improvement from this ray's view of all voxels that are within a radius r of the hit point. We define the knowledge gain as the negative dot product of the surface normal at the hit point and the ray's direction, meaning that a more direct view of the surface (surface normal and ray direction are pointing towards each other) adds more confidence. We maximize with 0 to avoid decreasing confidence due to inaccuracies or edge cases with object normals. Finally, the confidences for each voxel are updated by adding the best potential confidence increase over all rays that hit the voxel, and we clip the result between 0 and 1.

This algorithm builds a record of surfaces that have been observed and how directly they have been observed. This map can be used as a certificate in determining if the object has been fully observed, and can also be used to select additional regrasps to optimally display unseen surfaces.

Regrasp Selection

After the object is displayed using grasps determined with Method 2, we can continue to regrasp and display the object if we determine that the surface has not been sufficiently observed, using the updated uncertainty map. We define a threshold for confidence $\tau_{\text{confident}}$ for a voxel to be considered sufficiently observed and a threshold for the fraction of all voxels in the map that are sufficiently observed τ_{full} for the whole object surface to be considered sufficiently observed. We check if

$$\text{percent\_observed} = \frac{\sum (\mathbf{c}_i > \tau_c)}{|\mathbf{C}|} \geq \tau_o \tag{4.17}$$

to determine if the object has been fully observed after the first two grasps, where $\mathbf{C} = \{\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_N\}$ is the set of all confidences of the voxels in the map.

Given the current state of the uncertainty map, we use the following method to determine the next grasp. The object will be grasped from this position and then displayed in front of the camera using a 360 degree wrist rotation as for the first two grasps. Then, from the sampled grasp candidates, we can estimate the observations that would be made. Then, we can compute the expected gain in confidence in the uncertainty map.

When computing a grasp candidate, we know the points of the point cloud that are within the gripper's closing range. We assume that these points will be occluded by the gripper fingers during the display trajectory.

We first use the candidate grasp pose, the initial object pose, and the planned gripper poses for the display trajectory to calculate 8 object poses spaced evenly through the 360 degree display rotation, given by Algorithm 2 where $N = 8$.

Algorithm 2 Generate Camera Poses In Map Frame During Display Trajectory

Require: Grasp candidate pose $\mathbf{X}_{WG}$, observation camera pose $\mathbf{X}_{WC}$

Ensure: Sequence of camera poses in the voxel map frame: $\{\mathbf{X}^i_{W'C}\}_{i=1}^N$

1: $\{\mathbf{X}^i_{WG'}\}_{i=1}^N \leftarrow$ GetYawDisplayTrajectory() *Sample N gripper poses from display traj*

2: $\{\mathbf{X}^i_{WW'}\}_{i=1}^N \leftarrow \{\mathbf{X}^i_{WG} \cdot \mathbf{X}^{-1}_{WG'}\}$ *New uncertainty map frame in old*

3: $\{\mathbf{X}^i_{W'C}\}_{i=1}^N \leftarrow \{\mathbf{X}^{-1}_{WW'} \cdot \mathbf{X}^i_{WC}\}$ *Camera poses in voxel map frame*

4: **return** $\{\mathbf{X}^i_{W'C}\}_{i=1}^N$

Then, we use the same logic from 1 to calculate the confidence values in the map after collecting observations from these views. Now, instead of masking certain rays with the gripper mask, we use the set of points in the gripper closing region and do not update the confidences of these presumably occluded points. We do not update the confidence values in the map, but note the change in percent_observed from Equation 4.17. The confidence improvement cost of this grasp is then

$$\text{confidence\_improvement\_cost} = -w_{\text{conf\_improve}} \cdot \Delta\text{percent\_observed} \tag{4.18}$$

The final grasp cost for the additional grasp is then a linear combination of the individual grasp cost from Method 2:

$$\begin{aligned} \text{additional\_grasp\_cost} = \ & \text{individual\_grasp\_cost} \\ & + \ \text{confidence\_improvement\_cost} \end{aligned} \tag{4.19}$$

The grasp with the lowest additional_grasp_cost that is kinematically feasible is selected for the next pick and display. This process is repeated until the percent_observed of the uncertainty map crosses the desired threshold.

We found that Method 2 is successful in collecting complete visual data of the object surface in most instances, with the additions in Method 2.5 further increasing the robustness of our strategy. Further elaboration is given in Chapter 6.

4.4 Other Strategies Considered

We investigated a few other strategies for data collection, which were not chosen for the final pipeline. We will briefly cover these strategies here.

Turntable

First, we considered installing a turn table in the setup to place the object on. Then, either a motor or the robot would turn the table to display the sides of the object before the robot

picks up the object once horizontally in order to display the top and bottom of the object. This method was not chosen since the extra hardware would go against our goal of having a simple, accessible setup. But, this method was still implemented and tested in simulation as a possible benchmark collection method, since this method would be guaranteed to produce unoccluded views of all surfaces.

Wrist Camera + Front Camera

Another method that was investigated was attaching a separate data collection camera to the wrist of the robot arm. Then, the robot moves its end effector around the object to collect views of the sides and top. Finally, the robot picks up thee object once horizontally in order to display the top and bottom of the object to the front camera. This method was implemented and tested in simulation, but we found that it was difficult, although not impossible, to reach the wrist camera around the object unless the object was precisely placed in a feasible location. Additionally, when tracking the object between the video from the two cameras, the same origin frame for the object would have to be chosen, resulting for more calibration necessary and more room for error.

4.5 Pick and Place Pipeline Construction

In this section, we briefly go over some details about our autonomous pick and place pipeline, including grasp selection at other stages in the pipeline. The complete codebase for the pick and place setup is available at https://github.com/evelyn-fu/pickplace_data_collection.

4.5.1 Grasp Selection For Other Pipeline Stages

In this section, we review the implementation details for grasp selection at the bin picking, physical parameter identification, and placing steps of the autonomous robotic pipeline. At each stage, we generate grasp candidates following the method explained in 4.1.1, and score them based on different cost functions based on the requirements at each stage.

Bin Picking Grasps

Due to the angle of our camera in our particular setup, the point cloud measurements of individual objects are often incomplete. Specifically, we can mostly only see the top and one side of the objects, and many object sides are occluded by the other objects in the highly cluttered bin. Because of this, the antipodal score if often misled in our implementation. Therefore, we do not use the antipodal cost for ranking bin picking grasps.

Instead, we use two alignment scores similar to that in Method 1: The first to align the gripper x axis to the vertical world axis, and the second to align the gripper z axis to the

first principal axis of the 2D flattened point cloud of the desired object. The first alignment cost is self explanatory, since we are picking into a cluttered bin and can only guarantee a good point cloud measurement of the tops of the object, so we want to grasp straight down.

The second alignment cost requires us to separate the point cloud between the separate objects in the bin so we can find the principal axes individually between axes. We cluster nearby points, and for each cluster, we translate all the points in the cluster up to the highest z value of points in the cluster and downsize to the desired voxel size again to "flatten" it to the xy plane. This way, we can find the clusters' principal axes along the xy world plane. Then, grasps are sampled from the flattened point cloud, and the second alignment cost measures the alignment between the gripper z axis and the first principal axis, since we want the fingers to close on the wider surface of the object.

Physical Parameter Identification Grasps

More information about the physical parameter identification step of the pipeline can be found in our paper, [1], but it is not the focus of this thesis, which covers the identifying the visual and geometric properties and the construction of the physical pipeline. Since the physical parameter identification step is a part of the physical pipeline, it is worth mentioning here. The physical parameter identification involves grasping the object and moving it through an excitation trajectory, which may involve high accelerations and requires a very secure grasp on the object.

We use the same hybrid antipodal cost as in 4.8 and the same filtering conditions as in Method 2 to select stable grasps. Additionally, once a grasp is selected, we shift it along the gripper's y axis so that the two fingers are equally distant from the nearest surface on the point cloud. This ensures that when the gripper closes, the fingers do not tilt the object in one direction or another, resulting in a more secure grasp.

Bin Placing Grasps

We use the same stable grasp scoring method as for the physical parameter identification grasps.

4.5.2 Miscellaneous

Motion Planning

We use [50] for motion planning. We found that this motion planner integrated well into our pipeline due to its fast planning times and minimal planning failure rate, which allowed us to keep the data collection time in our pipeline low.

Place Flipped

At the end of the second grasp's pick and display, we place the object down rotated 180 degrees along the vertical axis from its original orientation, but generally in the same translational position. This way, the camera can get an unoccluded view of both the front and back sides of the object with minimal extra movement. Given the original pick grasp pose, X_{WG}, we calculate the new place grasp pose

Algorithm 3 Compute 180° Z-axis Flipped Gripper Pose

Require: Gripper pose X_{WG}, gripper length l_g

1: Compute gripper center in world frame:

$$\mathbf{t}_{\text{center}} = X_{WG} \cdot \begin{bmatrix} 0 \\ 0 \\ \frac{3}{4} l_g \end{bmatrix}$$

2: Define 180° rotation around the z-axis:

$$R_z^{180} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3: Extract current rotation and translation:

$$R_{\text{curr}} = X_{WG}[0:3, 0:3], \quad \mathbf{t}_{\text{curr}} = X_{WG}[0:3, 3]$$

4: Rotate translation around $\mathbf{t}_{\text{center}}$:

$$\mathbf{t}_{\text{final}} = R_z^{180} \cdot (\mathbf{t}_{\text{curr}} - \mathbf{t}_{\text{center}}) + \mathbf{t}_{\text{center}}$$

5: Compute final rotation:

$$R_{\text{final}} = R_z^{180} \cdot R_{\text{curr}}$$

6: Construct flipped gripper pose:

$$X_G^{\text{place}} = \begin{bmatrix} \mathbf{R}_{\text{final}} & \mathbf{t}_{\text{final}} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

In the above algorithm, we estimate the center of the object to be grasped near 3/4 of the distance from the end effector to the tips of the finger. This estimate was chosen for our particular hardware setup. We get the position of this estimated center $\mathbf{t}_{\text{center}}$ and rotate the pick pose by 180 degrees about the z-axis, centered around $\mathbf{t}_{\text{center}}$, to calculate the gripper pose for flipping the object to the other side. Note that this generally works better for more

vertical grasps, since mirroring a horizontal grasp would likely be far in joint space, and this extra motion could cause the object to slip in the grasp during the execution of the trajectory. Therefore, we modify Method 2 to use the more vertical grasp of the pair as the second

Chapter 5

3D Reconstruction of Assets

In this section, we go over our method for reconstructing the 3D visual and collision geometries of the objects after we have collected data with the methods in Chapter 4. A full implementation of this section can be found at https://github.com/nepfaff/scalable-real2sim.

5.1 Background

5.1.1 Mathematical Background on SOTA Implicit 3D Reconstruction Methods

We first briefly review the framework behind NeRF, a current state-of-the-art 3D reconstruction methods which many other photometric reconstruction methods build off of. This will set us up for presenting our recipe for modifying these methods and their numerous adaptations to object-centric applications with dynamic occlusions, such as our pick and place setup.

NeRFs

NeRFs are trained from images of a static scene where the camera pose within the scene is known for each image. Given the camera pose, an image of the scene taken from the camera pose can be rendered from the NeRF by marching along a ray $\mathbf{r} : r(t) = \mathbf{x_o} + t\mathbf{d}$

$$\mathbf{I}_\theta(\mathbf{x_o}, \mathbf{d}) = \int_{t_n}^{t_f} w(t)\mathbf{c}_\theta(\mathbf{r}(t), \mathbf{d})dt \tag{5.1}$$

where

$$w(\mathbf{x_o}, \mathbf{d}, t) = \exp\left[-\int_{t_n}^{t_f} \sigma_\theta(\mathbf{r}(s))ds\right]\sigma_\theta(\mathbf{r}(t)) \tag{5.2}$$

is the likelihood of the ray terminating at t.

Then, loss is the total squared error between the rendered and true pixel colors in the given batch of images and sampled rays.

$$\mathcal{L} = \mathop{\mathbb{E}}_{\mathbf{I}\in\mathcal{D},\mathbf{r}\in\mathcal{R}(\mathbf{I})} ||\mathbf{I}(\mathbf{x_o}, \mathbf{d}) - \mathbf{I}_\theta(\mathbf{x_o}, \mathbf{d})||_2^2 \tag{5.3}$$

5.1.2 6D Pose Estimation

6D pose estimation is an actively developing field of research pertaining to identifying the 6 dimensional pose, $(x, y, z, \phi, \theta, \psi)$. More specifically, 6D pose tracking is the task of using temporal cues to estimate the pose of an object throughout the frames of a video. Our reconstruction method requires us to extract the object's pose out of the video data we collect, so we review some current SOTA methods for 6D pose estimation and tracking here.

Recent approaches for 6D object tracking include BundleSDF [27], which simultaneously learns a Neural Object Field and leverages the online learned 3D representation to assist in pose graph optimization. In our experiments, we found BundleSDF to be reliable for estimating accurate object poses and producing 3D reconstructions. One limitation is that BundleSDF only runs at near real-time rate. FoundationPose [51] provides a real-time tracking method, but requires either a predefined 3D model of the object or a set of images forming a complete view of the object. Finally, a recent work, Any6D [52] provides a model-free 6D tracking method that works in real time, but the implementation has not yet been released at the time of this writing. We use BundleSDF for object tracking in our 3D reconstruction recipe.

5.1.3 Object Segmentation

Object segmentation is the process of classifying each pixel in an image into a specific class or object. For our method, it is important to identify which pixels belong to the object and the robot gripper. We use Segment Anything 2 (SAM2) [53], a foundation model for promptable object segmentation in both images in videos. SAM2 can be prompted for video data by providing points within an object of image, a bounding box, or a mask on one frame of a video. Then, it will produce a mask for the object of interest on the given frame and continue to propagate through the video an mask the object of interest as it moves through the video. Therefore, we use SAM2 in our implementation for segmenting both the object we want to reconstruct and the robot gripper.

5.2 Geometric Reconstruction for Visual Geometry

We obtain the object's visual mesh and texture map using SOTA implicit reconstruction methods. These methods are typically designed to reconstruct an entire static scene rather

than dynamic, object-centric scenarios [20]. In this section, we propose a general recipe for adapting these reconstruction methods for object-centric reconstructions from our dynamic scenes where the object of interest may be partially occluded. We demonstrate this recipe on three reconstruction approaches, Nerfacto from Nerfstudio [32], Gaussian Frosting [54], and Neuralangelo [55]. This recipe applies broadly to any method relying on photometric losses.

Standard 3D reconstruction methods like NeRF [25] assume a static scene and moving camera, and require the pose of the camera in the frame of the scene for each sample image in the training data. In our application, the scene we want to reconstruct is the moving object. Therefore, we redefine the world frame for reconstruction to a frame centered on the object. Then, to get the pose of the camera in the object frame, we track the object's pose in the world, X_{WO} through the video frames. Then, knowing the camera's stationary pose in the world, W_{XC}, we can get the pose of the camera in the object frame, $X_{OC} = (X_{WO})^{-1}X_{WC}$.

Additionally, since we only want to reconstruct the object and not any of the background (which is moving in the object frame and therefore not constant), we use object masks $\{\mathcal{M}^O\}^N$ to only train on pixels that belong to the object. These object masks are retrieved using video segmentation from SAM2 [53]. Either a bounding box of the object, generated using GroundingDINO [56] given a text prompt, or an initial point or set of points within the object on the first video frame, selected through our custom made GUI, is used as a prompt to SAM2, which propagates throughout the video to get object masks in all frames.

However, excluding background pixels can lead to density bleeding (see Figure 5.1), where the model assigns nonzero density to empty space due to a lack of supervision. To address this, we employ *alpha-transparent training* [57], which enforces zero density in the background without additional hyperparameters.

This method replaces background pixels in the training data with iteration-dependent random colors and blends those same colors into the predicted image based on the model's density predictions. Since the model cannot predict these random colors, it minimizes the loss by assigning zero density outside the object, allowing the colors to shine through.

For each training iteration, a random background color $\mathbf{C}_{\text{bg}} \sim \mathcal{U}(0,1)^3$ is sampled uniformly and used to composite both the ground-truth and predicted images.

Let a sampled camera ray (as explained in Section 5.1.1) be defined as $\boldsymbol{r} = (\boldsymbol{x_o}, \boldsymbol{d})$, where $\boldsymbol{x_o}$ is the camera origin and $\boldsymbol{d}$ is the direction. Let:

- $\boldsymbol{I}_{\text{rgb}}(\boldsymbol{x_o}, \boldsymbol{d}) \in [0,1]^3$ denote the ground-truth RGB colors along ray $\boldsymbol{r}$,
- $\alpha(\boldsymbol{x_o}, \boldsymbol{d}) \in [0,1]$ be the ground-truth opacities along the ray,
- $\boldsymbol{I}_\theta^{\text{rgb}}(\boldsymbol{x_o}, \boldsymbol{d}) \in [0,1]^3$ be the model-predicted RGB colors,
- $\alpha_\theta(\boldsymbol{x_o}, \boldsymbol{d}) \in [0,1]$ be the model-predicted opacities.

Then, the composited ground-truth and predicted pixel colors used for training are:

$$\boldsymbol{I}(\boldsymbol{x_o}, \boldsymbol{d}) = \alpha(\boldsymbol{x_o}, \boldsymbol{d}) \cdot \boldsymbol{I}_{\text{rgb}}(\boldsymbol{x_o}, \boldsymbol{d}) + (1 - \alpha(\boldsymbol{x_o}, \boldsymbol{d})) \cdot \mathbf{C}_{\text{bg}} \tag{5.4}$$

$$\boldsymbol{I}_\theta(\boldsymbol{x_o}, \boldsymbol{d}) = \alpha_\theta(\boldsymbol{x_o}, \boldsymbol{d}) \cdot \boldsymbol{I}_\theta^{\text{rgb}}(\boldsymbol{x_o}, \boldsymbol{d}) + (1 - \alpha_\theta(\boldsymbol{x_o}, \boldsymbol{d})) \cdot \mathbf{C}_{\text{bg}} \tag{5.5}$$

The loss is then defined as in Equation 5.3. Then, for locations along the rays where the representation should be transparent, $\alpha_\theta(\boldsymbol{x_o}, \boldsymbol{d})$ is driven towards 0 so that the random color matches between the blended predicted and blended ground truth images. We found that this method of addressing density bleeding is reliable, and unlike directly putting a loss on the transparency for pixels not in the object mask, we do not need to tune any hyperparameters for this method.

Alpha-transparent training resolves background issues but cannot handle occlusions, such as those caused by the gripper, as it would incorrectly make occluded regions transparent. This could result in regions on the surface of the object to have holes in the reconstruction. To address this, we use gripper masks $\{\mathcal{M}^G\}^N$ to exclude occluded pixels from the reconstruction objective. More information about how we acquire these gripper masks automatically is given in Section 5.2.2. Gripper masks take precedence over object masks.

For featureless objects like single-color surfaces, depth supervision, which constrains reconstruction with depth data, improves accuracy by resolving ambiguities in photometric losses. This additional geometric constraint is particularly effective for objects such as bowls, where photometric methods often struggle.

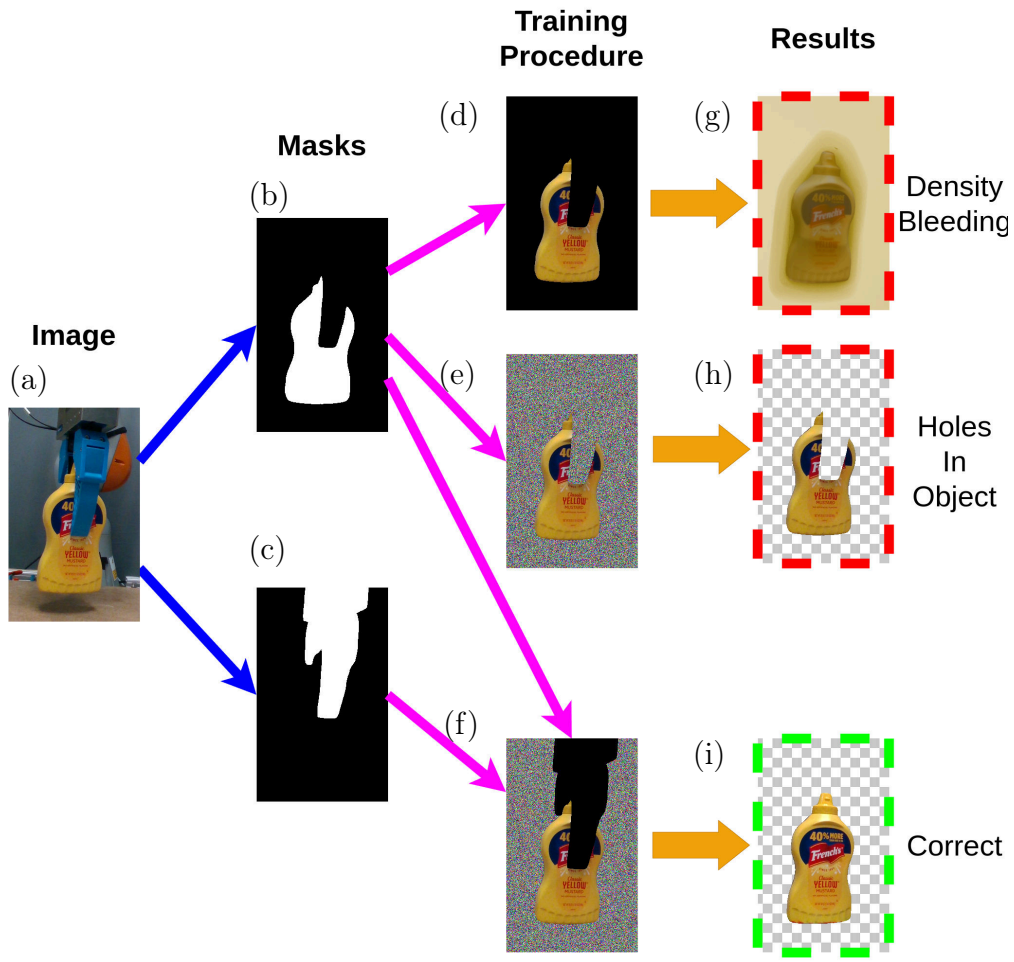


Figure 5.1: **Our object-centric visual reconstruction recipe.** From the collected RGB images (a), we obtain the object masks (b) and gripper masks (c). Using only the object masks to ignore background pixels during training (d) results in density bleeding into unoccupied regions (g). Applying alpha-transparent training (e) mitigates density bleeding but incorrectly drives occluded object regions toward transparency (h). Ignoring pixels inside of the gripper mask during training, along with employing alpha transparent training (f), successfully reconstructs an unoccluded object view with no density bleeding (i).

5.2.1 Collision Geometries

The visual geometry $\mathcal{V}$ is simplified into a convex collision geometry $\mathcal{C}$ for physics simulation. Following prior works [19, 20], we use approximate convex decomposition algorithms [58, 59], which split $\mathcal{V}$ into nearly convex components and simplify each using convex hulls. This process yields a computationally efficient and simulatable geometry $\mathcal{C}$.

We note that simulating a collection of convex pieces can be suboptimal, especially when meshes overlap or contain gaps. In Drake's hydroelastic contact model [60], gaps can distort the pressure field, causing dynamic artifacts. In the point contact models used in most robotics simulators, overlaps may generate interior contact points. Another option is to

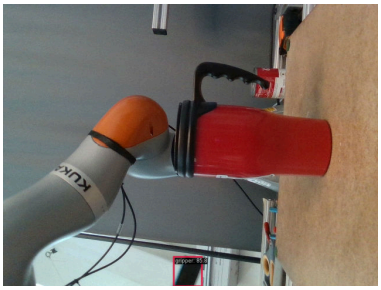
use primitive geometries to represent the collision geometry. For instance, sphere-based approximations might enable rapid simulations on GPU-accelerated simulators. The optimal representation depends on the simulator and the tradeoff between simulation speed and accuracy. While we found convex decomposition effective in most cases, particularly with point contact models, a more general approach remains an avenue for future work.

5.2.2 Gripper Segmentation

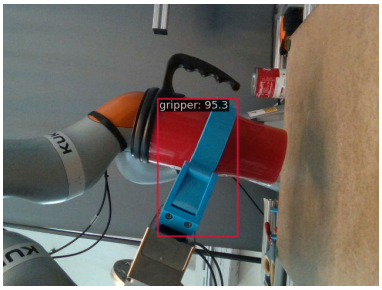
In this section, we explain extra steps we took to automatically retrieve gripper masks. Since the robot gripper is not a common object, it is not in the distribution of the foundation models we are using for object detection and segmentation. Therefore, we found that manually prompting SAM2 [53] with points within the gripper on the initial video frame was not sufficient in multiple instances. The mask was lost multiple times, requiring human intervention to reprompt. This manual step is tedious, and strays us away from our goal of having a fully autonomous pipeline. Additionally, prompting GroundingDINO [56] with a text prompt to automatically generate a bounding box to input into SAM2 also fails since it often cannot recognize the gripper either.

Therefore, we use all successfully segmented gripper masks using the default SAM2 model as training data to fine tune both SAM2 and GroundingDINO for recognizing the gripper object. We use the training script provided in SAM2's repo and mmdetection's implementation of GroundingDINO training for fine tuning SAM2 and GroundingDINO, respectively [53][56][61].

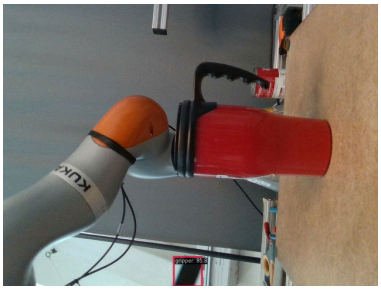
The results can be seen in Figures 5.2 and 5.3.



(a) Successful object detection.

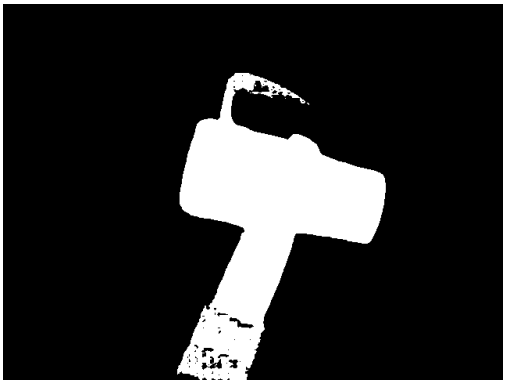


(b) Another successful object detection.

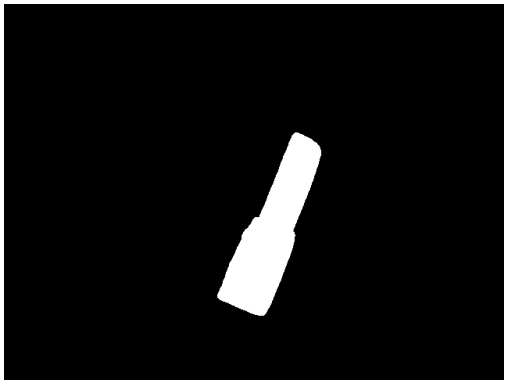


(c) Successful object detection on the very first frame of gripper entering the camera view.

Figure 5.2: **Gripper object detection using a a fine tuned GroundingDINO model.** The pipeline can now automatically detect the gripper by using the text prompt "gripper" and use this bounding box as an input for segmentation. Previously, using the text prompt "gripper" would often not detect the gripper in the image, producing 0 bounding boxes.



(a) Failed segmentation using default Grounding DINO + SAM2 models.



(b) Successful segmentation using fine tuned Grounding DINO + SAM2 models

Figure 5.3: **Gripper segmentation mask on the same video frame using the default SAM2 and GroundingDINO models vs using custom fine tuned versions.** These masks are produced from the same frame of the same video.

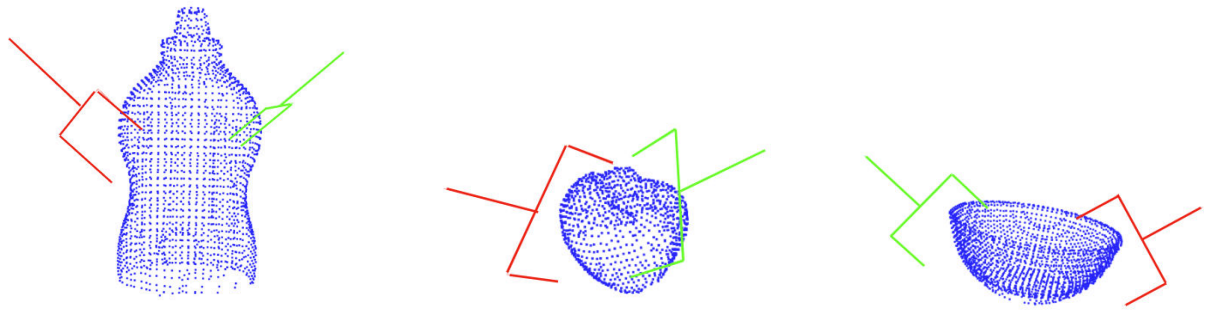
Chapter 6

Results and Evaluation

In this section, we go over the results of the methods previously presented, including examples of selected grasp pairs and a visualization of the confidence map. We also provide some examples of fully simulatable assets we reconstructed from real objects using our pipeline.

6.1 Grasp Selection and Confidence

We provide some examples of grasps generated from the method in Section 4.3.2 in Figure 6.1. These grasps are generated for point clouds of 3 objects, a mustard bottle, bell pepper, and a bowl. The result on the bell pepper is interesting because of how small the object is, but the grasps selected in the pair are still very far apart both translationally and rotationally, while being orthogonal to each other. The result on the bowl is also interesting because Method 1 would tend away from grasping on the lip of the bowl, since it does not align to any principal axis of the bowl, but intuitively, we know that it is the most stable location to grasp. Here, using Method 2, our grasp selection process is able to easily find two stable grasps on the bowl's lip which are orthogonal to each other.



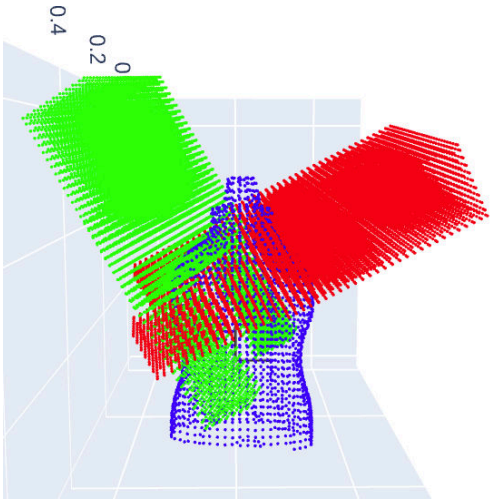
(a) Grasps selected for mustard bottle

(b) Grasps selected for bell pepper

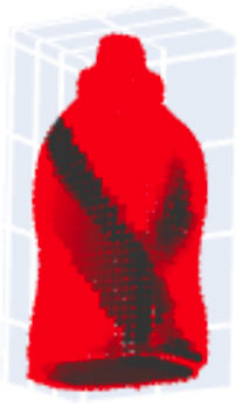
(c) Grasps selected for bowl

Figure 6.1: Grasps selected jointly using the method from Section 4.3.2

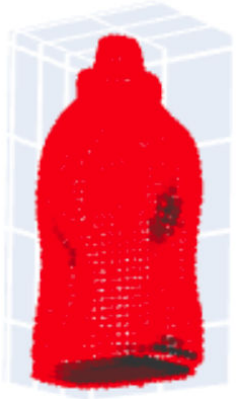
Figure 6.2, illustrates a scenario where the uncertainty-based regrasp method from Section 4.3.3 is applied. This scenario was created and tested in simulation for easy access to ground truth object masks and serves as a proof of concept. In this scenario, the two pair grasps selected are along the diagonal of the mustard bottle and perpendicular to each other, with the first grasp (in red) being slightly more horizontal. The bottom of the bottle should be displayed during the first grasp, but due to slippage, the gripper ends up resting more vertically along the bottle as it rotates. This can be seen in 6.2b, where the gray low confidence region marks where the gripper had occluded the bottle, which is at a steeper angle than originally planned. Therefore, after the two pair grasps, the bottom of the object is still at low confidence 6.2c. We solve for another regrasp to maximize new information gain from the display trajectory and select a very horizontal grasp 6.2d to collect data from the bottom of the object which we missed before. After this regrasp and display, we have now gained information about the bottom of the object 6.2e. The difference in the reconstruction result before and after including the observations from the additional grasp is shown in Figure 6.3.



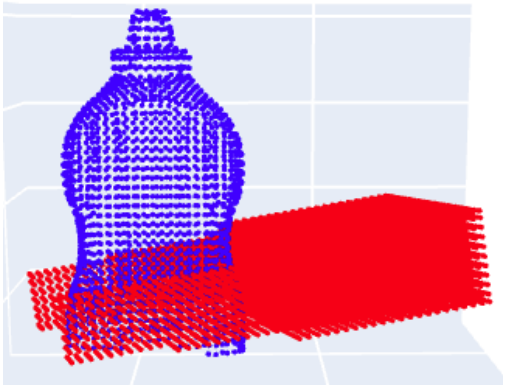
(a) Pair grasps selected for mustard bottle using method from Section 4.3.2



(b) The uncertainty map after the first grasp. Note how the surfaces occluded by the grasp are less confident. The bottom of the bottle also has low confidence.



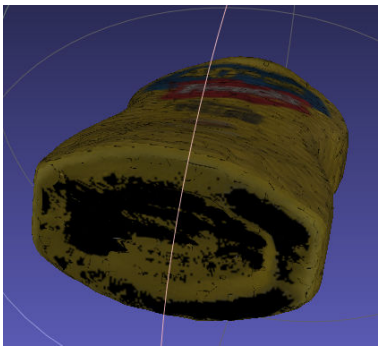
(c) The uncertainty map after the second grasp. Most surfaces on the sides are now observed, but the bottom is still uncertain.



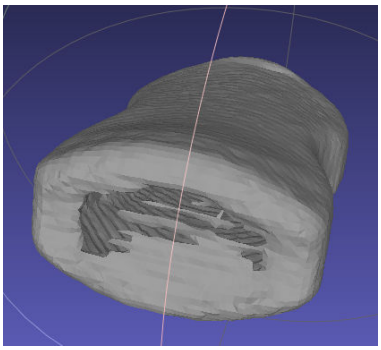
(d) Additional grasp selected to display uncertain surfaces. A very horizontal grasp is chosen so that the bottom of the object gets pointed towards the camera during the display trajectory.

(e) The uncertainty map after the additional grasp

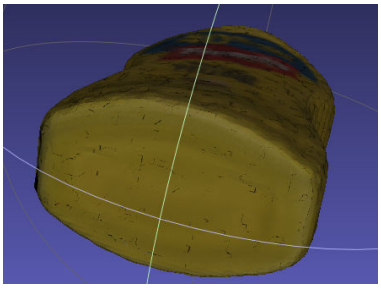
Figure 6.2: **Evolution of the object's uncertainty map as we collect views.** The uncertainty maps of the object are visualized as red for higher confidence and gray for lower confidence voxels. Due to slippage during the first (red in subfigure (a)) grasp, we do not get a good view of the bottom of the object initially. An additional grasp is computed in order to target collecting data of the unobserved bottom surface.



(a) Bottom of reconstructed mustard bottle using only data from grasps 1 and 2



(b) Geometry of bottom of reconstructed mustard bottle using only data from grasps 1 and 2



(c) Bottom of reconstructed mustard bottle using data from grasps 1 and 2 and additional uncertainty based grasp

Figure 6.3: Reconstructions using data collected before and after additional uncertainty based grasp from the scenario in Figure 6.2. These reconstructions were made from data gathered in simulation as a proof of concept.

6.2 Reconstruction Results

Qualitative results are provided in Figures 6.4 and 6.5. Notably, Figure 6.5 highlights our ability to reconstruct entire objects, including previously occluded regions such as the bottom, which would be inaccessible without object interaction. We found BundleSDF produces meshes with poor topology, including scattered boundary faces and non-manifold vertices. These artifacts complicate rendering and may pose challenges for simulators that require watertight meshes. One way to mitigate rendering issues is to use a high-quality but slow renderer like Blender Cycles [62]. Alternatively, a higher-quality mesh can be generated using a SOTA reconstruction method like Neuralangelo, following our geometric reconstruction recipe from Section 5.2. Figure 6.6 presents a qualitative comparison between BundleSDF and Neuralangelo, highlighting the rendering artifacts caused by poor topology. Additional examples, including Nerfacto and Gaussian Frosting reconstructions, are available on our project page. A dataset of 20 objects reconstructed through this pipeline can also be found on the project page.

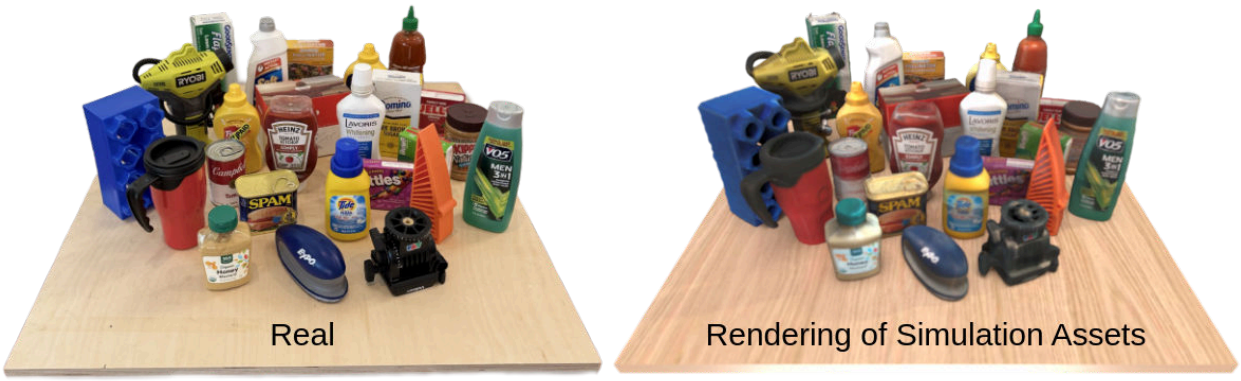


Figure 6.4: **Real-world objects (left) and their reconstructed counterparts (right).** Each object on the left was individually reconstructed using our pipeline. These assets were then manually arranged in simulation to approximately match their real-world poses and rendered to produce the image on the right. The strong visual similarity is notable, especially given that the reconstructions are rendered triangle meshes rather than neural renders.

Figure 6.5: **A selection of geometric reconstructions.** The first two columns show multiple views of the same object (a power inflator in the first column and a Lego block in the second), demonstrating the completeness of our reconstructions. The last two columns highlight close-up views of other objects, illustrating the accuracy of both geometric and visual reconstruction, even for parts that were occluded during scanning. We provide interactive 3D visualizations on our project page.

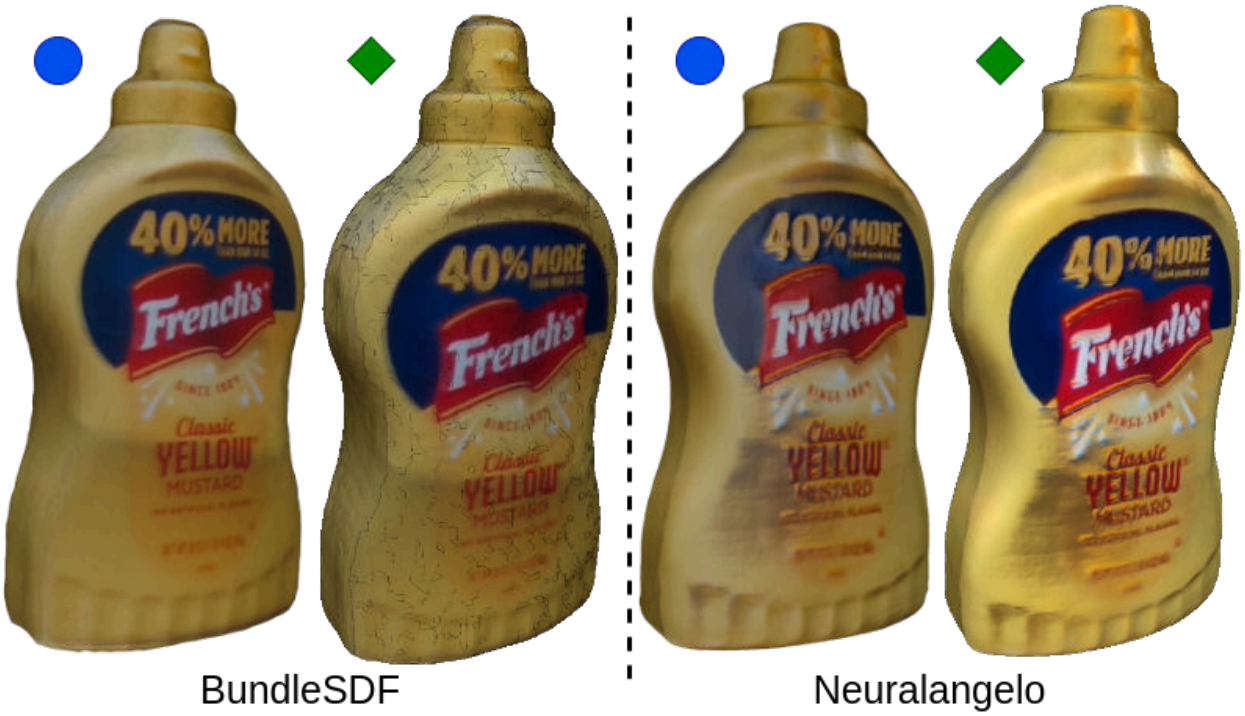


Figure 6.6: **Comparison of BundleSDF [27] and Neuralangelo [55] reconstructions of a mustard bottle.** Blue circles denote Blender [62] renders, while green diamonds represent Meshlab [63] renders. The BundleSDF mesh appears best in Blender but worst in Meshlab due to poor topology (e.g., scattered boundary faces), which requires a powerful renderer to compensate. In contrast, the Neuralangelo mesh maintains consistent quality across both renderers due to its well-structured topology. The effects of poor topology in the BundleSDF mesh appear as black lines, which originate from the mesh itself rather than the texture map. These artifacts are particularly noticeable at the top of the bottle's body.

We assess reconstruction accuracy by computing the Chamfer distance between our BundleSDF reconstructions of selected YCB [64] objects and their corresponding 3D scanner models from the original YCB dataset. Our method achieves reconstruction errors of 0.93mm, 1.68mm, 5.58mm, and 0.80mm for the mustard bottle, potted meat can, bleach cleanser, and gelatin box, respectively. These low errors indicate that our system produces both accurate and complete object reconstructions. However, if the physical product dimensions have changed since the YCB dataset's release, the measured reconstruction error may be artificially inflated.

6.3 Simulation Performance

Our pipeline produces simulatable assets that can be directly imported into physics simulators such as Drake. However, individual simulation rollouts are highly sensitive to initial conditions, making direct one-to-one comparisons with real-world rollouts challenging. A

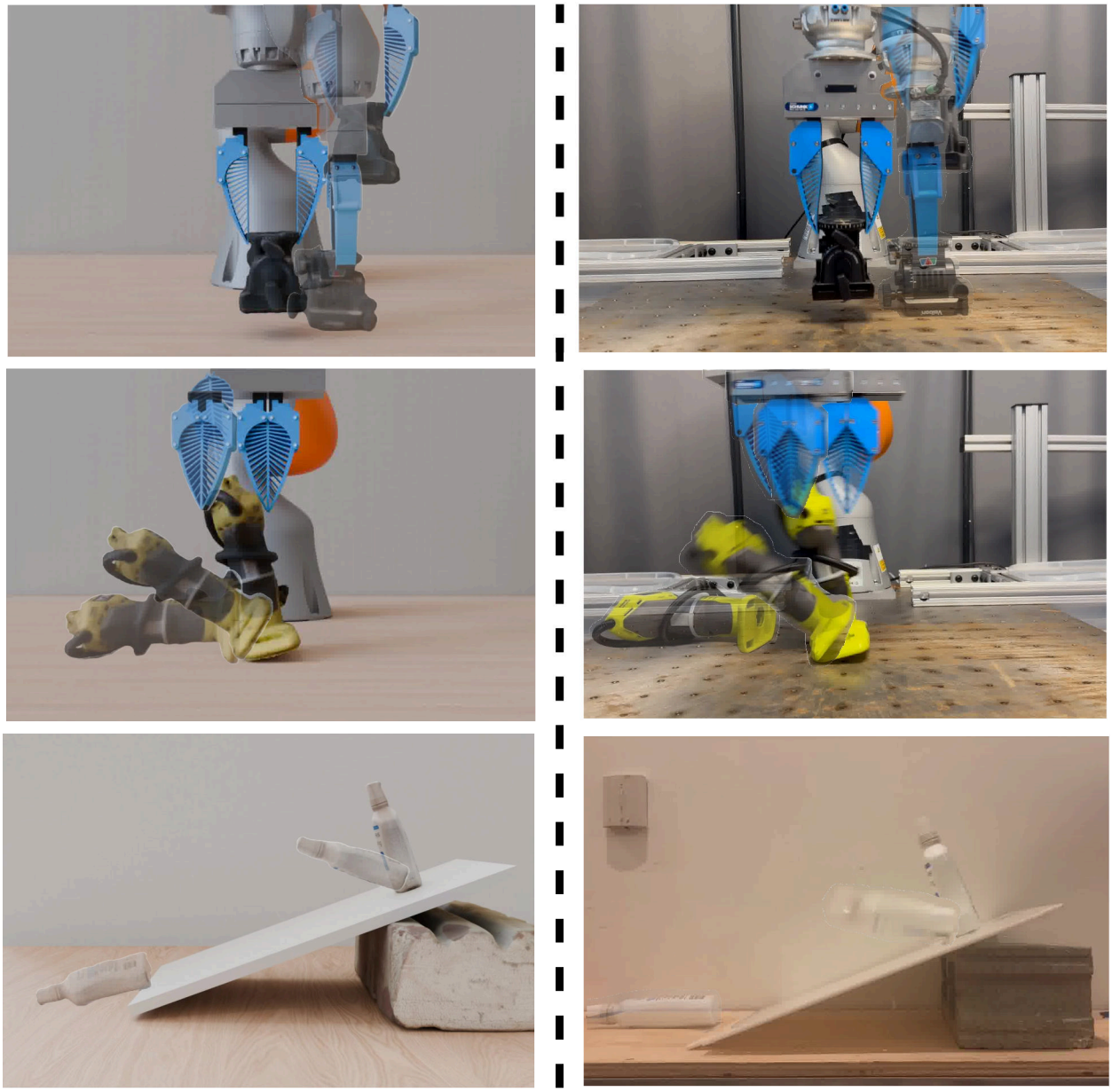


Figure 6.7: **Our simulation experiments.** The left column represents simulations, and the right column represents their real-world counterparts. The first row is pick-and-place, the second is knocking over, and the third is falling down a ramp. Different frames are overlaid transparently to show motion, and videos are available on the project page.

robust evaluation would require comparing the distributions of rollout trajectories rather than individual instances. One approach to mitigate sensitivity to initial conditions is to compute equation error metrics [65, Chapter 18] by resetting the simulation state to match the real-world state at every timestep. However, this would require a precise state estimation system, which is beyond the scope of this work. Instead, we focus on qualitative evaluation, presenting interactive simulations and side-by-side comparisons of real-world and simulated rollouts on our project page. See Figure 6.7 for an overview.

Chapter 7

Conclusion

We developed an autonomous pipeline for creating simulatable assets from real objects, integrated into a simple pick-and-place setup. Our work offers a stepping stone into bridging the Sim2Real gap by providing a scalable, accessible method for bringing real objects into simulation with high detail and accuracy.

7.1 Key Results

We found that with a standard pick and place setup with a 7-DOF robot arm, parallel finger grippers, 2 bins, and a set of RGBD cameras, we can construct a pick and display trajectory for an object of interest in the starting point with minimal (around 2) regrasps, to gather observations of the full surface of the object, following the generalized grasping and display strategies from Sections 4.2 and 4.3.

Additionally, we proposed a general recipe for reconstructing object-centric, dynamic scenes with occlusion that can be extrapolated to arbitrary photometric reconstruction methods and validated our recipe on 3 SOTA methods: Nerfacto, Guassian Frosting, and Neuralangelo, and found the method to work with all three, with interactive results online.

7.2 Contributions

In summary, the key contributions of this work are:

- A fully automated pipeline that generates complete simulation assets (visual geometry, collision geometry, and physical properties) using pick-and-place setups without hardware modifications or human intervention.
- A general recipe for obtaining object-centric triangle meshes from photometric reconstruction methods such as NeRF for moving, partially occluded objects by employing

alpha-transparent training and distinguishing foreground occlusions from background subtraction.

- Extensive real-world experiments validating the effectiveness of the pipeline and its individual components.
- A benchmark dataset of 20 assets generated by our pipeline, including raw sensor observations used in their creation. This dataset enables researchers to improve aspects of our pipeline, such as object tracking, geometry reconstruction, and inertia estimation, without requiring access to a robotic system.

7.3 Future Work

The immediate next step following this work would be to incorporate Method 2.5 from Section 4.3.3 for additional regrasping based on uncertainty on hardware. This would ideally involve integrating online, model-free object tracking such as with Any6D [52], and online image segmentation.

Looking forward, while our pipeline works on common pick and place setups, it would be interesting to collect data for reconstruction without disturbing the natural flow of the pick and place setup. One situation where this could be useful could be in a warehouse scenario, where multiple of the same object may be moved back and forth over the course of a day. After enough repetitions of the same, a camera stationed at the setup could collect enough data for 3D reconstruction. This would require keeping a constant data collection system to identify and group images of the same type of object and store those images towards the same reconstruction.

References

[1] Nicholas Pfaff, Evelyn Fu, Jeremy Binagia, Phillip Isola, and Russ Tedrake. *Scalable Real2Sim: Physics-Aware Asset Generation Via Robotic Pick-and-Place Setups*. 2025. arXiv: 2503.00370 [cs.RO]. URL: https://arxiv.org/abs/2503.00370.

[2] Adam Wei, Abhinav Agarwal, Boyuan Chen, Rohan Bosworth, Nicholas Pfaff, and Russ Tedrake. *Empirical Analysis of Sim-and-Real Cotraining Of Diffusion Policies For Planar Pushing from Pixels*. 2025. arXiv: 2503.22634 [cs.RO]. URL: https://arxiv.org/abs/2503.22634.

[3] Abhiram Maddukuri et al. *Sim-and-Real Co-Training: A Simple Recipe for Vision-Based Robotic Manipulation*. 2025. arXiv: 2503.24361 [cs.RO]. URL: https://arxiv.org/abs/2503.24361.

[4] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. *ACRONYM: A Large-Scale Grasp Dataset Based on Simulation*. 2020. arXiv: 2011.09584 [cs.RO]. URL: https://arxiv.org/abs/2011.09584.

[5] HeeSun Choi et al. "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward". *Proceedings of the National Academy of Sciences* 118.1 (2021), e1907856118. DOI: 10.1073/pnas.1907856118. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1907856118. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1907856118.

[6] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. "Legged Locomotion in Challenging Terrains using Egocentric Vision". In: *6th Annual Conference on Robot Learning*. 2022. URL: https://openreview.net/forum?id=Re3NjSwf0WF.

[7] Wenxuan Zhou and David Held. "Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity". In: *Conference on Robot Learning (CoRL)*. 2022.

[8] Zhikun Wang and Shiyu Zhao. *Sim-to-Real Transfer in Reinforcement Learning for Maneuver Control of a Variable-Pitch MAV*. 2025. arXiv: 2504.07694 [cs.RO]. URL: https://arxiv.org/abs/2504.07694.

[9] Roya Firoozi et al. "Foundation models in robotics: Applications, challenges, and the future". *The International Journal of Robotics Research* (2024).

[10] Andrew Wagenmaker, Kevin Huang, Liyiming Ke, Kevin Jamieson, and Abhishek Gupta. "Overcoming the Sim-to-Real Gap: Leveraging Simulation to Learn to Explore for Real-World RL". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=JjQl8hXJAS.

[11] Albert Yu, Adeline Foote, Raymond Mooney, and Roberto Martín-Martín. "Natural Language Can Help Bridge the Sim2Real Gap". URL: https://par.nsf.gov/biblio/10560811.

[12] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World". *CoRR* abs/1703.06907 (2017). arXiv: 1703.06907. URL: http://arxiv.org/abs/1703.06907.

[13] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. *Auto-Tuned Sim-to-Real Transfer*. 2021. DOI: 10.48550/ARXIV.2104.07662. URL: https://arxiv.org/abs/2104.07662.

[14] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. *Planar Robot Casting with Real2Sim2Real Self-Supervised Learning*. 2021. DOI: 10.48550/ARXIV.2111.04814. URL: https://arxiv.org/abs/2111.04814.

[15] Vladimír Petrík, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. "Learning Object Manipulation Skills via Approximate State Estimation from Real Videos". *CoRR* abs/2011.06813 (2020). arXiv: 2011.06813. URL: https://arxiv.org/abs/2011.06813.

[16] Zhenggang Tang, Balakumar Sundaralingam, Jonathan Tremblay, Bowen Wen, Ye Yuan, Stephen Tyree, Charles Loop, Alexander Schwing, and Stan Birchfield. "RGB-Only Reconstruction of Tabletop Scenes for Collision-Free Manipulator Control". In: *ICRA*. 2023.

[17] Arunkumar Byravan et al. *NeRF2Real: Sim2real Transfer of Vision-guided Bipedal Motion Skills using Neural Radiance Fields*. 2022. DOI: 10.48550/ARXIV.2210.04932. URL: https://arxiv.org/abs/2210.04932.

[18] Simon Le Cleac'h, Hong-Xing Yu, Michelle Guo, Taylor A. Howell, Ruohan Gao, Jiajun Wu, Zachary Manchester, and Mac Schwager. *Differentiable Physics Simulation of Dynamics-Augmented Neural Objects*. 2022.

[19] Luobin Wang, Runlin Guo, Quan Vuong, Yuzhe Qin, Hao Su, and Henrik Christensen. *A Real2Sim2Real Method for Robust Object Grasping with Neural Surface Reconstruction*. 2022. DOI: 10.48550/ARXIV.2210.02685. URL: https://arxiv.org/abs/2210.02685.

[20] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. *Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation*. 2024. arXiv: 2403.03949 [cs.RO]. URL: https://arxiv.org/abs/2403.03949.

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

[22] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[23] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[24] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction". *NeurIPS* (2021).

[25] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV*. 2020.

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[27] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Muller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. "BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects". *CVPR* (2023).

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. 2023. arXiv: 2308.04079 [cs.GR]. URL: https://arxiv.org/abs/2308.04079.

[29] Markus Worchel, Rodrigo Diaz, Weiwen Hu, Oliver Schreer, Ingo Feldmann, and Peter Eisert. "Multi-View Mesh Reconstruction with Neural Deferred Shading". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022.

[30] Rui Chen, Songfang Han, Jing Xu, and Hao Su. "Visibility-Aware Point-Based Multi-View Stereo Network". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.10 (2021), pp. 3695–3708. DOI: 10.1109/TPAMI.2020.2988729.

[31] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss. Cham: Springer International Publishing, 2018, pp. 55–71. ISBN: 978-3-030-01252-6.

[32] Matthew Tancik et al. "Nerfstudio: A Modular Framework for Neural Radiance Field Development". *arXiv preprint arXiv:2302.04264* (2023).

[33] William E. Lorensen and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, pp. 163–169. ISBN: 0897912276. DOI: 10.1145/37401.37422. URL: https://doi.org/10.1145/37401.37422.

[34] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction". In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 61–70. ISBN: 3905673363.

[35] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". *ACM Transactions on Graphics* 42.4 (July 2023). URL: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[36] Michael Krainin, Brian Curless, and Dieter Fox. "Autonomous generation of complete 3D object models using next best view manipulation planning". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5031–5037. DOI: 10.1109/ICRA.2011.5980429.

[37] Sho Kobayashi, Weiwei Wan, Takuya Kiyokawa, Keisuke Koyama, and Kensuke Harada. "Obtaining an Object's 3D Model Using Dual-Arm Robotic Manipulation and Stationary Depth Sensing". *IEEE Transactions on Automation Science and Engineering* 20.3 (2023), pp. 2075–2087. DOI: 10.1109/TASE.2022.3193691.

[38] Daryl Peralta, Joel Casimiro, Aldrin Michael Nilles, Justine Aletta Aguilar, Rowel Atienza, and Rhandley Cajote. *Next-Best View Policy for 3D Reconstruction*. 2020. arXiv: 2008.12664 [cs.CV]. URL: https://arxiv.org/abs/2008.12664.

[39] Xuran Pan, Zihang Lai, Shiji Song, and Gao Huang. *ActiveNeRF: Learning where to See with Uncertainty Estimation*. 2022. arXiv: 2209.08546 [cs.CV]. URL: https://arxiv.org/abs/2209.08546.

[40] Liren Jin, Xingguang Zhong, Yue Pan, Jens Behley, Cyrill Stachniss, and Marija Popović. *ActiveGS: Active Scene Reconstruction Using Gaussian Splatting*. 2025. arXiv: 2412.17769 [cs.RO]. URL: https://arxiv.org/abs/2412.17769.

[41] Liren Jin, Xieyuanli Chen, Julius Rückin, and Marija Popović. *NeU-NBV: Next Best View Planning Using Uncertainty Estimation in Image-Based Neural Rendering*. 2023. arXiv: 2303.01284 [cs.RO]. URL: https://arxiv.org/abs/2303.01284.

[42] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. *Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items*. 2022. arXiv: 2204.11918 [cs.RO]. URL: https://arxiv.org/abs/2204.11918.

[43] Klaus Greff et al. *Kubric: A scalable dataset generator*. 2022. arXiv: 2203.03570 [cs.CV]. URL: https://arxiv.org/abs/2203.03570.

[44] Huang Huang, Marcus Dominguez-Kuhne, Jeffrey Ichnowski, Vishal Satish, Michael Danielczuk, Kate Sanders, Andrew Lee, Anelia Angelova, Vincent Vanhoucke, and Ken Goldberg. *Mechanical Search on Shelves using Lateral Access X-RAY*. 2020. arXiv: 2011.11696 [cs.RO]. URL: https://arxiv.org/abs/2011.11696.

[45] Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. *KeypointDeformer: Unsupervised 3D Keypoint Discovery for Shape Control*. 2021. arXiv: 2104.11224 [cs.CV]. URL: https://arxiv.org/abs/2104.11224.

[46] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. *Grasp Pose Detection in Point Clouds*. 2017. arXiv: 1706.09911 [cs.RO]. URL: https://arxiv.org/abs/1706.09911.

[47] Russ Tedrake. *Robotic Manipulation. Perception, Planning, and Control*. 2024. URL: http://manipulation.mit.edu.

[48] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[49] Brian Curless and Marc Levoy. "A Volumetric Method for Building Complex Models from Range Images". In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 1st ed. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400708978. URL: https://doi.org/10.1145/3596711.3596726.

[50] Peter Werner, Richard Cheng, Tom Stewart, Russ Tedrake, and Daniela Rus. "Superfast Configuration-Space Convex Set Computation on GPUs for Online Motion Planning". *Under review* (2025).

[51] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. *FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects*. 2024. arXiv: 2312.08344 [cs.CV]. URL: https://arxiv.org/abs/2312.08344.

[52] Taeyeop Lee, Bowen Wen, Minjun Kang, Gyuree Kang, In So Kweon, and Kuk-Jin Yoon. *Any6D: Model-free 6D Pose Estimation of Novel Objects*. 2025. arXiv: 2503.18673 [cs.CV]. URL: https://arxiv.org/abs/2503.18673.

[53] Nikhila Ravi et al. "SAM 2: Segment Anything in Images and Videos". *arXiv preprint arXiv:2408.00714* (2024). URL: https://arxiv.org/abs/2408.00714.

[54] Antoine Guédon and Vincent Lepetit. "Gaussian Frosting: Editable Complex Radiance Fields with Real-Time Rendering". *ECCV* (2024).

[55] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. "Neuralangelo: High-Fidelity Neural Surface Reconstruction". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[56] Shilong Liu et al. *Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection*. 2023. arXiv: 2303.05499 [cs.CV].

[57] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

[58] E Lengyel Khaled Mamou and AK Peters. "Volumetric hierarchical approximate convex decomposition". In: *Game Engine Gems 3*. AK Peters, 2016, pp. 141–158. DOI: https://doi.org/10.1201/b21177.

[59] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. "Approximate Convex Decomposition for 3D Meshes with Collision-Aware Concavity and Tree Search". *arXiv preprint arXiv:2205.02961* (2022).

[60] Joseph Masterjohn, Damrong Guoy, John Shepherd, and Alejandro Castro. *Velocity Level Approximation of Pressure Field Contact Patches*. 2021. DOI: 10.48550/ARXIV.2110.04157. URL: https://arxiv.org/abs/2110.04157.

[61] Kai Chen et al. "MMDetection: Open MMLab Detection Toolbox and Benchmark". *arXiv preprint arXiv:1906.07155* (2019).

[62] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018.

[63] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. "MeshLab: an Open-Source Mesh Processing Tool". In: *Eurographics Italian Chapter Conference*. 2008. ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[64] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. "The YCB object and Model set: Towards common benchmarks for manipulation research". In: *2015 International Conference on Advanced Robotics (ICAR)*. 2015, pp. 510–517. DOI: 10.1109/ICAR.2015.7251504.

[65] Russ Tedrake. *Underactuated Robotics. Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. 2023. URL: https://underactuated.csail.mit.edu.