Improving Robotic Manipulation via Reachability, Tactile, and Spatial Awareness

Iretiayo Adegbola Akinola

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2021

Abstract

Improving Robotic Manipulation via Reachability, Tactile, and Spatial Awareness

Iretiayo Adegbola Akinola

Robotic grasping and manipulation remains an active area of research despite significant progress over the past decades. Many existing solutions still struggle to robustly handle difficult situations that a robot might encounter even in non-contrived settings. For example, grasping systems struggle when the object is not centrally located in the robot's workspace. Also, grasping in dynamic environments presents a unique set of challenges. A stable and feasible grasp can become infeasible as the object moves; this problem becomes pronounced when there are obstacles in the scene. This research is inspired by the observation that object-manipulation tasks like grasping, pick-and-place or insertion require different forms of awareness. These include *reachability awareness*– being aware of regions that can be reached without self-collision or collision with surrounding objects; *tactile awareness*– ability to feel and grasp objects just tight enough to prevent slippage or crushing the objects; and *3D awareness*– ability to perceive size and depth in ways that makes object manipulation possible. Humans use these capabilities to achieve a high level of coordination needed for object manipulation. In this work, we develop techniques that equip robots with similar sensitivities towards realizing a reliable and capable home-assistant robot.

In this thesis we demonstrate the importance of reasoning about the robot's workspace to enable grasping systems handle more difficult settings such as picking up moving objects while avoiding surrounding obstacles. Our method encodes the notion of reachability and uses it to generate not just stable grasps but ones that are also achievable by the robot. This reachability-aware formulation effectively expands the useable workspace of the robot enabling the robot to pick up objects from difficult-to-reach locations. While recent vision-based grasping systems work reliably well achieving pickup success rate higher

than 90% in cluttered scenes, failure cases due to calibration error, slippage and occlusion were challenging. To address this, we develop a closed-loop tactile-based improvement that uses additional tactile sensing to deal with self-occlusion (a limitation of vision-based system) and adaptively tighten the robot's grip on the object– making the grasping system tactile-aware and more reliable. This can be used as an add-on to existing grasping systems. This adaptive tactile-based approach demonstrates the effectiveness of closed-loop feedback in the final phase of the grasping process. To achieve closed-loop manipulation all through the manipulation process, we study the value of multi-view camera systems to improve learning-based manipulation systems. Using a multi-view Q-learning formulation, we develop a learned closed-loop manipulation algorithm for precise manipulation tasks that integrates inputs from multiple static RGB cameras to overcome self-occlusion and improve 3D understanding. To conclude, we discuss some opportunities and directions for future work.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I am super grateful to God who has blessed me with an abundance of opportunities and for the people he has strategically placed as help at various junctures in my journey.

My PhD years have been a time of growth research-wise and professionally, primarily due to the conducive environment created by my advisor Peter Allen. I have enjoyed his full support on all fronts – research direction and professional tutelage on how to be an excellent multi-disciplinary researcher. He gave me the intellectual room to engage in multiple research efforts and grow as a researcher. For all these and many more, I remain grateful to Peter.

Jacob Varley, a senior PhD candidate at the time I joined, let me into the robotics lab on my first day on Columbia campus in the middle of winter. He ensured that I had the best possible transition from an Electrical Engineering background into the nitty-gritties of robotics research. His counsel on all things including research, career, and others have been super valuable. I am very glad to call him a friend. I also greatly benefitted from my interactions with the Faculty of the department providing advisory and academic support at every turn. Particularly, Paul Sadja, Matei Ciocarlie, Shuran Song, Carl Vondrick and Augustine Chaintreau. Thanks for making yourself available at all time for both short and long conversations. I am also grateful to Anne Flemming, Jessica Rosa, Maria Joanta, Cindy Meekins, Ivy Elkins and the entire staff of the department for all their support in navigating all the milestones of the program.

My time at Columbia has been enriching due to the excellent collaborators I have had the privilege of interacting with. I was always mentally stimulated and challenged by my interactions with Boyuan Chen, Bohan Wu, Zizhao Wang, Jingxi Xu, Jack Shi, Abhi Gupta, David Watkins, Chad DeChant, Jonathan Koss, Carlos Martin, Aalhad Patankar, Xiaomin He, Pawan Lapborisuth, Vaibhav Vavilala among others.

I would like to thank the sponsors of my PhD program: the *Presidential Special Scholarship for Innovation and Development, Nigeria* was vital in the early years of my program and the *Microsoft Research PhD Fellowship Program* provided invaluable support over the last few years of my PhD program. The awards relieved a lot of pressure and enabled me to focus on my PhD research in a meaningful way.

My research internships at AutoDesk and Google gave me unique perspectives and exposure to industry-level research and an opportunity to learn from excellent researchers and good people. I benefitted greatly from my interactions with Heather Kerrick, Evan Atherton, Nick Cote and the entire team at the Autodesk Applied Research Lab. At Google, I had the honor of working closely with Jacob Varley, Dmitry Kalashnikov, Vikas Sindhwani, Anelia Angelova, Micheal Ryoo, Yao Lu and the entire robotics research teams. The various interactions facilitated my growth as a robotics researcher.

Last but not the least, I am deeply indebted to my family for all the support and encouragement every step of the way. My wife, Simone Fobi, is the first filter that has the challenge of listening to and refining my very raw ideas. My deepest gratitude to my parents for giving me the foundation and the belief to even take on the daunting journey of the PhD. I am so grateful for their counsel, encouragement and prayers. Finally, I would like to thank my siblings, close family and friends for their love and support.

To my family

Chapter 1

Introduction

Robotic manipulation research has made significant progress in developing high-performing algorithms that work well in static well-structured environments. These largely open-loop algorithms work well when the target objects are right in the middle of the robot's workspace, singulated without clutter, firmly affixed so as not to move on contact, and generally grasped from a limited approach direction (e.g. overhead). However, robotic manipulation becomes much harder in cluttered, dynamic environments where the objects can be static or moving, collision avoidance is harder, and the target object may be near the reachable limits of the robot workspace. For example, picking up a moving object from a conveyor belt or during a human-robot handover requires the robot to reason about which part of the object is within reach, what the motion profile of the object is, what geometric properties of the object would aid a successful grasp, how tight to hold the object once picked etc.

Humans are able to reason about these elements to seamlessly manipulate objects; getting robots to achieve a similar level of coordination remains a challenge. One main difference between human and robotic grasping systems is situational awareness of the environment. Specifically, two of the main awarenesses possessed by humans to aid object manipulation are *spatial awareness* and *sensor awareness*. Using our eyes and head motions, humans are generally aware of the spatial arrangement of objects and obstacles in a grasping environment; and we also possess proprioceptive kinesthetic awareness of our own *grasping hardware* (i.e. arm, hand and fingers). One aim of this thesis is to mimic these sensibilities in robots.

1.1 Spatial Awareness

In this dissertation we present grasping and manipulation algorithms that equip robots with reachability awareness - being aware of regions that can be reached without self-collision or colliding with surrounding objects. Robotic grasping for static and moving objects in the presence of clutter requires generating grasps that satisfy three requirements: 1) stable grasps to enable pick up success; 2) feasible grasps that the robot can reach and 3) grasps that can be achieved via collision free arm motion. Most grasping systems optimize for just the first objective i.e. grasp stability. This assumes that the grasp generated can be reached and achieved by the robot. However, the generated grasp solutions might be unusable for the robot if they do not satisfy the other collision-free reachability requirement.

This challenge of generating stable infeasible grasps stems from the fact that grasp planning and motion generation are decoupled in many systems. To address this, we use a grasp planning formulation that incorporates information about the robot's workspace used for arm motion generation into the grasp planning objective. We explore the application of this reformulation in different settings especially along two dimensions: first grasping a target object in the presence of obstacles and second grasping static versus moving objects.

1.2 Sensor Awareness

Adding external sensory data is a hallmark of the prototypical sense-perceive-act robotic cycle. It is especially important in grasping, where many DOF's are in play in complex spatial environments. Adding sensor awareness can create closed-loop algorithms that can adjust to changing, dynamic conditions during grasping. The vision and tactile capabilities of humans are amazing, and imbuing robots with some of these capabilities is a continuing challenge.

1.2.1 Tactile Awareness

Even when the objects are well within the reachable region of the robot, high performing grasping algorithms still suffer from common failure cases like poor calibration, object slip, and collisions among others. These failure cases usually show up in the final moments of the grasping process and require online adjustments which are difficult for the popular purely vision-based open-loop systems. Typically, a vision system is occluded by the robotic hand when these problems occur, rendering it useless. One solution is to enable tactile feedback from finger/palm contacts to reason about the state of a grasp and also its potential recovery if failing. We will present a closed-loop algorithm that uses tactile and proprioceptive information to adaptively act through both fine finger motions and larger regrasp movements to execute stable grasps.

1.2.2 Visual 3D Awareness

Beyond grasping, there are other manipulation tasks like stacking, insertion and precision kitting, that require a higher level of manipulation precision. These sorts of tasks require accurate 3D geometric knowledge of the task environment including object shape and pose, relative distances and orientation between key locations in the scene among others. For example, solving an insertion task requires picking up an object using the geometry and pose of the object and sticking it in a hole using the pose of object relative to the hole.

Many of the existing vision-based robotic manipulation systems employ a single camera to observe the task scene. However, the rich 3D information required for solving precision-based tasks is usually limited from a single camera input. For example, it is usually hard to resolve scale and alignment from a single view. Even for humans, navigating a room or completing a task with one eye closed becomes more challenging from a lack of depth perception. In addition, single view systems are very susceptible to occlusion during task learning requiring the robot to actively move out of the way and reset during task execution.

To address these limitations, we propose using a multi-view camera setup to solve precision-based object manipulation. Since cameras are cheap and ubiquitous, adding a few more cameras to capture multiple views of the task scene is a practical and feasible option. This research develops techniques for combining multiple camera views to improve the state estimation and increase the robustness of robot action in learning-based robotic manipulation systems. Our approach is a reinforcement learning based method that takes in multiple color (RGB) images from different viewpoints as input and produces safe, collision-free robot actions in a closed-loop fashion.

1.3 Contributions Per Chapter

Our contributions presented in the chapters of this dissertation are as follows:

- In Chapter 2, we present a workspace-aware grasping framework that incorporates a notion of reachability into the online grasp planning process. This framework greatly improves the performance of standard online grasp planning algorithms by biasing the hand towards reachable end-effector configurations during grasp search effectively reducing the search space. The bias keeps the grasp planner in accessible regions of the planning scene so that the resulting grasps are tailored to the situation at hand. This results in a higher percentage of reachable grasps, a higher percentage of successful grasp executions, and a reduced planning time. We also present experimental results using simulated and real environments.

- In Chapter 3, we show that we can extend the reachability-aware grasping approach to improve grasping in dynamic environments, where objects might be moving on a conveyor belt or during handovers between humans and robots. In addition, we use object motion prediction and trajectory seeding to continually generate arm motion till the grasp is realized. Experiments on linear and smooth nonlinear motions demonstrate that our method outperforms the baseline methods in picking fast moving targets within static obstacles.

- In Chapter 4, we introduce Multi-Fingered Adaptive Tactile Grasping, or MAT– a high-performance deep reinforcement learning (RL) algorithm that leverages tactile and proprioceptive information for multi-fingered grasping in an adaptive, closed-loop manner, improving state-of-the-art open-loop grasping systems. By smartly choosing observation and action modalities that maintain small sim-to-real gaps, MAT is trained in simulation and directly transfers to real in a high fidelity way (without additional learning). Finally, MAT demonstrates substantially improved pick-up success rates in real-robot experiments over a vision-based, open-loop grasping system.

- In Chapter 5, we present an effective multi-view approach to closed-loop end-to-end learning of precise manipulation tasks that are 3D in nature. Our deep-RL algorithm learns to accomplish these tasks using multiple statically placed but uncalibrated RGB camera views without building an explicit 3D representation such as a pointcloud or voxel grid. This multi-camera approach achieves superior task performance on difficult stacking and insertion tasks compared to single-view baselines.

Chapter 2

Workspace Aware Online Grasp Planning

In this chapter, we develop an approach that reasons about the robot arm kinematic structure and reachability during grasp generation[1]. Specifically, we encode a humanoid robot's reachable spots into a novel signed-distance-field representation in an offline stage. This representation is then used to constrain grasp planning within accessible regions, enabling it to generate not just stable grasps but ones that are also achievable by the robot. This approach leads to increased grasp execution success and reduced planning time. We demonstrate that our method enables to robot to reliably grasp objects near the limit of the robot's workspace, efffectively expanding the usable workspace of the robot.

2.1 Workspace-Awareness during Grasp Planning

Grasp planning and motion planning are two fundamental problems in the research of intelligent robotic manipulation systems. Most of the research has treated these problems as

[1] This work first appeared in IROS 2018: *Workspace Aware Online Grasp Planning*, by Iretiayo Akinola, Jacob Varley, Boyuan Chen, and Peter Allen[3]

distinct research areas focusing either on: 1) how to get a set of high quality candidate grasps [18][35] or 2) how to generate viable trajectories to bring the gripper to the desired hand configuration [96][92][25]. While it is often convenient to treat grasp and motion planning as distinct problems, we demonstrate that solving them jointly can improve performance and reduce computation time. Grasp planners unaware of the robot workspace and without a notion of reachability often spend significant time and resources evaluating grasps that may simply be unreachable. For example, a reachability unaware planner is equally likely to return impossible grasps that assume the robot will approach the object from the object's side furthest from the robot (see Figure 2.1). Our planner avoids unreachable areas of the workspace dramatically reducing the size of the search space. This lowers online planning time, and improves the quality of the planned grasps as more time is spent refining quality grasps in reachable portions of the workspace, rather than planning grasps that may be stable but are unreachable.

In order to generate grasps which are stable and reachable, we propose a new energy function for use with simulated annealing grasp planners [18][35]. This energy function is a weighted combination of a grasp stability term, and a grasp reachability term. Our grasp reachability term is inspired by the idea of a precomputed reachability space which has demonstrated utility in other robotics tasks [85][100]. In our work, we generate a densely sampled reachability space for our robot as shown in Figure 2.3(Top Row). This offline computation checks whether an *inverse kinematic* (IK) solution exists for a given pose. This database in and of itself has utility for filtering lists of precomputed-grasps and quickly removing unreachable grasps, leaving only reasonable grasps as possible results. We further postprocess our reachability space and compute a *signed distance field* (SDF) representation in Figure 2.3(Bottom Row). The SDF is used in our online grasp planning energy function

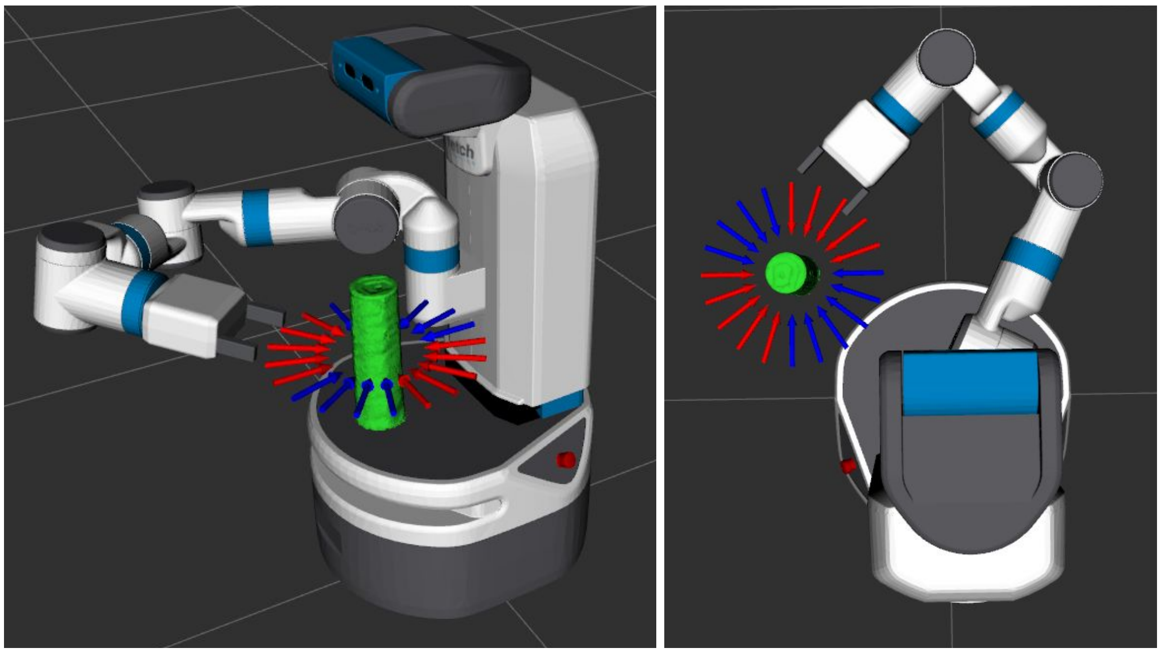


Figure 2.1: Reachability results for uniformly sampled grasps (Red arrows reachable, blue unreachable). Even for objects well within the bounds of the robot workspace, there are many invalid approach directions which are not easily modeled by simple heuristics.

to guide the hand towards reachable regions of the robot's workspace. Obstacles present in the grasping scene can be embedded into the space prior to SDF generation which results in a field that repels the grasp search away from poses that collide with the obstacles.

Experiments in both simulation and physical environments have been performed to assess the performance of our method on multiple objects and under various poses. We demonstrate that our reachability aware grasp planner results in a larger number of reachable grasps, a larger number of successful grasp executions, and a reduced planning time compared to other online grasp planning methods.

Our method for workspace aware online grasp planning is overviewed in Figure 2.2. This framework has a number of advantages. First, it combines two closely coupled processes into one; rather than going back and forth between grasp planning and trajectory planning

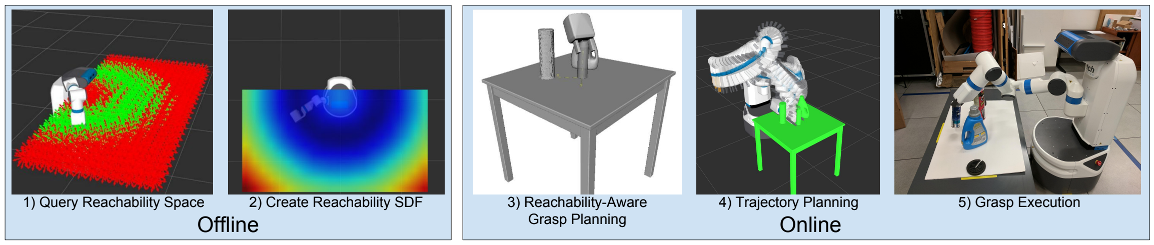

1) Query Reachability Space
2) Create Reachability SDF
Offline
3) Reachability-Aware Grasp Planning
4) Trajectory Planning
Online
5) Grasp Execution

Figure 2.2: Workspace Aware Online Grasping Framework - **Offline**: 1) the robot's reachability space is queried for IK solutions that are free of collisions with the robot itself and fixtures such as walls and tables. 2) An SDF is created from the reachability space. **Online**: 3) Grasp planning is quickly accomplished utilizing the reachability space SDF. 4) A motion plan for one of the planned grasps. 5) Trajectory executed by the robot for a stable grasp.

until a feasible grasp is found, we solve for a fully reachable grasp at once. It increases the probability of finding feasible grasps quickly since the optimization process is guided by our energy function within reachable spaces. Also, our method is well suited for online planning which in general is more adaptable and robust than the traditional offline grasp planning approach and works in the case of novel objects for which precomputed grasps do not exist.

2.2 Offline Reachability Space Generation

A grasp is reachable if a motion plan can be found to move the arm from its current configuration to a goal configuration that places the hand at desired grasp location. This is not always possible for a number of reasons typically because no inverse kinematics(IK) solution exists to place the hand at the desired grasp pose, self collision with other parts of the robot, or collision with obstacles in the planning scene.

2.3 Reachability Space Representation

We observe that this definition of the original reachability space is binary, either 1 (reachable), or 0 (not reachable), and has no gradient that indicates the direction from non-reachable regions to reachable portions of the workspace. In the context of *simulated annealing* for grasp planning, it is beneficial to provide an energy function that has a landscape that can guide solutions to more desired regions of the annealing space. This observation informed a novel signed-distance-field (SDF) representation of the reachability space that is amenable to optimization formulations. Our SDF maps a grasp pose to a value representing the distance to the manifold or boundary between the unreachable poses and reachable poses and can be interpreted as follows $d_{sdf} = SDF(pose)$:

- $d_{sdf} = 0$: pose lies exactly on boundary between reachable and unreachable grasp poses.
- $d_{sdf} > 0$: pose lies within the reachable region of the workspace, and is distance d_{sdf} away from the boundary.
- $d_{sdf} < 0$: pose lies outside the reachable region of the workspace, and is distance d_{sdf} away from the boundary.

This field can then serve as a criteria for classifying a grasp as either lying in a reachable space or not. And nearby hand configurations can be ordered based on d_{sdf}, their distance from this boundary.

Grid resolution and metric choices are two important parameters for the reachability space and SDF generation. The 6D hand pose of a given grasp has the 3D translational (linear) component and the 3D rotation (angular) component, two physically different quantities with different units. We systematically determine good resolution levels and a good ratio between unit linear measurements and unit angular measurements, to obtain a unified

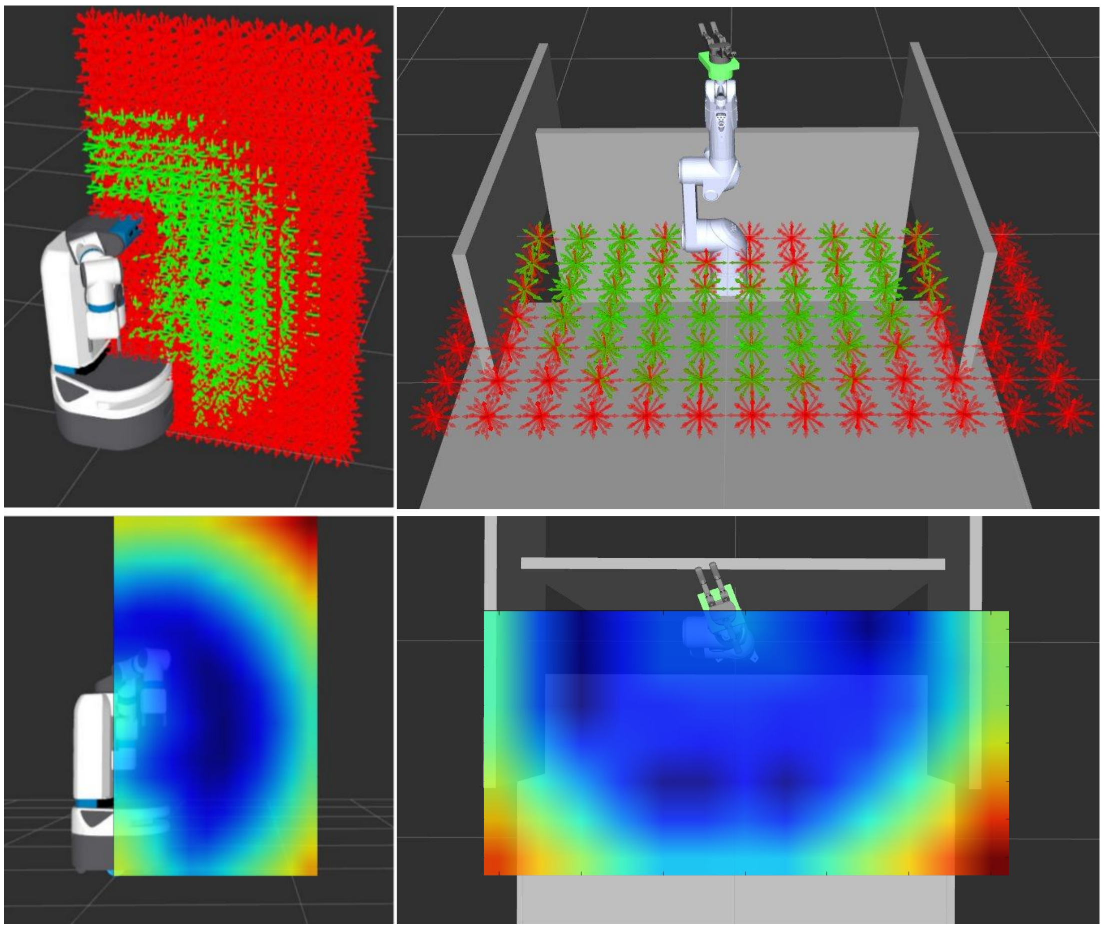


Figure 2.3: Top Row: Visualization of cross sections of the precomputed reachable space for a Fetch Robot and Staubli Arm with Barrett Hand. Green arrows represent reachable poses, red arrows unreachable. This space is computed offline, once for a given robot. Bottom Row: Signed Distance Field generated from the above reachability spaces.

6D metric space suitable for our purpose. Effectively, our metric can be defined as:

$$d_{sdf} = \pm\sqrt{||\Delta xyz/res_{lin}||^2 + r||\Delta rpy/res_{rot}||^2} \tag{2.1}$$

where Δxyz and res_{lin} (in centimeters) are the translational distance and resolution respectively, Δrpy and res_{rot} (in radians) are the rotational distance and resolution, and r is the relative metric ratio i.e. a distance of res_{lin}cm $\equiv r * res_{rot}$rad.

We vary translational resolution $res_{lin} = 5, 10, 20$ cm, angular resolution $res_{rot} = \pi/8, \pi/4, \pi/2$ rad, and translational to rotational metric ratio $r = 0.1, 1, 2, 5, 10$; and checked the performance of the SDF generated for each combination. To evaluate different resolution combinations, we first randomly generate 10,000 grasp poses and checked for the existence of an IK solution i.e. the grasp's reachability. Then, for each of the resolution levels, we evaluate the quality of the resulting SDF by checking what percentage of the random grasp poses were correctly classified as reachable (or not) by the SDF manifold. Reachable grasps evaluate to positive sdf values, unreachable grasps evaluate to negative sdf values. The time to generate SDF from binary reachability space is recorded in each case. From this parameter sweep, the densest resolution level $res_{lin} = 5$cm, $res_{rot} = \pi/8$rad gives 0.992 & 0.973 as accuracy and precision respectively but takes 177secs to generate the SDF. We observe that 10cm translational, $\pi/4$ rotational resolution levels and metric ratio $r = 1$ still gives a high accuracy (0.979) and precision (0.92) with the sdf generation time under 3secs. These choices define the metric space for the 6D hand pose reachability space: a 10cm translation and a $\pi/4$rad rotation are equidistance in the reachability space. This resolution analysis is done on the Fetch robot and the resolution choices is found to work well for other robots as well. Note that we take into consideration the cyclical nature of the rotational degrees of freedom during SDF generation as angles wrap around at 2π.

2.4 Online Reachability-Aware Grasp Planning

Grasp planning is the process of finding "good grasps" for an object that can be executed using an articulated robotic end effector. A grasp is typically classified as "good" based on how well it can withstand external disturbances without dropping the object. The Ferrari-Canny method [30] is a common way to evaluate robot grasps; it defines how to quantitatively measure the space of disturbance wrenches that can be resisted by a given grasp. Other techniques are typically built around this metric as shown in this review [87]. Grasp planning frameworks such as GraspIt! use these grasp metrics of force and form closure as objective functions for optimization. Since this formulation has no notion of reachability, the grasp results, while stable, might require the end effector to be placed in a pose that is impossible to reach given the robot's current location.

2.4.1 Simulated Annealing for Grasp Planning

Grasp planning can be thought of as finding low energy configurations within the hand object space. This search is done in a multidimensional space where grasps are sampled and evaluated. The 6 dimensional (x, y, z, roll, pitch, yaw) space we generated with our SDF is a subspace of the 6 + N dimensional grasp search space. The additional N dimensions represent the EigenGrasps or eigen vectors of the end effector Degree of Freedom (DOF) space as described in [18]. For the Fetch (1 DOF), N=1, and for the Barrett Hand (4 DOF), N=2. These 6 + N dimensions describe both the pose and DOF values of the end effector. The goal of grasp planning is to find low energy points in this 6 + N dimensional space. These points represent the pose and hand configuration of "good grasps".

Simulated annealing (SA) [42] – a probabilistic technique for approximating global optimums for functions lends itself nicely for our formulation. SA presents a number of advantages in grasp planning[4]. For example, it does not need an analytical gradient which can be computationally infeasible; it handles the high nonlinearity of grasp quality functions; and it is highly adaptable to constraints. Our approach implicitly guides the annealing process ensuring that sampling is done in regions of reachable space which increases the probability of getting useful grasp solutions. While online grasping is typically avoided due to time cost of exploring a larger search space, this work makes online grasping more feasible since the majority of the space which is unreachable need not be searched. With our new energy formulation, the annealing process will quickly drive the hand towards reachable grasp locations. Section 2.5 below shows that this new energy function increases the success probability of online grasping significantly.

Algorithm 1 Reachability-Aware Energy (SA-OURS)

```
1:  procedure ReachabilityAwareEnergy
2:      $e_p$ = potentialEnergy()
3:      $e_{reach}$ = reachabilityEnergy()
4:      stable = $e_p > 0$
5:      reachable = $e_{reach} < 0$
6:      if reachable && stable then
7:          return $e_p + \alpha_1 \cdot e_{reach}$
8:      else if reachable && !stable then
9:          $e_{contact}$ = contactEnergy()
10:         return $e_{contact} + \alpha_2 \cdot e_{reach}$
11:     else //!reachable
12:         $e_{contact}$ = contactEnergy()
13:         return $e_{contact} + \alpha_3 \cdot e_{reach}$
```

2.4.2 Novel Grasp Energy Formulation

In our work, we augment the *Contact and Potential* grasp energy function from [18]. The potential energy term (e_p) measures the grasp's potential to resist external forces and torques while the contact energy term ($e_{contact}$) defines the proximity of the hand to the object being grasped. These two terms define the grasp fitness function used by the optimization algorithm (simulated annealing) to search for good grasp configurations. Our *Reachability Aware* method augments the *Contact and Potential* energy function with a *reachability* energy term (e_{reach}) that encodes the kinematic constraints of the robot. We use the reachability SDF ($e_{reach} := d_{sdf}$) described previously as a regularizing term to obtain a new cost functions that drives the grasp planning optimization towards reachable hand configurations. As shown in [80] discretized SDFs still model and behave like a continuous function so it can be used during the simulated annealing optimization process to drive the search towards reachable hand configurations. Our reachability-aware energy function E is given as:

$$E = G + \alpha R \tag{2.2}$$

where G is a metric of grasp quality (e.g. force closure), R is a measure of reachability of a given grasp pose and α defines the tradeoff between the conventional grasp metric and the reachability value. To optimize the overall grasp energy, we use the simulated annealing search according to [19].

In terms of implementation, we build on the GraspIt! simulator and combine the existing energy metrics with the new reachable energy term. The energy components have forms that we exploit. The grasp energy term (as described in [18]) is negative when the grasp has force closure and positive otherwise. The reachability energy has similar meaning which gives four quadrants of possibilities. We chose the weights of each quadrants so the range of values for the reachability term and grasp term are of the same order of magnitude. This provides a gradient from bad to good quality (reachable and force-closure) grasps.

Algorithm. 1 shows specifically how the energy terms are combined. α values were set to $\alpha_1 = -0.1$, $\alpha_2 = -10$, $\alpha_3 = -10$ to obtain a function where more negative energy values correspond to more reachable stable grasps, and positive high energy values correspond to unreachable and unstable grasps.

Since our reachability space is represented as a grid of discretized SDFs, we use multilinear interpolation to get ($e_{reach} := d_{sdf}$) for a given grasp pose from the reachability SDF grid. Each query pose during the optimization is situated in the robot's reachable space to get the cell location and the reachability value is given as the weighted sum of the values for the 2^N ($N = 6$) corners of the hypervoxel where the query point falls in the pre-computed SO(6) reachable space. The 2^N corner values are looked up in the pre-computed reachability space.

$$R[p_{query}] = \sum_{i=1}^{2^N} w_i R[p_i] \tag{2.3}$$

where $p_{query} = [x_q, y_q, z_q, r_q, p_q, y_q]$ is the query pose, $p_i = [x_i, y_i, z_i, r_i, p_i, y_i]$ for $i = 1...2^N$ correspond to the neighbouring corners in the reachability space grid and w_i is the proportion of the grid occupied by creating a volume from connecting the query point and the corner p_i. See [107] for more details on multilinear interpolation.

2.4.3 Embedding Obstacles in the Reachability Space

Obstacles whose poses are not expected to change in relation to the robot can be placed in the scene while generating the initial reachability space. This approach is applicable to modeling tables, walls and fixtures around fixed base robotic arms, or static room environments for mobile manipulators. This is useful for the Staubli Arm and attached Barrett Hand as shown in Figure 2.3. This robot is fixed to the table, next to the walls, making it reasonable to incorporate these obstacles during the initial reachability space creation.

In order to incorporate other static obstacles detected at runtime into our precomputed reachability space, we mask out regions in the robot's binary reachability space that overlap with any detected obstacle prior to generating the SDF representation; this corresponds to modifying the reachable space in the first block of Figure 2.2. Given the obstacles' geometries and poses, we first collate grid locations of the reachability space that overlap with obstacle geometries. All grasp poses at these locations (including those that were otherwise reachable) are marked as unreachable. Then, we regenerate an SDF representation of the resulting binary reachable space. The SDF generation is fast, taking 2-3 seconds on average, and this step is only required when the workspace changes. While this fast obstacle-masking-procedure considers only the hand (not full arm configuration) and might miss out on some other grasp poses that become infeasible due to obstacles far from the end-effector, we've found that it is a reasonable approximation that works well in practice, especially when the obstacles affect arm links close to the end-effector. The masked out obstacle locations imposes a negative field that pushes the SDF manifold further away from the obstacles and our experiments show that this effect results in an improved grasp planner.

2.5 Experiments

We describe different experiments in both simulation and physical environments to assess the performance of our method on multiple objects and under various poses. All experiments (simulated and real) are run on two robot platforms: the Fetch mobile manipulator and the Barrett hand mounted on a Staubli Arm (Staubli-Barrett). The Staubli-Barrett set-up (see Figure 2.3) has additional walls to limit the workspace of the robot as a safety precaution to ensure the robot does not elbow objects outside of it's workspace during grasping. This safety box heavily constrains the range of grasping directions as many grasp poses will be invalidated because of collision with the walls. Workspace fixtures such as the walls were included in the scene when constructing the reachability space offline.

For each experiment, we compared a typical grasp planning method with our reachability-aware method:

- **Sim. Ann. Contact and Potential (SA-C&P)**: Simulated Annealing in GraspIt! using the Contact and Potential energy function.
- **Sim. Ann. Reachability-Aware (SA-Ours)**: Simulated Annealing in GraspIt! using the newly proposed energy function from Algorithm 1.

2.5.1 Evaluation Metrics

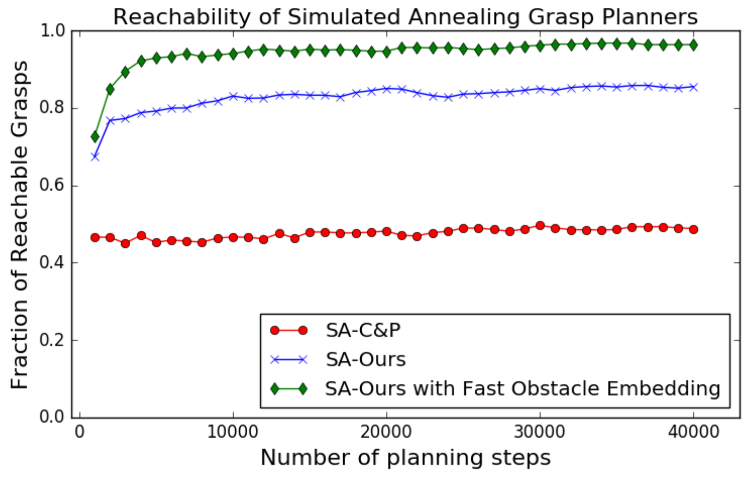
The metrics used for evaluating our methods include:

- **Percent Reachable Grasps**: The number of reachable grasps divided by the total number of grasps generated from a run of a given grasp planner.
- **Number of Required Plan Attempts**: The grasp planner returns a list of grasps ordered according to quality. We record how many of these grasps we have to go through until a valid arm trajectory can be planned.
- **Lift Success**: Here we execute the first valid grasp from the set returned by the planner. A grasp is successful if the gripper placed at the grasp pose can successful lift the object off the ground. We use the Klamp't simulator [36] to test this in simulation. For the real world experiments, we executed the grasps on the real robot and checked if the object was picked up.

2.5.2 Grasp Planning with Runtime Obstacles Experiment

Here we set-up a typical grasp planning scenario: in a virtual scene, three objects are placed on a table. One is the target object to be grasped while the other two are ”runtime detected” obstacles that the robot must not collide with during grasping (Figure 2.5). We use 4 different meshes as the target object. We place the target object in 9 difficult-to-reach poses that are either at the extent of the robot's workspace, or extremely close to the robot. Both of these situations drastically limit the number of valid approach directions that the robot can use to grasp the object. For each pose, GraspIt!'s Simulated Annealing planner is run for varying number of planning steps using both the **SA-C&P** and **SA-Ours** energies. We use two versions of our reachability-aware planner (**SA-Ours**), one with *unmodified SDF* (Section 2.4.2) and the other with *obstacles-embedded SDF* (Section 2.4.3). For each planner, we check the reachability of all planned grasps and evaluate the fraction of planned grasps that are reachable after running the planner for a given number of steps.

The results (Figure 2.4) show the significance of our reachability-aware grasp planning approach compared with a typical grasp planner on the same setup. Given the same amount of time, the planners each return twenty grasp solutions and we check what fraction of them are achievable by the robot and report the average over all 36 simulated object poses (4 objects, 9 poses). As expected, a typical grasp planner that has no notion of reachability (shown in red) will yield a large percentage of grasps that are not feasible for the robot. Even with additional planning time, there is no improvement in the odds of returning reachable grasps. On the other hand, our method (shown in blue) yields significantly higher fraction of reachable grasps ($>$ 25% improvement). In addition, the fraction of reachable grasps increases quickly with planning time as our method optimizes for both stability



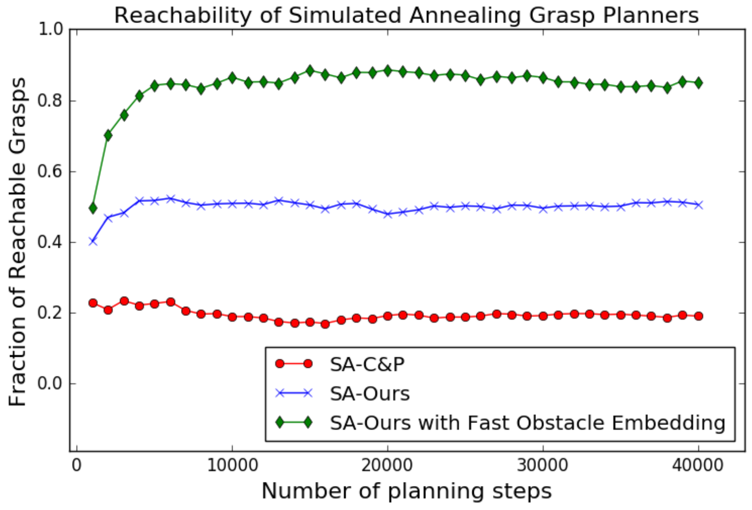


Figure 2.4: Mean fraction of reachable grasps using different energy functions for varying planning duration. The red plot shows that using pure grasp planning with no notion of reachability gives grasp results that have a low chance of being reachable (48.2% for Fetch and 18.8% for the Staubli-Barrett). The blue plot shows that our reachability aware grasp planner results in a higher fraction ($>$ 25% increase) of reachable grasps for both robots. The green plot shows the additional gain obtained when we embed obstacles into the SDF reachability space. This results in reachable grasps that have feasible IK results and do not collide with the obstacles hence an increase in the overall fraction of reachable grasps.

and reachability of grasps. While the blue plot uses the original offline SDF, the green plot shows that by embedding the runtime detected obstacles into the precomputed SDF, we are able to increase the percentage of reachable grasps (Fetch: 95.7%, Staubli-Barrett: 86.1%) compared to (84.3%, 50.2%) respectively when the precomputed SDF only avoids self collision and collision with workspace fixtures such as the walls.

After demonstrating that our method increases the feasibility of grasp planning results, we also observe that this method also leads to a significant speedup of the grasp planning process since the reachability energy term guides the annealing process quickly to reachable regions hence reducing the search space. Once GraspIt! returns a list of grasps, we iterate through the list in order of grasp quality and attempt to plan a valid path for each grasps. When a valid grasp is found, we use the Klampt! simulator to get the lift success. For a simple parallel jaw gripper like the Fetch gripper, there was no significant difference in the percentage of lift success (**SA-C&P**: **0.95** and **SA-Ours**: 0.93) but our method requires less number of motion planning attempts (**SA-C&P**: 2.75 and **SA-Ours**: **1.10**) to find a valid grasp in the list and overall returns a higher percentage of useful grasps (reachable and lift success). Our grasp energy formulation ensures that a top ranked grasp has a high chance of being reachable. The difference in grasp quality was more pronounced with the 4 DOF Barrett hand. Our method not only requires less number of motion planning attempts (**SA-C&P**: 6.2 and **SA-Ours**: **1.27**), it also achieves 20% higher lift success rate (**SA-C&P**: 0.75 and **SA-Ours**: **0.95**). This is because our planner spends most of it's time refining grasps in the much-reduced reachable regions.

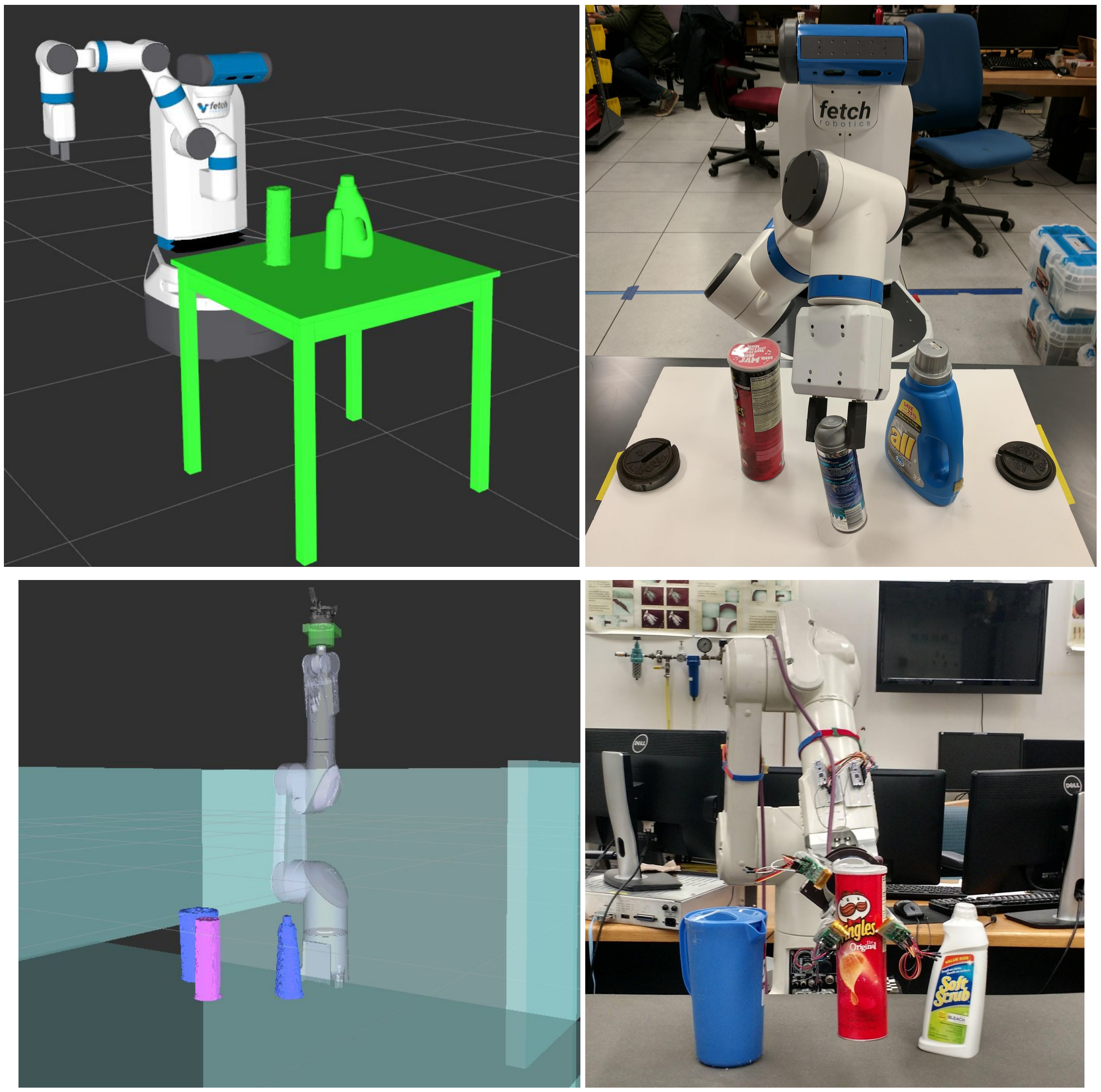


Figure 2.5: Crowded scene for real world experiments. **Top** (Fetch Robot): Our Reachability-Aware planner (SA-Ours) was able to successfully grasp the shaving cream bottle 3/3 times, while annealing without the reachability space (SA-C&P) failed 2/3 times. **Bottom** (Staubli-Barrett Robot): SA-Ours successfully grasped the pringles bottle 5/5 times, while SA-C&P failed 3/5 times.

Table 2.1: Grasp success results on real robot (Fetch) with a crowded scene. Each method was given 3 attempts to plan and execute a grasp on the shaving cream bottle.

Crowded Scene Grasp Planning (Fetch)			
Search Energy	# steps	Success Rate	Mean Grasp Planning Time (s)
SA-Ours	10K	**100.0%**	8.706
SA-C&P	10K	33.3%	8.360
SA-C&P	40K	66.6%	32.31

Table 2.2: Grasp success results on real Staubli-Barrett robot with a crowded scene (Figure 2.5). Each method is given 5 attempts to plan and execute a grasp on the pringles bottle.

Crowded Scene Grasp Planning (Barrett)			
Search Energy	# steps	Success Rate	Mean Grasp Planning Time (s)
SA-Ours	10K	**100.0%**	11.36
SA-C&P	10K	40%	9.53
SA-C&P	40K	60%	31.98

2.5.3 Real Robot Crowded Scene Experiment

To verify our planner and demonstrate that it works outside of simulation, the simulated annealing based grasp planner is run with a variable number of annealing steps and the success rate is reported in Table 2.1 and 2.2. This experiment compares two grasp energy formulations: **SA-Ours** and **SA-C&P**.

Multiple objects are placed in the planning scene as shown in Figure 2.5. The additional objects reduce the range of possible grasps since the obstacles make many grasp poses infeasible. Note that both methods are aware of the obstacles during grasp planning and avoid grasps that put the hand in collision with obstacles. Table 2.1 shows that our method, **SA-Ours**, is able to achieve grasp success – with the Fetch robot picking the object three times out of three within the limited planning time. Conversely, **SA-C&P** fails to produce a reachable grasp 2/3 times when run for 10,000 steps, and fails once even when it is allowed to run for 40,000 steps. Despite being allowed to run for a long duration, the naive planner spends much of its time exploring the back half of the bottle which is completely unreachable to the Fetch robot.

We repeat the same experiment with Staubli-Barrett robot (bottom of Figure 2.5). Though the objects are placed roughly at the centre of the workspace, the presence of walls and distractor objects significantly limit the range of feasible grasps for the target object. Each method has 5 trials and we observe that the results follow the same pattern: **SA-Ours** achieves grasp success all five times within the limited planning time (10,000 steps) while **SA-C&P** fails 2/5 times even when run for 40,000 steps.

2.6 Summary

This chapter provides a framework for a workspace aware grasp planner. This planner greatly improves performance over standard online grasp planning algorithms because of its ability to incorporate a notion of reachability into the online grasp planning process. This improvement is accomplished by leveraging a large precomputed database of over 675,840 unique end-effector poses which have been tested for reachability. At runtime, our grasp planner uses this database to bias the hand towards reachable end effector configurations. This bias allows the grasp planner to generate grasps where a significantly higher percentage of grasps are reachable, a higher percentage result in successful grasp executions, and the planning time required is reduced. It has been experimentally tested in both simulated and physical environments with different arm/gripper combinations.

Several future research directions include: utilizing our computed reachability workspace to help mobile robots navigate to optimal locations for manipulating objects and improvements for speeding up the process of incorporating dynamic objects into our notion of reachability to further assist the grasp planner.

Chapter 3

Motion-Aware Reaching and Grasping of Moving Objects in Cluttered Environments

In the previous chapter, we introduced a novel approach of reasoning about reachability of robot arms and showed how reachability awareness can effectively expand the useful workspace of robots. The experiments show that our approach enables robotic grasping in difficult-to-reach regions in static settings.

In this chapter, we extend to notion of reachability awareness to grasping moving objects and analyse other components required for dynamic grasping in a unified framework. [1] Our framework is aware of the arm's reachability and the object's motion. Specifically, we model the reachability space of the robot using a signed distance field and quickly screen unreachable grasps. Also, we train a neural network to predict the grasp quality conditioned on the current motion of the target. Using these as ranking functions, we

[1] This is based on joint work: *Dynamic Grasping with Reachability and Motion Awareness*, Iretiayo Akinola*, Jingxi Xu*, Shuran Song, and Peter K. Allen[2]

quickly filter a large grasp database to a few grasps in real time. In addition, we present a seeding approach for arm motion generation that utilizes solution from previous time step. This quickly generates a new arm trajectory that is close to the previous plan and prevents fluctuating arm motion. For modelling and predicting the object motion, we implement a recurrent neural network (RNN). Our extensive experiments demonstrate the importance of each of these components and we validate our pipeline on a real robot.

3.1 Introduction

Roboticists have made significant progress in developing algorithms and methods for robotic manipulation in static environments. However, robotic manipulation becomes much harder in dynamic environments which is often the case in the real world. For example, in dynamic grasping, ball catching, human-robot handover, etc., the targets and obstacles to be interacted with might be moving with an unknown motion. Providing robots with the ability to manipulate objects in dynamic environments, despite being less explored, can be extremely important in realizing automation in both industry and daily life. Figure 3.1 illustrates a conveyor belt setting; an ability to pick up the target object without pausing the conveyor belt or knowing the speed of the target object a priori can improve the overall efficiency of the system.

There are many challenges brought by dynamic environments. First, continuous changes in the environments require online and fast motion replanning. Sampling-based methods (RRT, PRM, etc.) are not well-suited for this requirement because the randomness of solutions leads to jerky and wavy motion due to the replanning at each time step. Optimization-based methods (CHOMP, STOMP, etc.) can be time-consuming in highly cluttered scenes, making fast replanning in dynamic environments extremely difficult. Second, most works

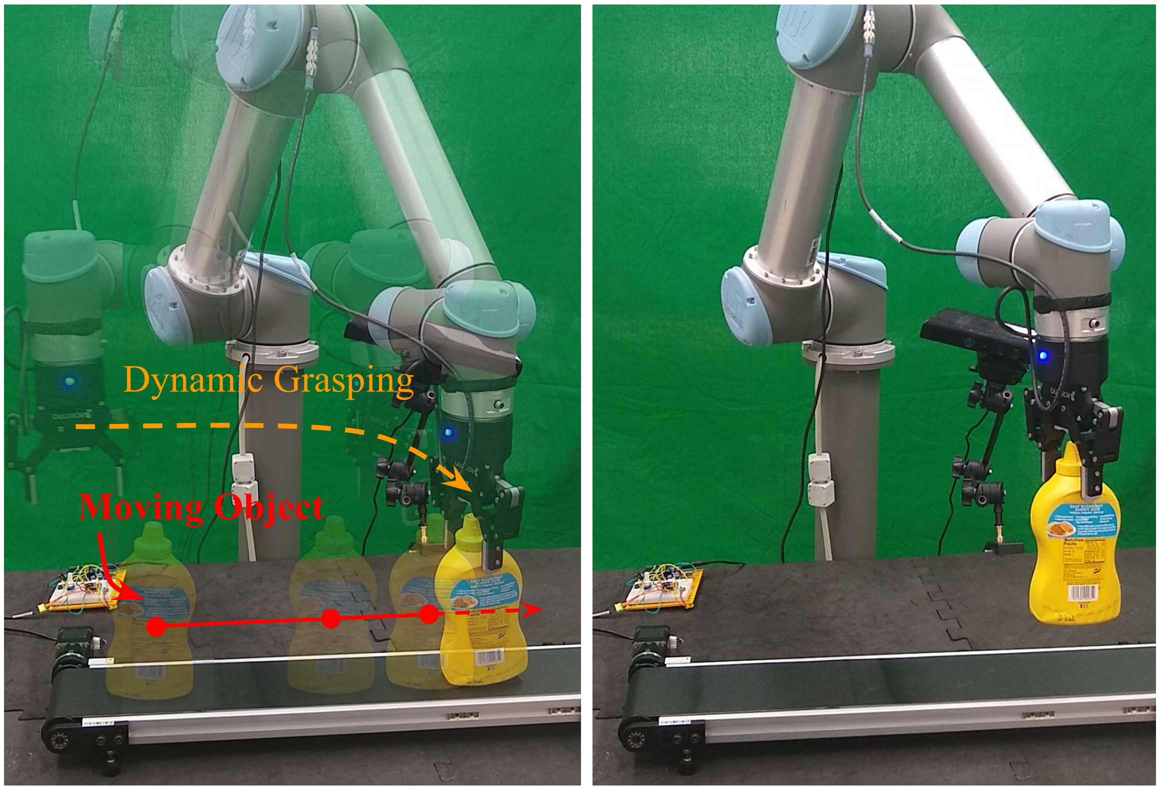
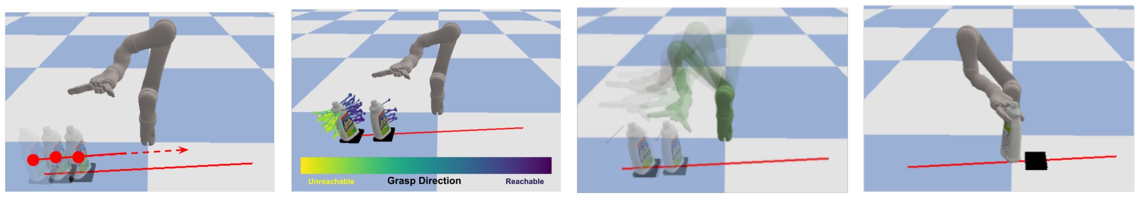


Figure 3.1: Dynamic Grasping Problem: A moving target object is to be grasped and lifted. The object pose and motion is not known a priori and has to be estimated online. Full degree-of-freedom grasps should be explored to come up with feasible grasps that can pick-up the object before it escapes the robot's workspace.

in the grasp planning literature rarely consider the approach and close motion of the grasp, which makes a difference for a moving target. For example, a grasp facing the moving direction of a target can have a higher success rate than a grasp catching the target from the back. Third, we need to understand and predict the motion of the object because computed plans are obsolete when executed.

Previous works have addressed dynamic grasping by introducing a number of assumptions such as prior knowledge of the object motion [5], waiting for the object to come to rest before grasping, limiting the grasping directions to a single direction (e.g. only top-down grasps [119]). In this work, we relax some of these assumptions and tackle the problem of



(a) Motion Prediction (b) Grasp Planning (c) Motion Generation (d) Grasp Execution

Figure 3.2: Dynamic Grasping Framework. **a)** Instantaneous pose estimation runs continuously to keep track of the moving object and we use a recurrent neural network to model the motion of the target object and predict its future pose. **b)** Full grasp database are ranked and filtered based on reachability. **c)** Pick the grasp from filtered list that is closest to the current arm configuration. Arm trajectory is generated based on the future pose of the moving object. Arm trajectory from previous time step is used to seed the planner in current step. **d)** Approach and grasp are executed when CanGrasp condition is satisfied.

robotic grasping for moving objects with no prior knowledge of the object's motion profile and no restrictions on the possible grasping directions of the object. The increase in the range of possible grasping directions has the advantage of expanding the workspace of the robot leading to more grasp options that can be very useful in the dynamic setting. However, as the range of feasible grasp options grows, so does the range of infeasible ones. Without a notion of reachability, it is usually preemptively time-consuming to compute IKs for all the grasps in the database. Our method, illustrated in Figure 3.2, embraces the advantage of an expanded workspace for full degree-of-freedom (DOF) grasps and mitigates the reachability problem by constraining the grasp selection process to the more reachable and manipulable regions of the workspace. In addition, we observe that the robustness of a grasp may vary depending on the speed and direction of the moving object. To handle this, we learn a function that predicts the robustness of grasps given the motion of the object. This is used to rank and select a robust reachable grasp. To generate arm motion, rather than planning from scratch each time, we seed the planning process by the solution from the previous time step. This method allows the newly planned trajectory to be similar and also speed up the computation. In summary, our main contributions are:

- **Reachability and motion-aware grasp planning** ranking functions that predict the reachability and success probability for different grasps based on target pose and motion of the target. These ranking functions are used for real-time grasp filtering.
- **Adaptive motion generation** an effective trajectory generation approach that incorporates the solution from previous timestep to seed the search process to achieve quicker and smoother transition between different motion plans.
- **Simulation and real robot evaluation** procedure of systematically evaluating dynamic grasping performance in simulation with randomized linear / nonlinear motion with different objects, and a real robot demonstration to pick up objects moving on a conveyor belt.

3.2 Related Works

3.2.1 Grasping in Dynamic Environments

Grasping of static objects can be achieved via visual servoing [58] [102] [57] [38] [113] [101] [71] [117]; however, grasping in a changing environment presents a unique challenge. The robot not only tracks the object but also has to reason about the geometry of the object to determine how to pick it up. Some learning-based grasping systems [77] have been applied to slightly moving scenes. Previous works demonstrate grasping of a static object in the midst of moving obstacles [55]. Our work deals with picking a moving object while avoiding collision with static obstacles.

3.2.2 Database-based Robotic Grasping

Previous works [32] [31] have looked at the idea of grasping using a precomputed database. Most of these methods sample different grasps and evaluate them in simulation using a geometry-based metric. These metrics use static analysis which does not account for the dynamics of the approach and lift process. Some other methods [64] [83] generate grasp database using a real robot which can be valuable but such data is very expensive to collect. A recent concurrent work [26] used this technique to examine different approaches for sampling grasps when generating a grasp database and evaluated their coverage of possible grasping directions. They very densely sample "billions" of grasping direction and measure the robustness of each grasp candidate using the success rate of it's neighbours. [112] collects grasps with randomly added perturbations on the object poses.

3.2.3 Object Tracking

Visual feedback is crucial to grasping and manipulation in dynamic environments. For a position-based system like ours, the visual input from a camera (color and/or depth) is continuously processed into the pose (position and orientation) of the objects in the environment. Bayesian methods [44] [116] or deep learning techniques [99] [108] can be applied to the input image stream to produce object poses in the camera's frame of reference. The noisy pose results from object pose detection systems can be filtered into more stable values using methods such as Kalman filtering [54]. Since Kalman filtering builds a model for the motion, this model serves as a good predictor for the future pose of the object being tracked.

3.2.4 Motion Generation

When obstacles are present, reaching a moving target requires some trajectory planning (such as RRT [60], PRM [56]) or trajectory optimization methods (such as CHOMP [86], STOMP [52]) that are able to generate collision-free paths for the arm. Recent works [90] [89] presented an approach to generate a sequence of constraint-based controllers to reactively execute a plan while respecting specified constraints like collision avoidance. Our work is more similar to works that generate arm motion from a library of stored arm motions [8] [21] [43]. Building on these works, we propose an approach that only keeps the solution from previous time step without a precomputed database of arm motions. This previous solution is used to initialize tree/roadmap for sampling-based methods or as a seed for trajectory optimization solver.

A recent work [119] presented an approach that uses motion prediction to grasp a moving block using top-down grasps; they illustrated their approach using simulated experiments. Our work differs from [119] in that we do not limit the grasp direction to only top-down direction, we handle different objects and we incorporate a notion of reachability [3] to guide the grasping process. In addition, we demonstrate our method on real hardware. Another recent work [73] looked holistically at the problem of dynamic grasping especially during handover between a human and a real robot. As the object moves, approximate inverse kinematics (IK) are computed on a database of pre-computed grasps and the quality of the IK solutions are computed and ranked. In our work, we compare the IK of filtered grasps to the current robot joint values and pick the closest grasp.

3.3 Problem Definition

The task is for a robot to pick up a moving object whose motion is not known a priori and avoid colliding with the surrounding obstacles. We assume that the models of the objects and obstacles exist so the system can model the environment using object detection and pose estimation. The task is successful if the robot is able to pick up and lift the correct target object without knocking over the surrounding objects/obstacles. We also want target object to be picked up as fast as possible. This task imitates many warehouse conveyor belt scenarios when both the obstacle and target objects are fragile and moveable with unplanned contact.

3.4 Method

In this section we describe the various components of our system (illustrated in Figure 3.2). First, we describe the visual processing unit that detects object poses. We then discuss the predictive component that estimates the future pose of the object. Next, we discuss the online grasp planning component that produces motion-conditioned reachable and stable grasps in real time. Finally, we present our arm motion generation method.

3.4.1 Overview

The overall algorithm is presented in Algorithm 2. Each grasp consists of a grasp pose and a pregrasp pose generated by backing off the grasp pose for distance b. Our pipeline takes in a known object O. It first retrieves a pre-computed database of grasps G_{DB} for the target object; grasps in G_{DB} are all in object frame. In the dynamic grasping loop, it estimates the current pose p_c of the target and predicts a future pose p_f with duration t.

t is defined as a step function of the euclidean distance d from the arm end-effector to the planned pregrasp: $t = 2\text{s}$ if $d > 0.3\text{m}$, $t = 1\text{s}$ if $0.1\text{m} < d \leq 0.3\text{m}$, and $t = 0\text{s}$ if $d \leq 0.1\text{m}$. We convert the grasps in G_{DB} from object frame to robot frame according to predicted pose p_f and then filtered the grasps using the reachability and motion-aware ranking functions described later in Section 3.4.3 and keep the shortlisted top 10 grasps G_F. We pick the grasp g_c from G_F that is closest to the current robot configuration and move the arm if p_c is reachable otherwise we continue to the next loop. We keep executing the algorithm until the condition for executing the grasp is satisfied. Define the euclidean distance between the end-effector position and the planned grasp position to be d_p, and the absolute quaternion distance between the end-effector orientation and planned grasp pose orientation to be d_q, then CanGrasp returns true if $d_p \leq 1.1b$ and $d_q \leq 20^\circ$, where b is the back-off distance.

After the condition is met, we get an updated estimate of the object pose, predict with horizon $t' = 1\text{s}$, and convert g_c using the newly predicted pose p'_f. The arm is then moved to g_c. The hand is closed while moving with the target for another $t'' = 0.1\text{s}$. In our pipeline, t, t', and t'' are configured experimentally but the optimal prediction horizon should be a function of the end-effector speed, the distance between the end-effector and the planned grasp pose, and the motion of the target. We leave this for future research. Finally, we check if the object has been lifted to determine success.

3.4.2 Object Motion Modelling

Picking up moving objects requires instantaneously detecting the relevant objects in the scene. We continuously track/model the motion of the object to be able to handle cases where the motion profile changes with time.

Algorithm 2 Dynamic Grasping Pipeline

```
1: function DynamicGrasp($O$)
2:     $G_{DB} \leftarrow$ RetrieveGraspDatabase($O$)
3:     while True do
4:         $p_c \leftarrow$ DetectPose($O$)
5:         $p_f \leftarrow$ Predict($p_c$, $t$)
6:         $G_W \leftarrow$ ConvertGrasps($G_{DB}$, $p_f$)
7:         $G_F \leftarrow$ FilterGrasps($G_W$, $p_f$)
8:         $g_c \leftarrow$ PickGrasp($G_F$)
9:         Continue to next iteration if $g_c$ is not reachable
10:        Move arm to $g_c$
11:        if CanGrasp() then
12:            $p'_c \leftarrow$ DetectPose($O$)
13:            $p'_f \leftarrow$ Predict($p'_c$, $t'$)
14:            $g_c \leftarrow$ ConvertGrasps($g_c$, $p'_f$)
15:            Move arm to $g_c$
16:            Close hand while moving with the target for $t''$
17:            Break loop
18:    return CheckSuccess()
```

3.4.2.1 Object Detection and Tracking

In real-world experiments of this work, we use a recent learning-based method (DOPE [99]) to get instantaneous poses of moving objects in the scene. DOPE trains a neural network model that takes an RGB image as input and outputs the pose of a target object relative to the camera frame. A different model is trained for each object of interest and each model can detect multiple instances of their target object. Images of the grasping scene are captured using a kinect and passed through the DOPE models to detect objects and obstacles in the scene. To achieve robustness, we use the published model [99] that was trained on data collected in different lighting condition. In simulation, we directly access the pose of the objects and obstacles in the grasping scene.

3.4.2.2 Recursive State Estimation / Object Pose Prediction

Grasp / motion planning has a time cost and a computed grasp / motion plan can become obsolete very quickly as the object moves. As a result, an ability to predict the future pose of the target object can improve the overall success of a dynamic grasping system. The motion prediction ability is needed for both planning a grasp and executing a grasp (the approach and close motion). While Kalman filtering (KF) [54] is a practical approach for linear motion prediction, we adopt a recurrent neural network (RNN) approach to be able to generalize to non-linear motions as well. The RNN continuously takes in a sequence of instantaneous pose measurements $(p_{t-n}, \cdots p_{t-1}, p_t)$ to update it's internal state which is used to predict future pose at different prediction horizon lengths $(p_{t_{f1}}, p_{t_{f2}}, \cdots, p_{t_{fm}})$. To train the RNN [2], we create a dataset contains planar linear, circular and sinusoidal sequence of waypoints (2000 each) randomly generated along different directions with different start points. To aid learning and generalization, each sequence data point is normalized to the start of the sequence i.e. $((0, \cdots, p_{t-1} - p_{t-n}, p_t - p_{t-n}) \rightarrow (p_{t_{f1}} - p_{t-n}, p_{t_{f2 - p_{t-n}}}))$. This RNN approach can also be used to model object motion during human-robot handovers.

3.4.3 Grasp Planning for Moving Objects

3.4.3.1 Grasp Database Generation

To generate grasps for moving objects, we pre-compute a database of grasps for all target objects while they remain static. Similar to [26], this database was collected and evaluated purely in simulation with dynamics turned on. First, we densely generate 5000 stable grasps for each object using a simulated annealling approach [19], and we then evaluate all the grasps in simulation; each grasp is executed to lift the object 50 times and each time we add random noise to the object pose [112]. The success rate gives a measure of the robustness of the grasp and we choose the top 100 robust grasps.

[2]RNN model: LSTM(100), 2$\times$ Dense(100), Dense(output_shape).
output_shape = num_future $\times$ dim

3.4.3.2 Reachability-Aware Grasping

In the dynamic grasping setting, it is important to have a fast way to choose a feasible grasp out of the list of stable and robust grasps. Generating collision-free IK for all the grasps can be time-consuming; instead, we use our pre-computed reachability space to quickly rank the grasps for the given object pose estimate (See [3] for more details). The larger reachability value the grasp has, with higher probability a valid IK can be found for that grasp. Reachability can also be an index of manipulability and it follows the intuition that the most reachable grasp has higher probability to continue being reachable in the future, reducing the number of grasp switches while the target moves around. The interpolation and indexing of a pre-computed 6D space gives a fast way to reduce the grasp database to a few more reachable grasps whose IK can then be found. This approach is much faster that computing IK for the entire database and is important in dynamic settings. We can use this reachability computation as a rank function for FilterGrasp in Algorithm 2. An example using reachability is shown in Figure 3.2.

3.4.3.3 Motion-Aware Grasping

We observe that the success rate of a stable grasp varies depending on the motion of the object. For example, picking up an object from behind as it moves away can result in different success rate statistics compared to approaching in the direction opposite it's motion. To address this, we learn a neural network model $\mathcal{M}(g, pg, v, \theta)$ that predicts the success probability of a grasp g given the motion profile of the object (speed v and motion direction θ). The input into the model includes:

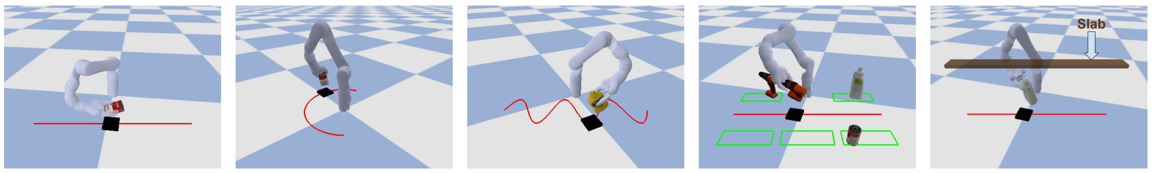
- The 6D grasp pose ($g \in \mathbb{R}^6$) i.e. the $\{x, y, z\}$ position and $\{roll, pitch, yaw\}$ orientation in the object's frame of reference.

- The 6D pre-grasp pose ($pg \in \mathbb{R}^6$) which is the grasp pose backed off (5 cm for Mico hand and 7.5cm for robotiq hand) along the approach direction (i.e. a vector pointing from the end-effector towards the object).
- The speed $v \in \mathbb{R}$ of the object.
- The motion direction $\theta \in [0, 2\pi]$. We assume a 2D planar motion parameterized by a polar angle direction around the z-axis of the object frame.

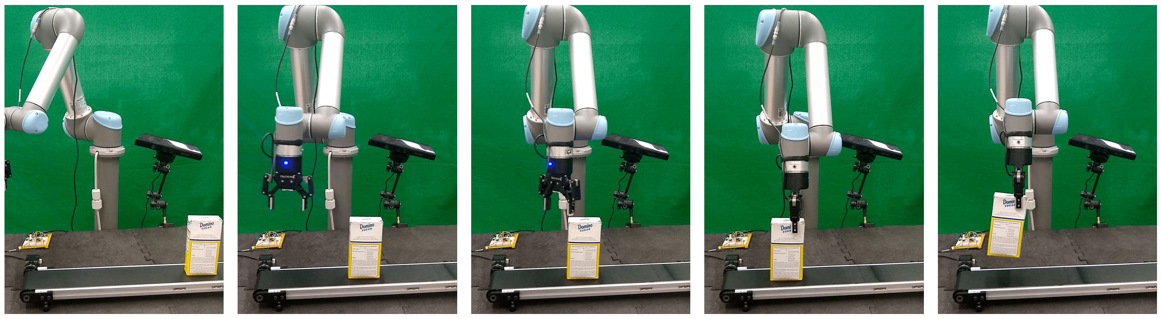
The model has two hidden layers (512 each) and an output predicting the success probability. We generated a dataset of 10000 grasp attempts each on 7 different objects in simulation using the robot's end-effector only. For each grasp attempt, the end-effector starts at the pregrasp pose and moves towards the object while the object moves in a randomly sampled planar direction, at a speed sampled uniformly between 0.5cm/s and 5cm/s. We record the result of the grasp attempts and use this as supervision to train the models (one for each object). We train 100 epochs for each object. The average training time is $\sim$ 5mins and the average validation accuracy is 0.963 with False Positive Rate (FPR) 0.017 and False Negative Rate (FNR) 0.117. Ultimately, the probability of success output by the network can be used as a motion-aware quality conditioned on the object motion. We can use this network to quickly filter grasps that has the highest motion-aware quality for FilterGrasp in Algorithm 2. In general, the motion-aware model prefers grasps facing the moving direction of the target.

3.4.3.4 Combining Reachability and Motion-aware

We want to include grasps in the shortlisted pool G_F that are both reachable and are stable conditioned on the object motion. There are many different ways to combine the reachability



(a) Linear (b) Circular (c) Sinusoidal (d) Obstacles (e) Slab Fixture



(f) Real Robot

Figure 3.3: Dynamic Grasping Tasks. Experimental scenarios for picking up objects on a conveyor belt. The red line shows the conveyor belt trajectory. **(a), (b), (c)** Linear, circular and sinusoidal motion of target object with no surrounding obstacles. **(d)** Linear motion with surrounding static obstacles. Green rectangles are the sub-regions where we sample obstacle locations. **(e)** Linear motion with slab fixture that limits feasible grasping directions. **(f)** Real Robot Demo: Linear motion of target object moving at 4.46 cm/s.

and motion-aware quality for each grasp in the database. We empirically find that simply including the top 5 grasps with highest reachability and the top 5 grasps with highest motion-aware quality outperforms other ways of combination, including the weighted sum of two values or filtering by reachability and then motion-aware quality, etc.

3.4.4 Motion Generation and Grasp Execution

There are three stages of motion generation when picking up an object: reaching, grasping and lifting [74]. In our implementation, we transform the planned grasp to match the predicted future pose of the object and generate arm motion for all three phases.

To be able to generate and update the reaching trajectories as the object moves, we introduce the idea of trajectory seeding that uses the trajectory solution of a previous time step as an initialization for finding a new trajectory at the current time-step. For sampling-based methods like RRT or PRM, this entails initializing the sampling tree or roadmap with the waypoints found from the previous time step. This seeds the search to be quite close to the previous path and empirically helps find new solutions that are not drastically different from the previous solution. An unconstrained sampling based approach can return drastically different trajectories in subsequent trajectories which can be disadvantageous in dynamic settings. Another benefit of our seeding approach is that a good initialization from seeding can reduce the time used to find a valid solution. We implement our seeding approach on CHOMP, RRT and PRM and find that PRM works best for our experimental tasks.

Note that the motion plan is executed once generated interrupting the previous trajectory that was being executed. To ensure that the arm does not slow down as new trajectories are computed and updated, we retime the trajectory solution from the solver so that it blends with the current arm velocities and it moves at fast as possible while also respecting joint limits [10]. We use cartesian control to move the arm during the grasping and lifting stages.

3.5 Experiments

We extensively evaluate the performance of our algorithm picking different target objects in randomized linear / nonlinear motion with / without static obstacles in simulation. We then demonstrate that our method works reliably on a real robot. Videos showing some of the experiments can be found at our project website `http://crlab.cs.columbia.edu/dynamic_grasping`.

3.5.1 Experimental Setup

We create different scenarios illustrated below using the Bullet simulator [22] to evaluate the performance of our methods on two robot arms with parallel jaw grippers: the Kinova Mico and the UR5-Robotiq robots. These two robots have different workspace dimensions, manifolds, joint limits as well as different gripper spans / width. UR5 arm has a wider span and moves quicker than Mico arm, so we intentionally make the tasks for UR5-Robotiq harder. For each robot, we simulate the task of linear and non-linear conveyor belt pickup, which plays a significant role in warehouse packaging and assembly lines. In these scenarios, there is a target object moving on a belt, possibly among surrounding static obstacles. Both the target and the obstacle objects can be fragile and and we cannot knock them over.

Linear Motion: The target object moves linearly at a constant speed (3cm/s for Mico and 5cm/s for UR5-Robotiq) as shown in Figure 3.3a. The conveyor trajectories are randomized using 4 parameters as shown in Figure 3.4. θ specifies the counter clockwise angle of the line perpendicular to the linear motion, connecting the middle point of the motion and the base of the arm. r is the distance between the linear trajectory and the arm base. l is the length of the linear trajectory. $d \in \{+1, -1\}$ indicates the direction of the motion, where $+1$ means moving counter clockwise and -1 means moving clockwise. We set $0 \leq \theta < 2\pi$ (radians), $0.15m \leq r \leq 0.4m$ for Mico and $0.3m \leq r \leq 0.7m$ for UR5-Robotiq, and $l = 1m$.

Linear with Obstacles: We add 3 static obstacles to the linear motion in the grasping scene. Specifically, we divide the near region (distance between 0.15m and 0.25m) surrounding the linear motion into 5 sub-regions, as shown in Figure 3.3d. For each obstacle, we randomly pick a sub-region and uniformly sample a location in the sub-region. We make sure there is no collision between obstacle and robot or between two obstacles. The arm has to avoid hitting both obstacles and the target.

Linear with Top Slab: A 2cm-thick slab of width 10cm is placed 40cm directly on top of the conveyor belt. This limits the grasping directions (e.g. top-down grasps) and makes motion planning/grasping more challenging.

Linear with Z Motion: We relieve the constraint of the linear motion so that the object can also move in the Z axis. The starting height and the end height is randomly sampled between 0.01m and 0.4m.

Linear with Varying Speed: The target is accelerated from 1cm/s to 3cm/s and from 3cm/s to 5cm/s for Mico and UR5-Robotiq respectively.

Circular Motion: A smooth non-linear circular motion as shown in Figure 3.3b. The speed of the conveyor belt is constant (2cm/s for Mico and 3cm/s for UR5-Robotiq). The circular motion trajectory is also randomized by 4 parameters as shown in Figure 3.4. θ controls the angle of the starting position on the circle. r is the radius of the circle. l specifies the length of the motion. $d \in \{+1, -1\}$ indicates the direction of the motion, where $+1$ means moving counter clockwise and -1 means moving clockwise. We set $0 \leq \theta < 2\pi$ (radians), $0.15\text{m} \leq r \leq 0.4\text{m}$ for Mico and $0.3\text{m} \leq r \leq 0.7\text{m}$ for UR5-Robotiq, and $l = 1\text{m}$.

Sinusoidal Motion: This is a more challenging non-linear motion where the object moves along a sinusoidal path as shown in Figure 3.3c. To do this, a sinusoid is super-imposed on the randomly generated linear motion as shown in Figure 3.4. In addition to the parameters of the linear motion (θ, r, l, d), we specify the amplitude A and frequency f of the sinusoid. We set $A = l/8$m and $f = 2\pi/(l/3)$Hz.

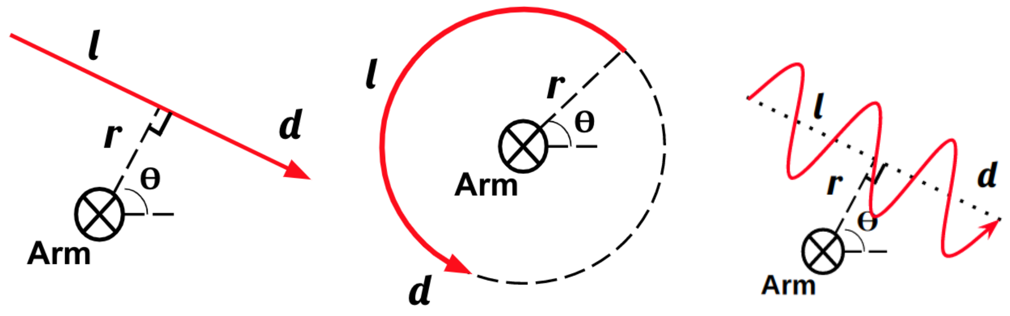


Figure 3.4: A bird's-eye view of randomized linear, circular and sinusoidal conveyor belt motion generation process. A random experiment motion is parameterized by angle θ, distance r, direction d, and length l. The cross indicates the position of the robot base. The red line shows the motion of the conveyor belt, with an arrow indicating the direction. The horizontal dashed line indicates the x-axis of the world frame. Left: linear motion. Middle: circular motion. Right: sinusoidal motion.

Each experiment in simulation is run on 7 different target objects shown in Figure 3.5 whose sizes can physically fit in the robots' hands. The randomized process for generating conveyor belt motion ensures that the results are not biased to a specific robot configuration. For example, a particular starting pose might be close to the robot arm end-effector and will have a higher success rate. For each object, we run each 100 times and report the average success rate and grasping time across 700 trials. In each setting, we compare the performance of the below methods.

- **Ours (R+M).** This is our proposed method that uses all the discussed modules and filters grasp in the grasp database combining both reachability and motion-aware quality as discussed in Section 3.4.3.4.
- **Ours (Reachability).** Same as *Ours (R+M)* except the grasps are filtered with only reachability.

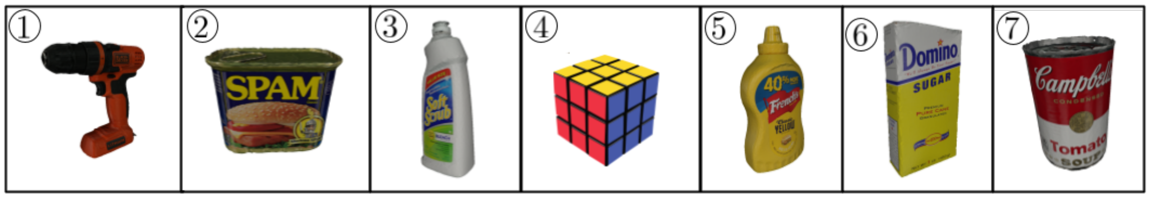


Figure 3.5: Seven objects from the YCB Object Database selected as the graspable objects in our experiments. All seven are used for simulation experiments while the last three are used for the real robot experimentation.

- **Ours (Motion-aware).** Same as *Ours (R+M)* except the grasps are filtered with only motion-aware quality.
- **Baseline.** Using randomly sampled 10 grasps from the grasp database as the filterd grasps. We need to use a subset because checking IK for all grasps from the database during dynamic grasping is unfeasible with very low success rate that is not worth comparing.
- **No Traj. Seeding.** This ablation study picks the model from above with best performance and removes the trajectory seeding module to study its importance.
- **No Prediction.** Similar to *No Traj. Seeding*, this removes the prediction module to study its importance.

We evaluate a subset of the experiments on a real robot hardware to validate our approach. For this, we use the UR5 robot arm fitted with a Robotiq parallel jaw gripper to pick up an object moving on a conveyor belt.

3.5.2 Experimental Results and Discussion

Shown in Table 3.1, our proposed methods with reachability and motion awarenesses or a combination of both outperform the baseline in all cases. The ablation studies demonstrate

Table 3.1: Simulation Experiments for the Kinova Mico (Top) and UR5 (Bottom) robot arms. For each entry, run on 7 objects and 100 trials each. We report success rate, dynamic grasping time (s) averaged over 700 trials.

Methods	Linear (3cm/s)	Linear (3cm/s) with Obstacles	Linear (2cm/s) with Top Slab	Linear (3cm/s) with Z Motion	Linear (1-3cm/s) Varying Speed	Circular (2cm/s)	Sinusoidal (1cm/s)
Ours (R + M)	0.799, 10.23s	**0.796, 11.43s**	**0.781, 21.92s**	**0.814, 10.39s**	**0.869, 9.988s**	**0.875, 10.38s**	**0.895, 8.661s**
Ours (Reachability)	**0.802, 10.21s**	0.795, 10.13s	0.759, 19.91s	0.806, 9.836s	0.836, 9.609s	0.861, 9.421s	0.867, 7.857s
Ours (Motion-aware)	0.722, 15.30s	0.780, 14.37s	0.697, 25.07s	0.710, 14.61s	0.819, 14.45s	0.857, 15.43s	0.827, 13.19s
Baseline	0.439, 23.85s	0.419, 24.15s	0.431, 38.12s	0.430, 23.19s	0.610, 25.72s	0.716, 24.89s	0.643, 20.22s
No Traj. seeding	0.769, 10.61s	0.777, 10.83s	0.706, 22.29s	0.800, 10.41s	0.827, 10.23s	0.857, 10.01s	0.827, 8.588s
No Prediction	0.610, 12.91s	0.614, 13.30s	0.609, 25.84s	0.737, 12.60s	0.761, 11.96s	0.767, 13.11s	0.807, 9.045s

Methods	Linear (5cm/s)	Linear (5cm/s) with Obstacles	Linear (3cm/s) with Top Slab	Linear (5cm/s) with Z Motion	Linear (3-5cm/s) Varying Speed	Circular (3cm/s)	Sinusoidal (1cm/s)
Ours (R + M)	**0.874, 8.134s**	**0.854, 9.104s**	**0.748, 19.86s**	0.858, 8.166s	**0.917, 7.895s**	**0.909, 8.311s**	**0.946, 9.243s**
Ours (Reachability)	0.857, 8.730s	0.841, 9.646s	0.652, 18.52s	**0.872, 8.864s**	0.907, 8.748s	0.890, 9.512s	0.925, 8.608s
Ours (Motion-aware)	0.744, 9.976s	0.752, 9.896s	0.675, 19.86s	0.717, 10.47s	0.854, 8.935s	0.840, 10.68s	0.930, 9.645s
Baseline	0.676, 12.64s	0.576, 13.63s	0.606, 22.89s	0.659, 12.65s	0.788, 12.84s	0.810, 14.09s	0.716, 17.19s
No Traj. seeding	0.849, 9.107s	0.810, 10.22s	0.631, 20.01s	0.836, 9.047s	0.906, 8.859s	0.899, 9.610s	0.904, 11.17s
No Prediction	0.284, 10.31s	0.269, 11.41s	0.457, 20.04s	0.261, 10.89s	0.344, 10.44s	0.594, 9.891s	0.310, 11.96s

the importance of the trajectory seeding and motion prediction components.

3.5.2.1 Effect of Grasp Planning

The results show that reachability and motion awarenesses help extenssively in dynamic grasping tasks. This is because our methods leverage these two ranking functions to quickly filter grasps that are likely to be reachable or stable conditioned on the current motion of the object. They help to focus on those grasps which are near-optimal given the object pose and motion and reduce unnecessary IK calls. Without reachability or motion awareness, even though all grasps in the grasp database are stable in static cases, the selected 10 grasps might not be as optimal given the current location where the object has moved or the current motion the object is following.

We also notice that *Ours (Reachability)* almost always performs better than *Ours (Motion-aware)*. This shows that in dynamic grasping setting, reachability awareness can be a slightly more important factor than motion-aware. Though being stable for the motion of the object, the selected grasps by motion-aware might still be unreachable and waste some IK computing time. We further investigate the relationship between reachabiity performance and the distance of the conveyor motion to the robot base, using linear motion while varying r, as demonstrated in Figure 3.6. We find that reachability being especially beneficial in difficult-to-reach near and far portions of the workspace.

Ours (R + M) outperforms *Ours (Reachability)* in all cases except linear motion for Mico and linear with Z motion for UR5 arm. Combining reachbility and motion awareness include grasps that are both reachable and robust for the current motion. This combines the advantages of both methods and provides a pool of wider variety for PickGrasp to choose from. For example, even though a grasp might not be the most reachable, but it can also be included in the filtered grasps because of high motion-aware quality. For the two cases where *Ours (Reachability)* outperforms *Ours (R+M)*, we believe it is because the most motion-aware grasps happen to have very low reachability but are closer to the current robot configuration. They are picked but cannot remain reachable and result in unnecessary grasp switches.

3.5.2.2 Effect of Seeding in Arm Trajectory Generation

We observe that the value of seeding during trajectory generation becomes significant for tasks when slab fixture is above the conveyor belt limiting the range of motion of the arm (column 3 of Table 3.1, Mico and UR5 show a performance drop of 7.5% and 11.7% respectively), and when the motion is sinusoial and hard to model (column 7 of Table 3.1, Mico and UR5 show a performance drop of 6.8% and 4.2% respectively). Without seeding, computing arm trajectory from scratch at each time step is computationally expensive. Besides, seeding makes the new trajectory solution similar to the previous one. Qualitatively, we noticed that seeding makes the arm motion less wavy given that the arm trajectories

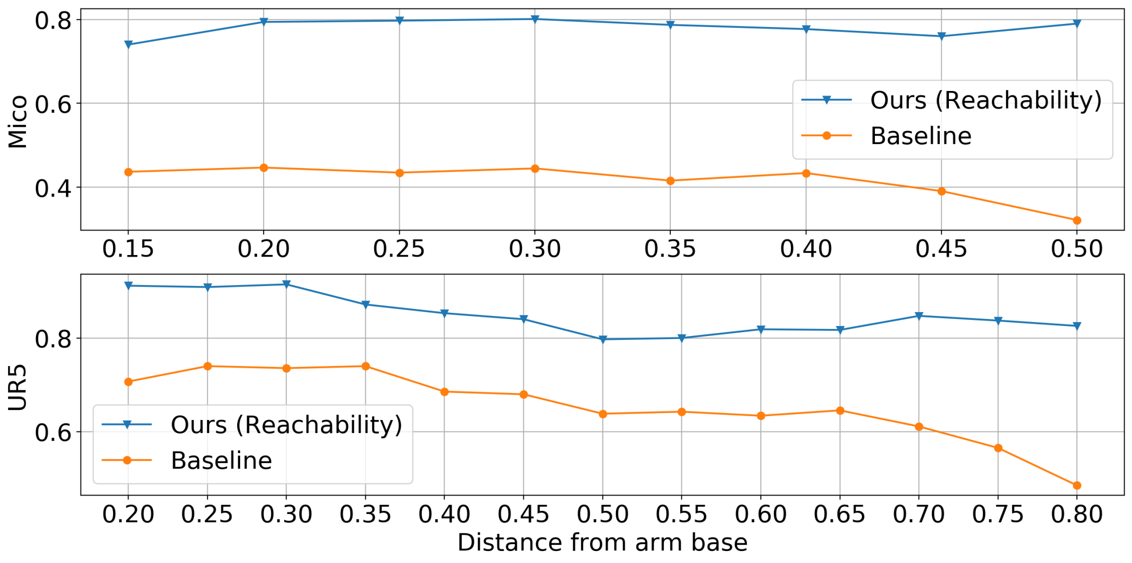


Figure 3.6: Success rate vs. distance. Improvement from reachability awareness becomes more significant when the moving object is extremely close to or far from the robot. This expands the effective workspace of the robot to better handle difficult-to-reach near and far grasp poses.

generated in subsequent time steps is seeded to be similar to the immediate previous one.

3.5.2.3 Effect of Object Motion Prediction

Our results also show that object motion prediction is an important component for dynamic grasping and we see the biggest drop in performance without motion prediction. This is expected as we can image without prediction in dynamic grasping, even with perfect grasp planning and optimal motion generation, the gripper will never catch the target because of the delay from computation. Its importance becomes more pronounced as the object's motion is hard to predict (non-linear / varying speed) and also when the gripper width is smaller. We observe that there is a bigger drop in performance for the UR5-Robotiq robot, compared to the Mico robot. Qualitatively, we observe that a lot of the failure cases occur during the approach-and-grasp phase where the robot finger narrowly knocks off the object. The wider span of the Mico gripper enables it to be more robust in this sense.

3.5.2.4 Real Robot Demonstration

We demonstrate our algorithm on the real robot by picking up objects 5, 6, 7 shown in Figure 3.5 as each object moves on a conveyor belt with no surrounding obstacles. We repeat this experiment 5 times and the success rates for objects 5, 6, 7 are 4/5, 5/5 and 3/5 respectively. Even though the object is moving relatively fast (4.46 cm/s), our method is able to pick the objects 5 and 6 reliably well. The robot is able to align its gripper along the narrow axis of the objects and pick them up while moving. The failure cases for object 7 is because the radius of the tomato can is slightly smaller than the gripper span with a tight margin for error in the approach and grasp stage.

3.6 Conclusion

This chapter presents a novel framework for grasping moving objects using reachability and motion awareness. The framework also includes an RNN-based motion predictive module and an adaptive arm trajectory planner that uses seeding to quickly produce smooth trajectories. We show in experiments with different settings that these elements are important to grasping; grasping systems that do not have motion prediction elements and are do not reason about grasp reachability perform worse in both experimental setups. This work is a model-based visual-pose feedback system. A future work will be an end-to-end image-based analogue where arm-hand trajectory commands are directly generated based on image/depth image features using learning-based techniques.

Chapter 4

Tactile-Aware Multi-fingered Grasping

In the previous two chapters, we showed that incorporating the notion of reachability during grasp planning expands the effective and usable workspace of the robot and is valuable when grasping in both static and dynamic settings. Despite improved grasping generation using vision-based perception, a lot of failure cases in open-loop robotic grasping occur during the grasp execution stage. Inspired by humans' ability to feel and reliably grasp objects, in this chapter, we develop an adaptive tactile grasping system that utilizes tactile sensor readings on the robot's hand to improve reliability of robotic grasping systems[1]. Using model-free deep reinforcement learning, our method obtains a closed-loop grasping behavior that uses additional tactile sensing to adaptively tighten grip on the object– making the grasping system tactile-aware and more reliable.

[1] This work first appeared in CoRL 2019. *MAT - Multi-Fingered Adaptive Tactile Grasping via Deep Reinforcement Learning*, by Bohan Wu, Iretiayo Akinola, Jacob Varley, and Peter Allen[115]

4.1 Multi-Fingered Adaptive Tactile Grasping

As multi-fingered grasping becomes more tractable thanks to advances in vision and deep reinforcement learning (RL), improving state-of-the-art methods that achieve 90%+ grasp success rates becomes more difficult. Among the few percentages of failed grasps are those caused by grasp slip, low friction, calibration error and adversarial object shapes. A promising direction for finishing the last mile of the race towards high-performance, high-success autonomous grasping is the idea of closed-loop grasping: continuously adjusting the robot's DOFs to improve the quality of the current grasp based on sensory feedback. Closed-loop grasping is attractive because it enables the robot to correct the initial grasp to achieve even higher pick-up success rates, given an approximately correct initial grasp pose.

Performing high-quality closed-loop grasping requires a sensor modality that is both free of external disturbances from the robot's ongoing actions and accurate in providing information about the state of the current grasp. Vision, RGB or RGB-D, becomes a less favorable candidate in this case due to visual occlusion. As the robot's end-effector approaches the graspable object, camera vision will be blocked by either the end-effector palm or fingers. Therefore, it is difficult to enable vision to provide undisturbed and accurate information about the status of the current grasp.

Tactile, in this case, is one of the best candidate sensory modalities for closed-loop grasping. Tactile sensors are both rich in information with many sensor cells on each finger (and palm in some cases) and free of external disturbances that visual systems usually face with different levels of occlusion. Figure 4.1 shows a common failure case of an open-loop grasping system due to calibration error. Highlighted in Figure 4.2, this paper introduces Multi-Fingered Adaptive Tactile Grasping, or MAT, a high-performance deep RL algorithm that

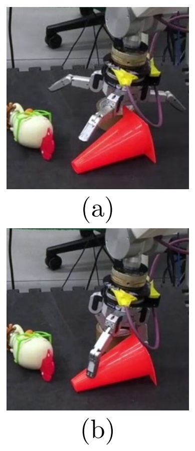


Figure 4.1: **Open-Loop Grasping.** Open-loop grasping **(a)** planned from an initial image of the scene fails to form a stable grasp **(b)**.

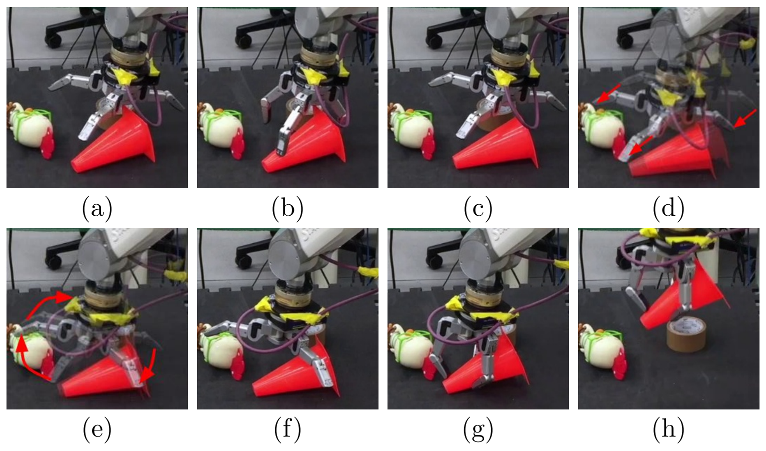


Figure 4.2: **Multi-Fingered Adaptive Tactile Grasping.** Behavior of our policy given **(a)** a coarse initial grasp pose. **(b)** The robot begins by closing the fingers in small increments to form a grasp adapting to tactile contacts as they occur. **(c)** Eventually, unsatisfied with the tactile and proprioceptive observations, the policy decides to reopen the hand and **(d-e)** adjust the end-effector position and orientation. **(f-g)** The policy closes the fingers incrementally again. **(h)** Finally, the policy ends the episode with a lift action and successfully picks the object up.

leverages tactile and proprioceptive information for multi-fingered grasping in an adaptive, closed-loop manner, with the ultimate purpose of substantially improving state-of-the-art open-loop grasping systems. First, MAT allows the robot to learn grasp action primitives in a generative manner via maximum entropy deep RL. These action primitives not only include decisions of granular movements of each of the fingers, lifting the end-effector for pick-up, but also reopening the fingers and adjusting the end-effector position and orientation, thus forming a tight, closed-loop grasping system. Second, since maximum entropy deep RL requires high sample complexity and training experiences that are diverse in terms of object types, quantities, poses, and clutter levels, direct learning or transfer learning in real-world environments becomes a challenge. MAT overcomes this challenge by training in simulation and directly transferring to real without additional learning in a high fidelity way: by choosing observation and action modalities that maintain small sim-to-real gaps to the real world, such as joint angles, binary tactile contacts, tactile contact Cartesian locations, etc. Finally, MAT demonstrates substantially improved pick-up success rates in real-robot experiments over a vision-based, open-loop grasping system. Video of this paper can be found at `http://crlab.cs.columbia.edu/MAT/`. In summary, our contributions are:

1. An adaptive, closed-loop, tactile grasping method that significantly improves single-object and cluttered scene grasp success rates over a strong vision-based open-loop baseline

2. A tactile grasping method that pairs with and improves any open-loop grasping system that generates initial grasp poses. This method **a)** vastly alleviates the need for a perfectly calibrated robot-camera setup and **b)** is robust to severe visual occlusion during grasping as its policy, which is active when the hand approaches the object, does not use vision

3. A method of sim-to-real transfer where an end-to-end tactile grasping policy trained in simulation transfers directly to the real world with high fidelity

4. A curriculum learning approach that learns a tightly closed-loop grasping policy from an initial open-loop policy, by gradually increasing the granularity of finger-close movements

4.2 Related Work

4.2.1 Vision-Based Closed-Loop Grasping

Recent works have focused on closed-loop grasping using vision or other non-touch sensor modalities. [53] provides a promising deep RL approach to learn closed-loop grasping using vision. [77][106][65] use supervised learning approaches to combine visual learning with closed-loop grasping. In some cases, the closed-loop mechanism in these works becomes less effective as the robot approaches the object due to visual occlusion from the arm or the end-effector.

4.2.2 Robotic Grasping with Tactile Only (Blind Grasping without Vision)

In this setting, tactile readings help design manipulation primitives [28][29] or estimate the location and geometry of the objects [49][50]. In [78], the robot makes sweeping motions around the scene to localize the object before attempting grasps roughly at the object centroid. Subsequently, tactile can be used to iteratively adjust the grasp until object pick-up. This approach depends heavily on obtaining a good object localization from touch scanning, which can displace scene objects in ways not detected by the limited tactile coverage, disturbing and negatively affecting the entire grasping process. Consequently, visual-tactile multi-modal methods are becoming more popular.

4.2.3 Improving Vision-Based Grasping using Tactile or Other Contact Force Modalities

Tactile sensors enhance vision-based grasping in a number of ways. Prior to grasping, tactile measurements can help improve geometric knowledge of the grasping scene especially for occluded regions. During grasping, they help evaluate the success likelihood of a grasp being executed [66][23] and decide against lifting the object if the tactile readings indicate a loose grip and an unsuccessful lift. Further, tactile can be used to predict a re-grasping plan that adjusts the current grasp into a more stable grasp pose [66][23][37]. Finally, tactile information can serve as feedback into a closed-loop grasping process that determines how to close robot's fingers given the tactile readings [75].

Leveraging Tactile Information for Shape-Completion Enabled Robotic Grasping. Recent research has focused on using tactile information to perform more accurate shape modeling of target objects in a scene in order to improve grasp success rate using traditional grasping planners [109][110][9][41]. However, these works are still open-loop.

Predicting Grasp Success and Stability from Tactile or Proprioceptive Information. Learning methods can predict the grasp stability [39][40] or success probability [66][23][24][95][120] [20][61][12][69][68] given tactile or proprioceptive data received from the robot hand. These readings are recorded after grasping an object, then the object is lifted to obtain a *grasp success or stability* label. The dataset obtained is used to train a grasp critic. In contrast, MAT does not require a critic but learns a unified end-to-end grasping policy.

Learning to Regrasp using Tactile Information. [15] successfully learns regrasping behaviors using a multi-fingered hand as well as a grasp success predictor for predicting grasping outcomes. [11] uses Gelsight tactile sensors to learn a state-action value function to predict grasping success and selects relatively good grasping actions from a set of randomly sampled candidates. In contrast to these approaches, our approach does not trigger reopening behaviors using a grasp success predictor but instead learns the task of tactile-enabled closed-loop grasping in a coherent framework, enabling the robot to reopen fingers at any point during grasping. Our algorithm performs the grasping control at a more granular resolution– moving each finger separately and in small increments. In addition, while [15][11] require real-world data which can be expensive in time and effort, our tactile grasping policy was trained purely in simulation and transfers directly to the real robot.

Reinforcement-Learning to Grasp Using Tactile and Contact Forces. In [75] contact forces are leveraged to improve multi-fingered grasping success and stability. However, their work focuses on simulation results without extending into the real world. [15] presented an RL approach to obtain reopening behaviors on a real robot for picking up a cylindrical object. However, they used a linear function approximator to represent the policy and noted that a more complex policy class can improve robustness especially when dealing with a wider variety of objects. Like [15], MAT learns to iteratively generate improved grasp poses when needed. In contrast to [15], MAT is active throughout the entire finger closing process to quickly adapt the current grasp to tactile sensory data as contact occurs, and handle various failure cases during grasping. Also, MAT was demonstrated in both real-world single-object and cluttered scenes.

4.3 Preliminaries

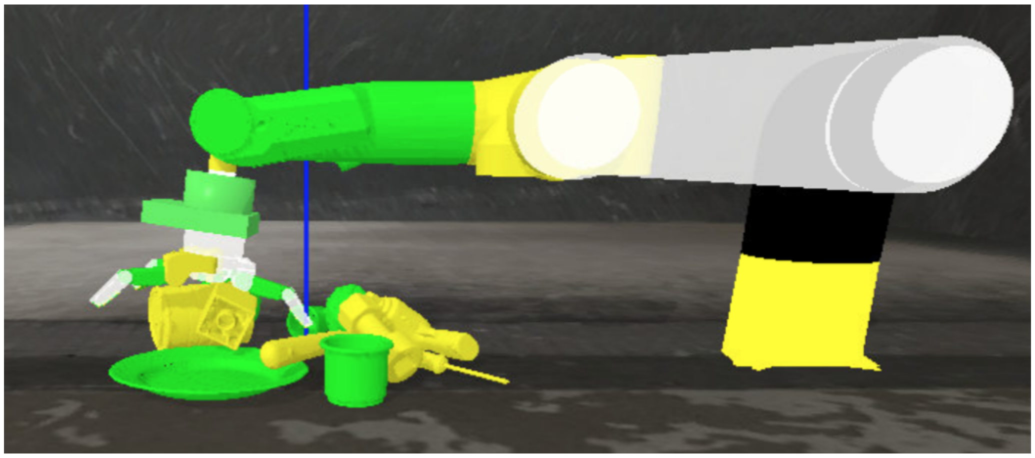
RL Formulation for MAT. In MAT's RL formulation, a tactile grasping robotic *agent* interacts with an *environment* to maximize the expected reward [97]. The environment is a Partially Observable Markov Decision Process (POMDP), since the agent cannot observe any visual information. Even with vision, this environment is still a POMDP since the agent can encounter visual occlusion and cannot observe the complete 3D geometry of any object or the entire scene. To foster good generalization and transfer, MAT models this environment as an MDP defined by $\langle \mathcal{S}, \rho_0, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma, H \rangle$ with an observation space

$\mathcal{S}$, an initial state distribution $\rho_0 \in \Pi(\mathcal{S})$, an action space $\mathcal{A}$, a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, a dynamics model $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$, a discount factor $\gamma \in [0, 1)$, and a finite horizon H. $\Pi(\cdot)$ defines a probability distribution over a set. The agent acts according to stationary stochastic policy $\pi : \mathcal{S} \rightarrow \Pi(\mathcal{A})$, which specifies action choice probabilities for each observation. Each policy π has a corresponding $Q_\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ state-action value function that defines the expected discounted cumulative reward for taking an action a from observation s and following π from that point onward.

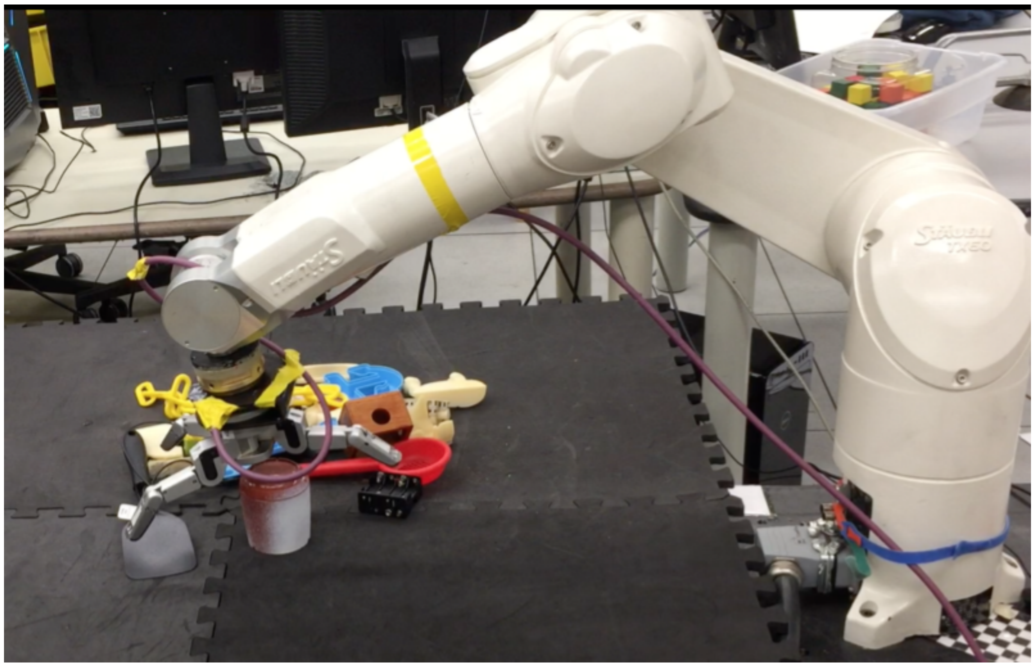
Hardware and Simulation Setup. In both PyBullet (Figure 4.3a) [22] simulation and real-world (Figure 4.3b), we use the Staubli-TX60 Arm and the Barrett Hand (BH-282), which has 24 capacitive tactile cells on each of the three fingers and the palm, totaling 96 cells. Hereafter, we use n to refer to the number of fingers the robotic hand has, and t_{final} to denote the last timestep of the episode. The finger joint angles of the Barrett Hand range from 0 rad (open) to 2.44 rad (close).

4.4 Multi-Fingered Adaptive Tactile Grasping

MAT models the task of closed-loop tactile grasping as a finite-horizon MDP. During each episode, the robot makes a single pick-up attempt on the scene. To begin, we assume that a grasping system of any kind generates an end-effector grasp pose for a cluttered scene, after which visual information is assumed to be unavailable as the arm occludes the scene while realizing the grasp. Next, the robot collects a set of observations, including tactile contacts, finger joint angles, and Cartesian positions for all tactile contacts, as elaborated in Section 4.4.1. Given such observations, the robot either 1) adjusts the joint angle of one or more fingers; 2) issues a "reopen" maneuver by reopening all fingers and adjusting the end-effector position and orientation; or 3) makes an attempt to lift the object, after



(a) Simulation Scene (Train)



(b) Real Scene (Test)

Figure 4.3: Setup. **(a)** A tactile-enabled Barrett Hand in PyBullet simulation for training. **(b)** Real robot for evaluation.

which the episode terminates. In the case of 1) or 2), the episode moves forward to the next time step, and a new set of tactile observations are collected. Details of these three types of action primitives are elaborated in Section 4.4.2. Through a simple reward structure (Section 4.4.3), the robot receives a higher reward for successfully picking an object up and a lower reward otherwise. Parameterized by a multi-modal network architecture and optimized through a soft surrogate objective (Section 4.4.4) and curriculum learning (Section 4.4.5), the robot gradually learns to perform a series of actions that ultimately leads to a higher grasp success rate.

4.4.1 Observation Space

The robot's observation includes recent history and delta values (Δ) of tactile contacts, finger joint angles, and Cartesian positions for all tactile contacts:

$$s_t = \{s_t^{contacts\_binary}, s_t^{\Delta contacts\_binary}, s_t^{joint\_angles}, s_t^{\Delta joint\_angles}, s_t^{contacts\_xyz}, s_t^{\Delta contacts\_xyz}\}.$$

Tactile Contacts. The robot can observe the history of tactile contacts over the last 20 timesteps, which are binary indicators of whether each of the 96 tactile cells is activated or not: $s_t^{contacts\_binary} \in \{0,1\}^{20\times 96}$, as well as the delta in binary values between adjacent timesteps (19 delta values for each cell): $s_t^{\Delta contacts\_binary} \in \{-1,0,1\}^{19\times 96}$.

Note that this binary contact aggregation is important for stabilizing real-world raw tactile readings. Tactile sensors on the physical Barrett BH-282 Hand stream tactile readings at a frequency of 246 Hz. These tactile readings provide the magnitude of force felt by each of the 96 tactile cells, which range from 0 to 20. To generate stable tactile readings for each cell, we calculate the running mean of the last 50 readings for each cell. Tactile readings in PyBullet simulation are stable so no averaging is required. To obtain binary tactile contacts ($s_t^{contacts\_binary} \in \{0,1\}^{20\times 96}$) from the raw running average of the tactile readings, we use a threshold of 0.8 for both the physical and the simulated hand. Values above this threshold indicate contact.

Finger Joint Angles. The robot can observe the history of all joint angles of the hand over the last 20 timesteps, as well as whether the delta in values between adjacent timesteps exceeds a small threshold $\delta_{joint\_angle\_threshold} = 0.05$ rad. Since there are 8 joints for the Barrett Hand: $s_t^{joint\_angles} \in \mathbb{R}^{20\times 8}, s_t^{\Delta joint\_angles} \in \{0,1\}^{19\times 8}$.

Tactile Contact Cartesian Positions. The robot can also observe the history of the $[x, y, z]$ Cartesian positions of all positive tactile contacts of the hand over the last 20 timesteps: $s_t^{contacts\_xyz} \in \mathbb{R}^{20\times 96\times 3}$, as well as the delta in values between adjacent timesteps: $s_t^{\Delta contacts\_xyz} \in \mathbb{R}^{19\times 96\times 3}$. The Cartesian positions are obtained via forward kinematics and expressed in the end-effector frame. If the tactile contact is not positive, the Cartesian position will be $[0, 0, 0]$ by default.

4.4.2 Action Space

Given the observations elaborated in Section 4.4.1, the robot learns a function approximator f that generates an action comprising of 1) movement of each of the n fingers, 2) decision to reopen all fingers or not, 3) the position and orientation adjustments of the end-effector pose in the case of reopening, and 4) decision to lift the hand for a pick-up attempt or not: $a_t = \{a_t^{finger_1}, a_t^{finger_2}, ..., a_t^{finger_n}, a_t^{reopen}, a_t^{wrist\_rotation}, a_t^{lift}\}$.

Finger Movements. The robot can decide whether to close each finger further or not: $a_t^{finger_i} \in \{0,1\}$, where $i \in [1, n]$. If $a_t^{finger_i} = 1$, the i^{th} finger will close by a small joint angle delta of δ_{finger_angle} (explained in Section 4.4.5). The robot samples each finger movement from an independent Bernoulli distribution given a learned sigmoid-activated parameter:

$$a_t^{finger_i} \sim \text{Bern}(\text{sigmoid}(f^{finger_i}(s_t))) \in \{0, 1\} \tag{4.1}$$

.

Finger Reopening and End-Effector Pose Adjustment. The robot's action can control whether to reopen or not: $a_t^{reopen} \in \{0, 1\}$. The robot samples a_t^{reopen} from a Bernoulli distribution given a sigmoid-activated parameter:

$$a_t^{reopen} \sim \text{Bern}(\text{sigmoid}(f^{reopen}(s_t))) \in \{0, 1\} \tag{4.2}$$

a_t^{reopen} is 1 also when no joints moved above $\delta_{joint\_angle\_threshold} = 0.05$ rad over the past 5 timesteps. If the robot decides to reopen, each finger movement action $a_t^{finger_i}$ is disabled for this timestep.

During a reopen maneuver, all fingers reopen to the pre-grasp joint angles, and the position and orientation of the end-effector adjusts. During position adjustment, the hand's $[x, y]$ coordinates re-locate to above the most recent center of all active finger-palm tactile centers. Each active finger-palm tactile center is the center of all Cartesian locations of the finger-palm's active tactile cells.Concretely, let M be the number of links equipped with tactile cells for a multi-fingered hand. For the BH-282 Barrett Hand, $M = 4$ since all three fingers and the palm have tactile cells. Let C be the total number of tactile cells on each finger or the palm. For the BH-282 Barrett Hand, $C = 24$. Let $T_{m,c}$ denote the binary tactile contact for the c^{th} cell on the m^{th} tactile link, where $m \in [1, M], c \in [1, C]$. Similarly, let $P_{m,c} = [x_{m,c}, y_{m,c}, z_{m,c}]$ be the Cartesian location of the tactile cell expressed in the world frame. Let $\bar{M} = \{m \in [1, M] : \sum_{c=1}^{C} T_{m,c} > 0\}$ be the set of tactile links that have at least one active tactile cell. Let $\bar{C}_m = \{c \in [1, C] : T_{m,c} = 1\}$ be the set of active tactile cells on an active tactile link m, where $m \in \bar{M}$. Let $P_{old} = [x_{old}, y_{old}, z_{old}]$ be the Cartesian location of the end-effector palm after reopening but before position adjustment, expressed in the world frame. During position adjustment, the robot examines the most recent timestep during which at least one tactile cell of the hand was activated. The end-effector palm's new Cartesian location is then calculated as:

$$P_{new} = [x_{new}, y_{new}, z_{new}] \tag{4.3}$$

where

$$x_{new} = \frac{1}{|\bar{M}|} \sum_{m \in \bar{M}} \frac{1}{|\bar{C}_m|} \sum_{c \in \bar{C}_m} x_{m,c} \tag{4.4}$$

$$y_{new} = \frac{1}{|\bar{M}|} \sum_{m \in \bar{M}} \frac{1}{|\bar{C}_m|} \sum_{c \in \bar{C}_m} y_{m,c} \tag{4.5}$$

$$z_{new} = z_{old} \tag{4.6}$$

Intuitively, the hand's $[x, y]$ coordinates re-locate to the $[x, y]$ coordinates of the center of all active finger-palm tactile centers. Each active finger-palm tactile center is the center of all Cartesian locations of the finger or palm's active tactile cells. On the other hand, the z coordinate of the hand stays unchanged. In the case where no tactile cell was activated throughout history: $P_{new} = P_{old}$.

During orientation adjustment, the robot's wrist is rotated by a learned angle to generate a better grasp. This learned angle ranges from $-180°$ to $180°$: $a_t^{wrist\_rotation} \in [-\pi, \pi]$. To generate this angle, the robot samples from a Gaussian distribution whose mean is a learned, tanh-activated parameter and then scales the sampled value by factor π:

$$a_t^{wrist\_rotation} \sim \mathcal{N}(\tanh(f^{wrist\_rotation}(s_t)), \sigma) \times \pi \tag{4.7}$$

$$\in [-\pi, \pi] \tag{4.8}$$

. Here, the Gaussian distribution's standard deviation $\sigma_{rotation}$ is also a learned parameter.

Lifting. The robot's action can control whether to lift the hand for a pick-up attempt or not:

$a_t^{lift} \in \{0, 1\}$. The robot samples a_t^{lift} from a Bernoulli distribution given a sigmoid-activated parameter:

$$a_t^{lift} \sim \text{Bern}(\text{sigmoid}(f^{lift}(s_t))) \tag{4.9}$$

$$\in \{0, 1\} \tag{4.10}$$

. During a lift maneuver, the arm is lifted up vertically by 25cm, and each finger movement action $a_t^{finger_i}$ is disabled. If the robot decides to both reopen and lift, the reopen maneuver takes higher priority and is performed instead of the lift maneuver. If the last timestep of the episode is reached: $t_{final} = H$, lifting automatically occurs.

4.4.3 Reward Structure

After a lift maneuver, the current episode is terminated because the robot has decided to attempt to pick-up an object. At this last timestep t_{final} of the episode, a binary success reward is collected, indicating whether the robot successfully picked an object up: $r_{t_{final}} = \mathbb{1}\{\text{pick-up is successful}\}$. In all timesteps earlier than the last timestep, the reward is zero if the robot decides not to reopen. If the robot decides to reopen, a penalty of -0.05 is given if no fingers closed beyond 0.2 rad: $r_t = -0.05 \times a_t^{reopen} \times (1 - \mathbb{1}\{\max_{i \in grip\_joint\_indices}[s_t^{joint\_angles}]_i > 0.2 \text{ rad}\})$, where $t \in [1, t_{final} - 1]$, which penalizes the robot against reopening fingers too frequently.

4.4.4 Soft Proximal Policy Optimization

Let θ be the parameter weights of the policy network and π_θ be the policy the robot is trying to learn: $\pi_\theta : \mathcal{S} \to \Pi(\mathcal{A})$. Shown in Figure 4.4, θ is a multi-input-branch deep neural network, where multiple observation modalities are handled by individual tanh-activated neural networks for feature extraction. The robot's goal is to maximize the cumulative discounted sum of rewards: $\underset{\theta}{\text{maximize}}\ \mathbb{E}_{\pi_\theta}[\sum_t \gamma^{t-1} r_t]$. To begin, we follow the standard policy (SP) optimization objective:

$$\underset{\theta}{\text{maximize}}\ \mathcal{L}^{SP} = \mathbb{E}_{\rho_0, \pi_\theta}[\pi_\theta(a_t \mid s_t) Q_{\pi_\theta}(s_t, a_t)] \tag{4.11}$$

Next, we introduce a baseline (BL) estimator parameterized by ψ for state-value prediction and variance reduction, after which objective (4.11) turns into [91]:

$$\underset{\theta}{\text{maximize}}\ \mathcal{L}^{PG} = \mathbb{E}_{\rho_0, \pi_\theta}[\pi_\theta(a_t \mid s_t) \hat{A}_t] \tag{4.12}$$

where $\hat{A}_t$ is an estimator of the advantage function [7]. We optimize ψ with the following loss:

$$\underset{\psi}{\text{minimize}}\ \mathcal{L}^{BL} = \mathbb{E}\left[\|V_\psi - V_{\pi_\theta}\|^2\right] \tag{4.13}$$

Next, we substitute the action probability $\pi_\theta(a_t \mid s_t)$ with the Clipped Surrogate Objective [93] and apply a soft advantage target to balance between exploration and exploita-

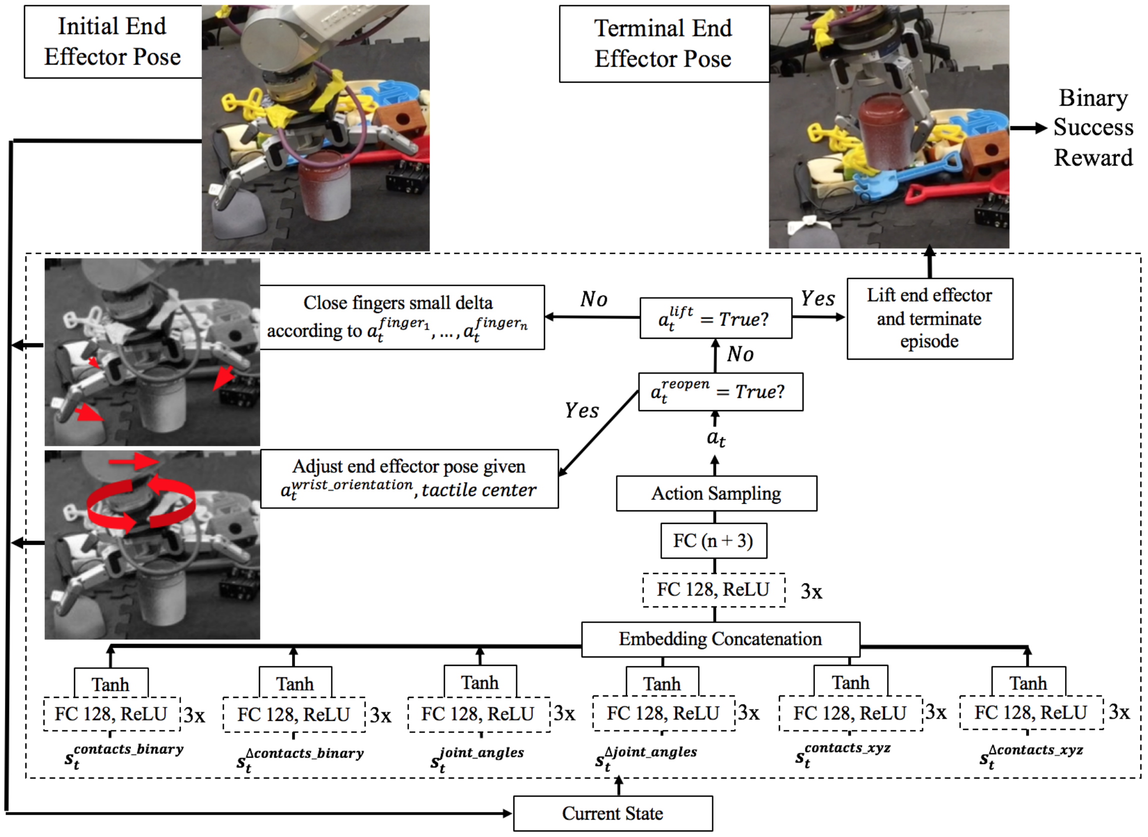

Initial End Effector Pose
Terminal End Effector Pose
Binary Success Reward
Close fingers small delta according to $a_t^{finger_1}, \ldots, a_t^{finger_n}$
No
$a_t^{lift} = True?$
Yes
Lift end effector and terminate episode
No
$a_t^{reopen} = True?$
Yes
a_t
Adjust end effector pose given $a_t^{wrist\_orientation}$, tactile center
Action Sampling
FC (n + 3)
FC 128, ReLU 3x
Embedding Concatenation
Tanh
FC 128, ReLU 3x
$s_t^{contacts\_binary}$
$s_t^{\Delta contacts\_binary}$
$s_t^{joint\_angles}$
$s_t^{\Delta joint\_angles}$
$s_t^{contacts\_xyz}$
$s_t^{\Delta contacts\_xyz}$
Current State

Figure 4.4: **Multi-Fingered Adaptive Tactile Grasping.** Given an initial grasp pose obtained from a vision-based system, for example, the architecture uses soft proximal policy optimization to learn a grasping behavior. The state space consists of tactile contact readings ($s_t^{(\Delta)contacts\_binary}$), contact Cartesian locations ($s_t^{(\Delta)contacts\_xyz}$) and finger joint angles ($s_t^{(\Delta)joint\_angles}$). Using a few deep neural networks, features are extracted from the six components of the state space, squashed to $[-1, 1]$ using *tanh* activations, and finally concatenated into an embedding. This latent embedding is passed through another set of fully connected layers and then outputs a grasp action that specifies how to incrementally close each finger ($a_t^{finger_1}, \ldots a_t^{finger_n}$), whether to lift ($a_t^{lift}$), whether to reopen ($a_t^{reopen}$), and how to adjust the end-effector in case of reopening ($a_t^{wrist\_rotation}$). The top portion of the figure shows how the robot's behavior is determined by a_t. First, it chooses to either continue current grasp or reopen and adjust the grasp. If it continues current grasp, it checks whether to lift, and if not, how to incrementally close the fingers. A binary reward is obtained if lifting results in pick-up success. This reward, along with a small penalty for frequent reopening (Section 4.4.3), is the signal used to train the network in a deep RL manner. "FC m, ReLU" refers to a fully connected layer with output dimension of m followed by a ReLU activation.

tion [34]:

$$\underset{\theta}{\text{maximize}}\ \mathcal{L}^{PG} = \mathbb{E}_{\rho_0,\pi_\theta}[\min(\lambda_t(\theta), \text{clip}(\lambda_t(\theta), 1-\epsilon, 1+\epsilon))(\hat{A}_t - \alpha \log \pi_\theta(a_t \mid s_t))] \quad (4.14)$$

where $\lambda_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. Hyperparameters are detailed in the Appendix section of [115].

As shown in Figure 4.4, at every timestep, the robot decides whether to reopen fingers via a_t^{reopen}. If it does not reopen, a_t^{lift} decides if it is safe to lift the object. If the robot does not reopen or lift, it decides how to close each finger $a_t^{finger_i}$. If the current timestep reaches the finite horizon, lifting automatically occurs and no action component is effective. Therefore, the log action probability is:

$$\begin{aligned}\log \pi_\theta(a_t \mid s_t) = [&\log \pi_\theta(a_t^{reopen} \mid s_t) + (1 - a_t^{reopen}) \times \log \pi_\theta(a_t^{lift} \mid s_t) \\ &+ (1 - a_t^{reopen}) \times (1 - a_t^{lift}) \times \sum_{i=1}^{n} \log \pi_\theta(a_t^{finger_i} \mid s_t)] \times \mathbb{1}\{t_{final} < H\}\end{aligned} \quad (4.15)$$

4.4.5 Curriculum Learning

The finger-closing component of MAT is curriculum-learned. Compared to an open-loop approach that uniformly closes all fingers on the object and lifts after a preset time, MAT incrementally adjusts each finger and decides to lift when certain the object is in hand. An important parameter is the resolution or delta of the finger joint movement δ_{finger_angle}. In the extreme, closing all fingers by a large δ_{finger_angle} reduces to the open-loop policy, however we desire a small δ_{finger_angle} decoupled for each finger to achieve smooth grasping. The challenge is that no reward is received until a lift is attempted and for small δ_{finger_angle} the reward signal will be too sparse. Initially during training, the policy decides to lift almost randomly: $\mathbb{E}[\pi_\theta(a_t^{lift} \mid s_t)] = 0.5$. Since picking an object up requires closing the fingers for many consecutive timesteps before lifting, the initial reward signal is extremely sparse under small δ_{finger_angle}. Empirically, learning becomes difficult when $\delta_{finger\_angle} < 0.4$ rad for the Barrett hand. Conversely, under large δ_{finger_angle}, the policy cannot control finer finger motions and becomes prone to failure.

To get benefits of both fine finger motions and short episode horizons, training in simulation is conducted using curriculum learning around δ_{finger_angle}. Initially, $\delta_{finger\_angle} = 0.4$ rad. Subsequently, the joint angle delta is given by

$$\delta_{finger\_angle} = \delta_{min} + (\delta_{max} - \delta_{min}) \times (1 - current\_max\_success\_rate) \tag{4.16}$$

, where $current\_max\_success\_rate$ is the highest pick-up success rate that the robot has achieved thus far during training. This curriculum learning procedure allows the finger movements to become more and more granular and sophisticated as grasp success rate improves, effectively "closing a tight loop" for tactile grasping.

4.5 Experiments

We train MAT entirely in simulation and test in both simulation and real-world. During training, a single-object or multi-object cluttered scene is loaded with equal probability. We place one object in a single-object scene, a random number of objects from 2 to 30 for a simulated cluttered scene (Figure 4.3a), and 10 objects for a real-world cluttered scene. Leveraging the ShapeNet Repository [13] in simulation, we use 200+ seen objects from the YCB and KIT datasets and 100+ novel objects (not seen during training) from the BigBIRD dataset. We train and test 500 grasp attempts per experiment in simulation. We evaluate real-world single-object performance across 10 trials for each of the 15 seen and novel objects, and real-world cluttered scene performance across 15 scenes.

4.5.1 Results and Discussions

Table 4.1: Experimental Results (% Grasp Success ± Standard-Dev)

	Single Object		Cluttered Scene	
Objects	Seen	Novel	Seen	Novel
	Simulation			
MAT	**98.2 ± 2.1**	**97.4 ± 1.6**	**97.7 ± 2.9**	**95.9 ± 3.9**
Open-Loop Baseline [114]	93.8 ± 2.6	94.9 ± 1.4	92.5 ± 1.8	91.1 ± 3.7
	Real			
MAT	**98.7 ± 3.5**	**98.0 ± 4.1**	**96.4 ± 4.6**	**95.8 ± 4.7**
Open-Loop Baseline [114]	96.7 ± 6.2	93.3 ± 8.1	92.9 ± 5.8	91.9 ± 6.7

Table 4.1 compares 1) a high success rate (above 90%) baseline open-loop vision-based approach [114] with 2) MAT using initial 6-DOF grasp pose provided by [114], and reports the percentage of grasp success and standard deviation across scenes. We fairly evaluate MAT against [114] using the same robot setup, objects, and cluttered scenes. The simulation results show that MAT gives a statistically significant improvement in grasp success rates in all cases: 4.4%, 2.5%, 5.2%, 4.8% for single-seen, single-novel, cluttered-seen, cluttered-novel respectively. On the other hand, the real-world results reveal statistically similar grasp success rates compared to simulation, showing high-fidelity sim-to-real transfer. While the vision-only grasping system already gives high success rates, MAT gives further improvement and is able to avoid or recover from failure cases that the vision-only system cannot handle. For example, MAT uses the tactile readings to finely control how the robot incrementally closes each finger and ensure that it does not lift the object until it is sure it has a firm grip. Also, MAT is able to re-generate a new grasp pose if the tactile readings suggest that the current grasp being executed would not result in a successful pick-up.

4.5.2 Grasping under Calibration Noise

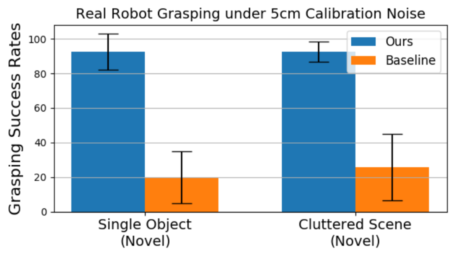
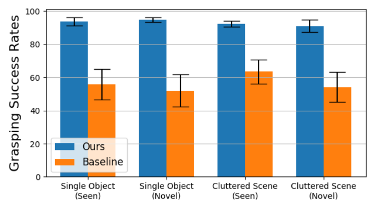


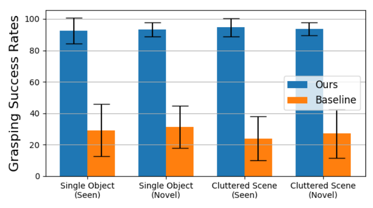
Figure 4.5: Real-World Grasping of Novel Objects with 5cm Y-Axis Calibration Noise

We observe robustness properties of MAT that are valuable if there is calibration error in the grasping setup– a common occurrence in robotics. To show this, we introduce varying levels of calibration noise to the grasping setup and measure how this affects the pick-up success rate. Calibration noise is added as an offset of δcm to the generated 6-DOF grasp pose (obtained from [114]) before grasp execution. On the real robot, we set the calibration noise to be $\delta = 5$cm and run the grasping experiments for the novel objects in both single-object and cluttered scene settings; the results are presented in Figure 4.5. In simulation, we repeat the experiments with $\delta = 2.5, 5, 7.5$cm to analyze the performance across different noise levels as well as to compare between seen and novel objects. Figure 4.6 shows the simulation results.

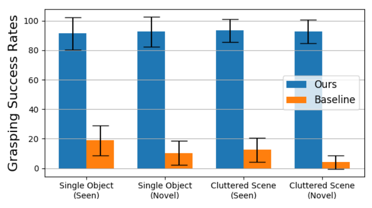
The results show that the tactile policy is significantly more robust to calibration error compared to the vision-only system. On the real robot, the grasp success rate of the vision-only baseline [114] degrades under calibration noise to 20.0% & 25.6% (single-object & cluttered scenes), while our tactile-based policy still achieves 90%+ success in both cases (Figure 4.5). Note that repeated trials do not increase the ability of the vision-only system to recover as the calibration stays an issue for such systems; as a result we terminate each run after 3 consecutive failed attempts per object or scene. For the novel cluttered scenes, the vision-only method picks up only a quarter of the objects present in the scenes and is unable to fully clear any of the scenes. In simulation on the other hand, results in Figure 4.6



(a) Calibration Noise: 2.5 cm



(b) Calibration Noise: 5.0 cm



(c) Calibration Noise: 7.5 cm

Figure 4.6: **Grasping in Simulation under Calibration Noise.** MAT (blue) shows robustness under increasing calibration noise compared to a strong vision-only baseline [114] (orange) which degrades significantly.

reveal a degradation in performance for the vision-based system [114] as calibration noise increases; conversely, the performance of MAT stays robustly high.

4.5.3 Extensive Ablation Experiments

Table 4.2 shows comparison of our MAT algorithm with a tactile baseline and also details extensive ablation studies that measures the value of each component of our MAT algorithm. To the best of our knowledge, MAT is among the first tactile methods that are multi-fingered, free of force-torque usage and compliance, and experimentally tested on cluttered scenes. Although a previous tactile work [48] requires compliance, we re-implemented the non-compliant version of [48] as a tactile baseline comparison, in which each finger stops upon detecting initial contact and then all fingers close in unison after all are in contact. Lifting is subsequently performed after a certain period of time.

Table 4.2: Ablation and Tactile Baseline Results (% Grasp Success ± Standard-Dev)

Objects	Noise	Single Object Seen	Single Object Novel	Cluttered Scene Seen	Cluttered Scene Novel
		Simulation			
MAT	0cm	**98.2 ± 2.1**	**97.4 ± 1.6**	**97.7 ± 2.9**	**95.9 ± 3.9**
Tactile Baseline [48]		94.4 ± 2.3	95.0 ± 1.9	93.3 ± 1.0	91.3 ± 3.1
		Ablation (Simulation)			
Finger-Closing Only	0cm	96.6 ± 2.9	96.2 ± 2.6	95.3 ± 1.1	94.7 ± 3.6
Regrasping Only		96.2 ± 1.8	96.0 ± 1.8	94.4 ± 1.8	93.5 ± 1.6
Position Adjustment Only		96.9 ± 3.8	96.4 ± 4.6	96.2 ± 4.0	94.8 ± 3.5
Orientation Adjustment Only		96.7 ± 3.4	96.4 ± 2.8	95.7 ± 4.6	94.8 ± 1.3
		Simulation			
MAT	2.5cm	**92.5 ± 8.1**	**93.3 ± 7.2**	**94.6 ± 5.8**	**93.7 ± 4.2**
Tactile Baseline [48]		64.1 ± 5.9	68.2 ± 5.6	66.1 ± 5.9	66.8 ± 4.6
		Ablation (Simulation)			
Finger Closing Only	2.5cm	75.2 ± 6.0	73.1 ± 5.3	73.6 ± 5.2	73.9 ± 4.8
Regrasping Only		75.9 ± 2.6	78.5 ± 3.7	76.0 ± 2.0	74.4 ± 2.0
Position Adjustment Only		76.9 ± 5.5	78.5 ± 7.2	80.4 ± 3.5	75.8 ± 7.5
Orientation Adjustment Only		81.2 ± 6.4	81.6 ± 6.4	83.1 ± 4.3	77.8 ± 6.0

Here are some important observations from the Table 4.2:

- By comparing Row "MAT" to Row "Tactile Baseline [48]" for 0cm calibration noise in Table 4.2, we observed statistically significant improvement of MAT's performance over the tactile baseline.

- By comparing Row "MAT" to Row "Tactile Baseline [48]" for 2.5cm calibration noise in Table 4.2, we observed a substantially more significant improvement of MAT's performance over the tactile baseline, mainly due to MAT's ability to adjust the robot's end-effector pose.

- By comparing Row "MAT" to Row "Finger Closing Only" for 0cm calibration noise in Table 4.2, we observe that only using finger closing degrades MAT success rate in simulation by 1.6%, 1.2%, 2.4%, 1.2% for single-seen, single-novel, cluttered-seen, cluttered-novel.

By comparing Row "MAT" to Row "Finger Closing Only" for 2.5cm calibration noise in Table 4.2, we see that the performance degradations are much higher: 17.3%, 20.2%, 21.0%, 19.8% respectively.

- By comparing Row "MAT" to Row "Regrasping Only" for 0cm calibration noise in Table 4.2, we observe that only using regrasping degrades MAT success rate in simulation by 2.0%, 1.4%, 3.3%, 2.4% for single-seen, single-novel, cluttered-seen, cluttered-novel.

  By comparing Row "MAT" to Row "Regrasping Only" for 2.5cm calibration noise in Table 4.2, we see that the performance degradations are much higher: 16.6%, 14.8%, 18.6%, 19.3% respectively.

- By comparing Row "MAT" to Row "Position Adjustment Only" for 0cm calibration noise in Table 4.2, we observe that disabling orientation adjustment degrades MAT success rate in simulation by 1.3%, 1.0%, 1.5%, 1.1% for single-seen, single-novel, cluttered-seen, cluttered-novel.

  By comparing Row "MAT" to Row "Position Adjustment Only" for 2.5cm calibration noise in Table 4.2, we see that the performance degradations are much higher: 15.6%, 14.8%, 14.2%, 17.9% respectively.

- By comparing Row "MAT" to Row "Orientation Adjustment Only" for 0cm calibration noise in Table 4.2, we observe that disabling position adjustment degrades MAT success rate in simulation by 1.5%, 1.0%, 2.0%, 1.1% for single-seen, single-novel, cluttered-seen, cluttered-novel.

  By comparing Row "MAT" to Row "Orientation Adjustment Only" for 2.5cm calibration noise in Table 4.2, we see that the performance degradations are much higher: 11.3%, 11.7%, 11.5%, 15.9% respectively.

4.6 Summary

In this chapter, we develop MAT, an adaptive, closed-loop tactile algorithm to reinforcement-learn dexterous robotic grasping for multi-fingered hands with noisy tactile readings and no visual information. Entirely trained in simulation, MAT achieves mid-to-high 90% success rates and significantly improves upon open-loop grasping under calibration error. As many vision-based grasping systems assume and require efforts to achieve near-perfect calibration, the robustness of MAT even under considerable calibration error reduces the need for a perfectly calibrated grasping setup. MAT can be easily added to any existing open-loop grasp planner to close the final gap between failed grasps and success.

Chapter 5

3D-Aware Closed-Loop Manipulation via Multiple Camera Views

In the previous chapter, we showed how tactile awareness can effectively address some failure cases during robotic grasping due to calibration error, slippage and occlusion. Using tactile sensing, we obtained a closed loop system that for the final phase of the grasping process.

In this chapter, we develop a vision-based closed-loop approach that is applicable to all stages of robotic manipulation[1]. To achieve this, we develop an end-to-end multi-view approach that learns a closed-loop manipulation algorithm for precise manipulation tasks that integrates inputs from multiple static RGB cameras to overcome self-occlusion and improve three dimensional (3D) spatial understanding.

[1] This work first appeared in ICRA 2020. *Learning Precise 3D Manipulation from Multiple Uncalibrated Cameras*, Iretiayo Akinola, Jacob Varley, and Dmitry Kalashnikov[1]

5.1 Learning Precise 3D Manipulation from Multiple Uncalibrated Cameras

Precise object manipulation remains an active area of robotics research; it finds applications in diverse domains such as manufacturing robotics, warehouse packaging and home-assistant robotics. Until recently, most automated solutions are designed for and deployed to highly instrumented settings where scripted robot actions are repeated to move through predefined set of positions. This approach often requires a highly calibrated setup which can be expensive and time-consuming. Also, they lack robustness needed to handle changes in environment, and configuring such methods for new settings requires significant engineering efforts. Advancements in computer vision have led to superior performances in robotic grasping in dense clutter [53][77][114][122][121] by allowing robotic systems to make use of vision systems for various manipulation tasks in less structured settings.

While there has been recent progress in vision-based robot manipulation such as grasping, other tasks like stacking, insertion and precision kitting, that require precise object manipulation, remain challenging for robotic systems[59]. These sorts of tasks require accurate 3D geometric knowledge of the task environment including object shape and pose, relative distances and orientation between key locations in the scene among others. For example, solving an insertion task requires picking an object using the geometry and pose of the object and sticking it in a hole using the pose of object relative to the hole.

We observe that the majority of the existing reinforcement-learned vision-based robotic manipulation systems employ a single camera to observe the task scene. However, the rich 3D information required for solving precision-based tasks are usually limited from a single camera input. For example, it is usually hard to resolve scale and alignment from a

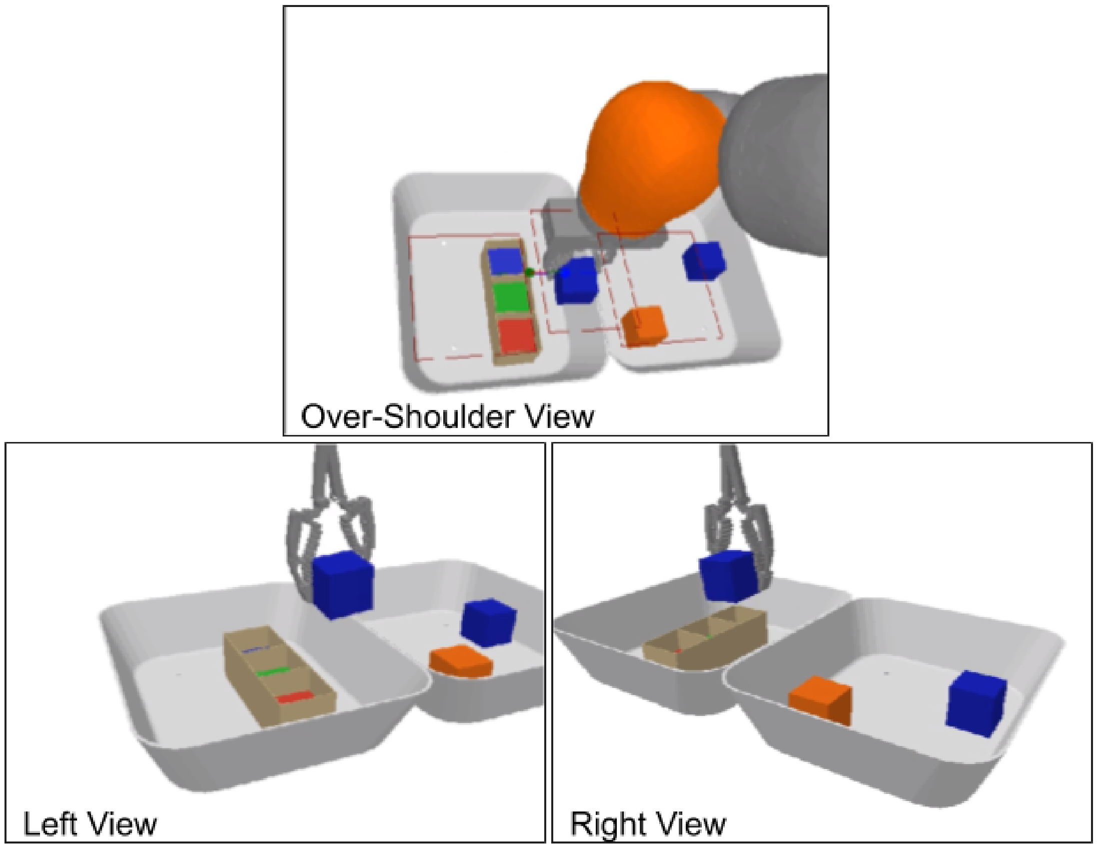


Figure 5.1: **Multi-view Task Learning.** An insertion task where a block is placed into a fixture. This task requires 3D understanding and alignment. A single view system that sees only one of the images would have a difficult time resolving the alignment challenge. Our system combines information from multiple views and achieves better performance on precision-based robotic tasks.

single view. Even for humans, navigating a room or completing a task with one eye closed becomes more challenging from a lack of depth perception. In addition, single view systems are very susceptible to occlusion during task learning requiring the robot to actively move out of the way and reset during task execution. To address these limitations, we propose using multi-view camera setup such as that shown in Figure 5.1 to solve precision-based object manipulation. Since cameras are cheap and ubiquitous, adding a few more cameras to capture multiple views of the task scene is a practical and feasible option.

This research develops techniques for combining multiple camera views to improve the state estimation and increase the robustness of robot action in learning-based robotic manipulation systems. Our approach is a reinforcement learning based method that takes in multiple color (RGB) images from different viewpoints as input and produces robot actions in a closed-loop fashion. The system is trained end-to-end.

Our key contributions include:

- A novel camera calibration free, multi-view, approach to precise 3D robotic manipulation
- An RL model architecture featuring a *sensor dropout* training regime that achieves large reductions to error rates on precision-based tasks (Stacking I: **49.18%**, Stacking II: **56.84%**, Insertion: **64.1%**)
- Analysis of a number of model-free RL architectures for efficiently learning precision based robotics from individual, depth, and multiple views

A video summarizing our approach can be found at `https://www.youtube.com/watch?v=02dhUfTJNK4`.

5.2 Related Work

5.2.1 Precision Robotics Manipulation

Previous works have considered robotic manipulations besides grasping such as stacking [84][79], insertion [98][105][104], screwing and kitting. These tasks require higher levels of precision with a slimmer margin for error, often necessitating extra algorithmic or sensory innovations. Some works focused on developing algorithms, in simulation [79] for precise robot manipulation tasks. Many of these methods leverage important state information such as accurate object poses that are available in simulation but difficult to obtain in the real world. While this approach of using ground truth pose information of relevant entities is useful for developing algorithms for low-dimensional input space, generalizing such systems to the real world requires accurate pose estimation and a well calibrated system. Our approach learns directly from images and does not depend on such calibrated conditions.

In order to use algorithms for low-dimensional input space, some methods use fiducial markers to obtain pose information about objects in the scene [98]. Other methods can be utilized to reason about object geometry [118][103], detect object pose [16][81][67][99][88][6], key points [72] and grasping points [82] from pointcloud and RGB observations after which robot actions are planned and executed to accomplish tasks. The approaches require an estimate of both where a point in the world is relative to a camera, and where the camera is in relation to the robot. Objects that are small, articulated, reflective, or transparent can complicate these methods. In contrast, our method learns precision tasks in an end-to-end fashion without any intermediate pose estimation or camera calibration. Our approach learns hand-eye coordination across multi cameras, without requiring explicit pose information. With objects and the target goal visible from RGB view(s), the robot coordinates ***relative*** displacements from its current pose rather than commanding the arm to absolute coordinates in a fixed reference frame.

Use of extra non-visual sensor modalities such as force-torque and tactile sensors is a common approach to enabling precision-based robotic tasks [51][62][70][45][111]. These methods leverage contact-rich interactions with the environment to reactively achieve the desired manipulation. Our vision-based approach is orthogonal and complementary to the use of non-visual inputs as demonstrated by hybrid methods that combine visual and non-visual sensing [104][63][46].

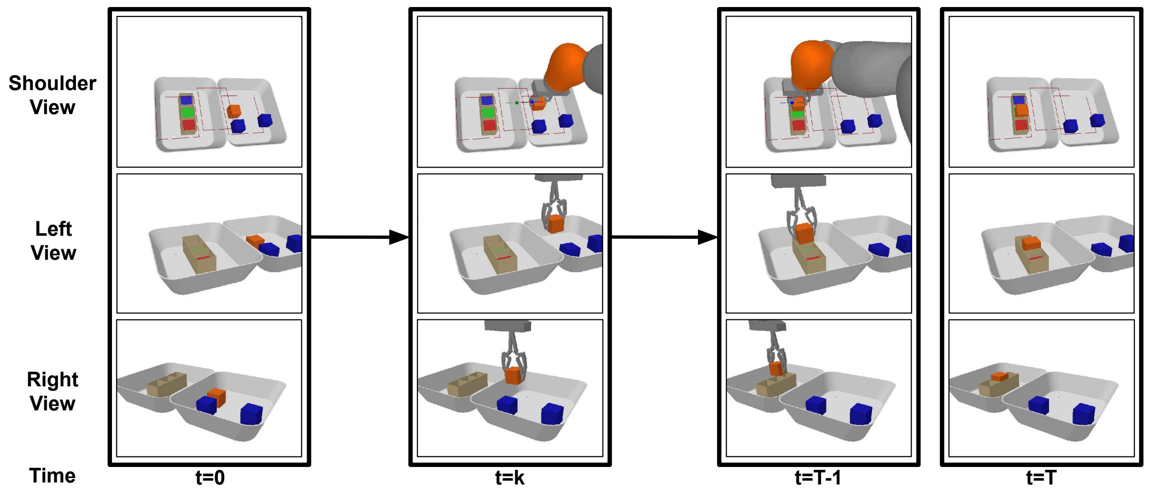
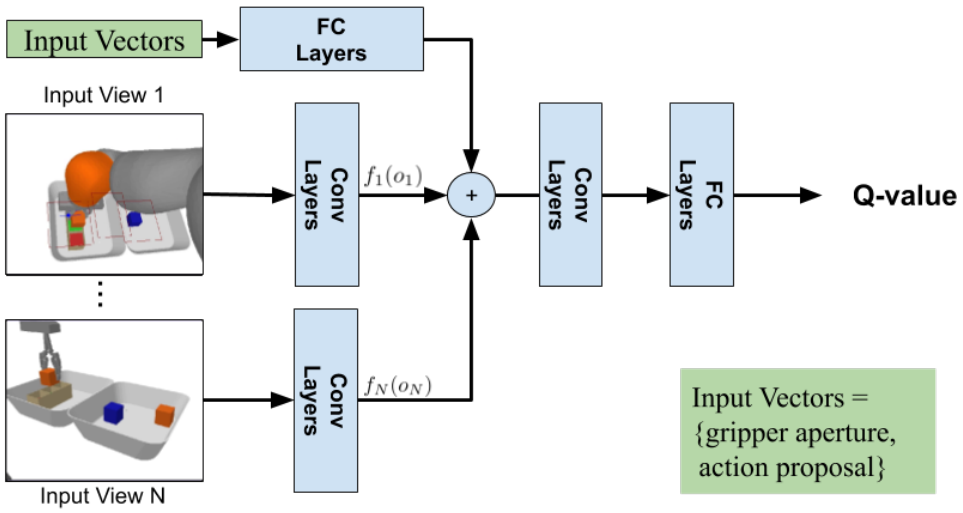


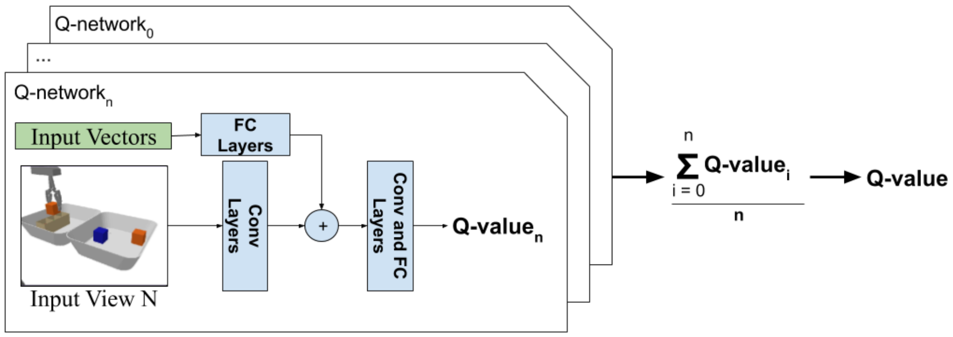
Figure 5.2: **Multi-view Insertion Task Learning.** A few key stages of the insertion task shown above include; start, pick, align, and drop. A single-view system that uses only the top image view struggles with stages that require 3D alignment. Our system combines information from multiple views and enhances performance on precision-based robotics. These 3 camera viewpoints are used for the Stacking I, Stacking II and Insertion tasks in the experimental section.

5.2.2 Vision-based Robot Manipulation

Many learning-based methods can now perform robot manipulation such as grasping directly from high-dimensional input such as images. For example, deep reinforcement learning (Deep-RL) algorithms can learn highly expressive neural networks by trial and error, that map image inputs to robot action. One important consideration in learning-based systems like Deep-RL is the choice of environmental state observation; usually only a partial observation of the state is possible. Color (RGB) image, depth image, 3D voxelized scene are all possible observation choices each with trade offs. For example, voxel representations give rich 3D information to a particular resolution level but it are difficult to obtain directly from existing sensors, it is very high dimensional and can be inefficient and limiting when used in learning. Depth image gives 2.5D information which is more efficient but depth cameras still struggle to handle textureless, dark-colored, transparent or reflective



(a) Multi-view Multi-Tower Q-Network Architecture



(b) Multi-view Aggregate Q-Network Architecture

Figure 5.3: a) A multi-tower architecture for incorporating multiple views. Each view has its own tower whose representations are then combined followed by additional network layers to produce a single Q-value. b) An aggregate architecture has a separate Q-network for each individual view, the final Q-value is the mean of the per view Q-values. See Figure 13 of QT-Opt[53] for details of the single-view architecture (with Conv and FC block definitions) from which our multi-view architectures were adapted. We use their original single-view network with modified input vectors shown above as a baseline.

materials depending on the technology used. RGB images give full color information useful for semantic understanding but lack depth information. RGBD combines the advantage of the previous two, but the 3D understanding is partial (i.e. 2D) and is still susceptible to occlusion.

Generative Query Network (GQN) [27] shows that a full scene can be represented using a vector encoding of multi-view images. Time Contrastive Networks (TCN) [94] demonstrate that multiple views can be utilized to learn rich viewpoint invariant representations. Both GQN and TCN representations can be utilized to learn robotic manipulation. While these works are similar to ours in the use of multi-view capture of the scene, our approach does not use auxiliary loss functions. Rather, we allow the neural network to focus on extracting features relevant solely to the task at hand, rather than reconstructing the scene, or producing viewpoint invariant representations in a way that may hurt asymptotic performance on the task[27].

Some recent works [33][17] use active sensing to determine successive sensor placements to improve state estimation for a task. In contrast, we use passive sensing via multiple cameras in a way that enables closed-loop reactive policy without adding intermediate camera-placement decision point into the sense-think-act loop. A previous work [76] used multiple static 3D cameras to capture and reconstruct the robot's environment in a non-learning based work. Our work differs in that we learn end-to-end directly from RGB images to robot action without any intermediate scene registration or reconstruction.

In summary, our approach addresses many limitations of these existing approaches to vision-based robot learning. Using an uncalibrated multi-camera system to capture RGB images from multiple viewpoints, we make the underlying state more observable and less susceptible to occlusion without suffering the computational and memory cost of an explicit

3D scene representation. Similar to [53][65], our method achieves closed-loop hand-eye coordination by learning the spatial relationship between the gripper and objects in the workspace, with reactive behaviors that ensure task success. While our experimental section is simulation only, there are variety of works that demonstrate how our multi-view system could be deployed on real hardware [53][6][47][14].

5.3 Preliminaries

While our method would work with most existing imitation learning and reinforcement learning approaches, we adopt QT-Opt [53]– a recent reinforcement learning algorithm that achieved state-of-the-art performance on a closed-loop vision-based robotic grasping task– as the foundation for our approach. Here, we briefly summarize the RL markov decision process (MDP) and QT-Opt formulation, for additional details please see [53].

5.3.1 RL Formulation for Task Learning

We use the standard RL MDP where the state of the task environment be given as $s_t \in \mathcal{S}$, the robot agent can make observation $o_t \in \mathcal{O}$ and can take action $a_t \in \mathcal{A}$ such that the transition dynamics is given as $s_{t+1} = \mathcal{T}(s_t, a_t)$. For notational convenience, state and observation are used interchangeably in the MDP. Given a reward function $\mathcal{R}(s_t, a_t)$ that captures the desired behavior and a discount factor $\gamma \in [0, 1)$, the goal of RL is to maximize the expected discounted cumulative reward across the episode. Similar to QT-Opt, we use the sparse binary reward signal in our setup which indicates task success at the end of the episode and zero otherwise. In addition, we have a small constant negative reward at each time step to incentivise the agent to solve the task faster. The action space includes 3D gripper displacement $\in \mathcal{R}^3$ and three binary commands (each $\in \{0, 1\}$) for gripper-open, gripper-close commands and a termination command to end the episode. Different from QT-Opt and most existing RL works, our observation space consists solely of multiple image views from uncalibrated statically-placed cameras and the gripper aperture (an ”Opened vs Closed” boolean).

5.3.2 Q Target Optimization (QT-Opt)

QT-Opt[53] is a continuous Q-Learning approach that learns a Q-value function which is then optimized, in a model predictive control fashion, to choose optimal actions that maximize the learned Q-function. QT-Opt achieves this by learning a Q-function, represented by a neural network, that captures the expected discounted cumulative sum of reward starting from a given state and taking the action.

$$\mathcal{Q}_\theta(s,a) = r(s,a) + \gamma \max_{a'} \mathcal{Q}_\theta(s',a') \tag{5.1}$$

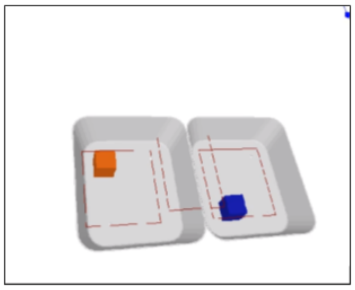
The policy is recovered from a learned Q-function via:

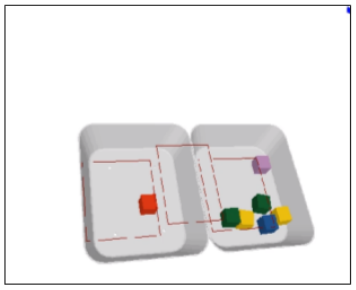
$$\pi(s) = \arg\max_{a} \mathcal{Q}_\theta(s,a) \tag{5.2}$$

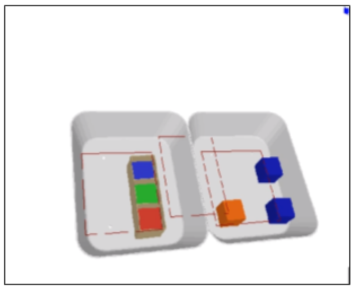
The Q-function maximization is done with a cross-entropy method (CEM)– a derivative-free optimization algorithm.

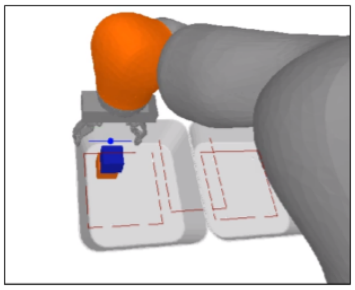
5.3.3 Robot Simulation Setup

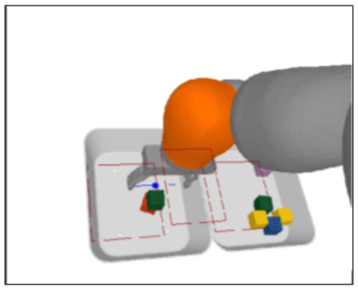
We use the Kuka IIWA arm with a parallel jaw gripper as the robot platform in our experiments, although our method is independent of the specific robot hardware. Using the Bullet Physics simulator [22], we create a simulation environment (Figure 5.2) where two bins are placed in front of the robot and based on the task at hand, objects are placed in


(a) Stacking I: Start


(b) Stacking II: Start


(c) Insertion: Start


(d) Stacking I: Goal


(e) Stacking II: Goal

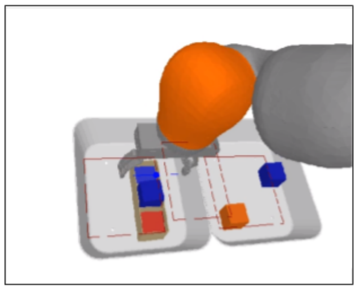

(f) Insertion: Goal

Figure 5.4: **Tasks with Varying Difficulty** The value of multi-view task learning depends on the level of 3D understanding and precision required for the task. The images above show sampled initial images and final images to illustrate the desired outcomes. **Left [a,d]**: The Stacking I task requires a block from the right side be placed on top of the block on the left side. The task has a large margin of error since the blocks are big enough that perfect alignment isn't required to succeed. **Middle [b,e]**: The blocks are smaller so there is a need for more precise placement, hence the performance benefit of having multiple views is potentially higher. **Right, [c,f]**: The insertion task requires the block placed into the middle placement location (green hole) of the fixture. This requires precise alignment which difficult from a single view, hence there is significant benefit to using multiple views.

the bins at random locations. Three cameras were mounted to overlook the bins where the task is being performed; with respect to the robot, one camera is over-the-shoulder, one is to the left and one is to the right.

5.4 Multi-View Task Learning

Most vision-based learning algorithms take a single camera image input as an observation of the state of the environment. This may be reasonably sufficient for tasks requiring more coarse manipulation, but for tasks that require high-level of precision such as the insertion task in Figure 5.2, a single view usually cannot capture enough of the state information to achieve superior performance.

For single-view task learning, we take a closer look at the Q-function expressed as a deep neural network and observe that it can be factorized given that the state input has visual and non-visual components i.e. $s = (s_{visual}, s_{non\_visual})$. The visual component is the image observation while the non-visual component is the gripper "Opened/Closed" status. As a result, the Q-function can be decomposed as:

$$\begin{aligned} \mathcal{Q}_\theta(s, a) &= \mathcal{Q}_\theta(s_{visual}, s_{non\_visual}, a) \\ &= \mathcal{Q}(f(s_{visual}), g(s_{non\_visual}), h(a)) \end{aligned} \tag{5.3}$$

where f, g and h are vector valued functions with f being a sequence of 2D convolutional layers while g and h are a sequence of dense layers.

For a single view system, $f(s_{visual})$ is a function over the input image while in the multi-view setting, there can be different functions $f_1, f_2, ..., f_n$ that process image observations from viewpoints 1 through n. Below, we present different ways of expressing and composing the functions for both single and multiple views systems.

5.4.1 Single View

We evaluate against several single view architectures. Single View RGB from the shoulder (SV_Shoulder). We additionally explore the use of depth images from the shoulder view. We look at (SV_RGBD) where the RGB and Depth are fed into separate CNN towers of the network.

5.4.2 Multi-Tower (MV_Towers)

Shown in Figure 5.3a, each image observation o_i is passed through a separate vision processing module f_i which is a sequence of convolution layers to produce visual embeddings $f_i(o_i)$ from each viewpoint. These embeddings are averaged across views and combined with action proposals before going through more convolution and dense layers to produce the Q-value $Q(s, a)$. Note that even though the function for each viewpoint have the same form, each function is different with a separate set of weights i.e. $f_i \neq f_j, \forall i \neq j$. The overall vision module is given as:

$$f = \frac{1}{n} \sum_{i=1}^{n} f_i(o_i)$$

5.4.3 Siamese (MV_Siamese)

MV_Siamese is a modification of MV_Towers such that the convolutional weights across the multi-views are shared i.e. $f_i = f_j, \forall i, j$. This reduces the total number of weights to be trained, shares the data from multiple views to train the same set of weights, imposes a constraint that visual data from all views should be processed similarly.

5.4.4 Sensor Dropout (MV_Dropout)

MV_Dropout is a modification of MV_Towers. The data from one or more of the cameras is randomly masked out during training. This aims to improve robustness to the absence of camera views.

$$f = \frac{1}{C} \sum_{i=1}^{n} \delta_i f_i(o_i)$$

where $\delta_i \sim \text{Bernoulli}(p)$ and $C = \sum_{i=1}^{n} \delta_i$ is the normalizing constant that ensures the scale of the output does not vary with the number of camera viewpoints selected. Note that with this formulation it is possible to have no camera selected which is undesirable and a waste of training cycle even if happens less frequently. An implementation detail is to list out all possible combination where one or more of the camera view-points are selected and put a uniform distribution to randomly select one of these possible options. A similar sensor-dropout idea was shown to improve multi-sensory fusion for autonomous navigation task [70].

5.4.5 Q-Aggregate Network (MV_Q_AGG)

Shown in Figure 5.3b this multi-view approach creates a separate Q-function per input viewpoint and all Q outputs are aggregated into a single Q-value. This is a consensus action approach where each view predicts the Q-value of each proposed action given the current state. The action with the highest Q-value (aggregated across views) is selected. During training, the mean aggregate is preferred so that gradient flows through the entire network for all views; min/max operations would only allow gradient flow through a single branch of the network for each training datapoint. A drawback of the Q-Aggregate approach, regardless of the aggregate function, is that there are more parameters to train.

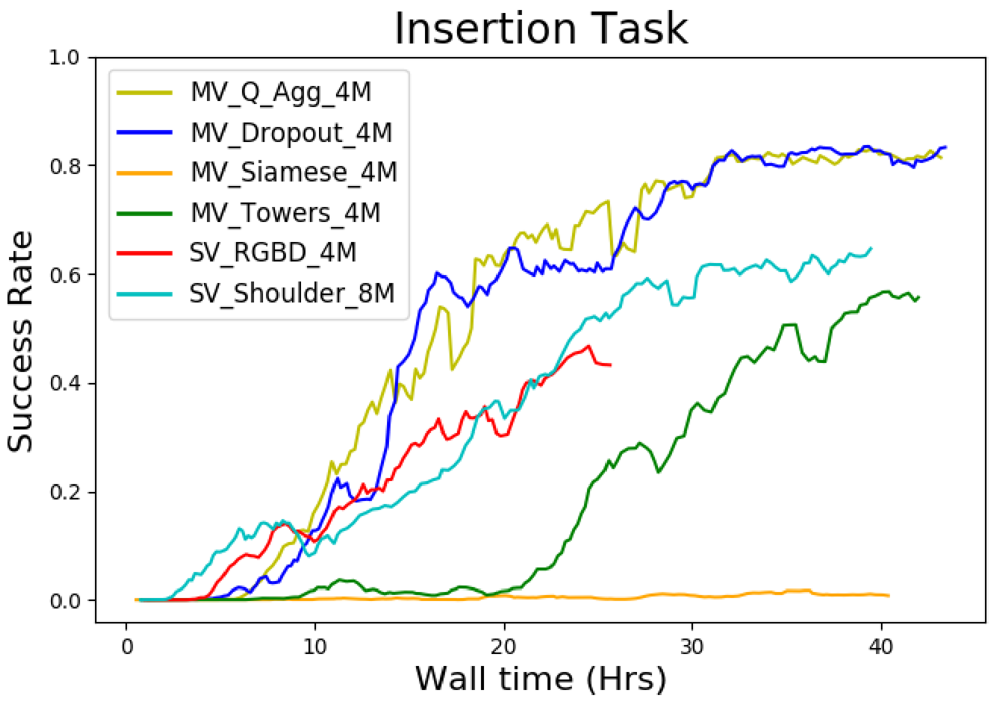


Figure 5.5: **Insertion Task Training Curves.** Running average comparison of SV and MV architectures trained either 4 million (4M) or 8 million (8M) training iterations taking up to 40 hours. MV_Dropout and MV_Q_Agg results achieve the best and comparable performance on this task. Importantly, sensor dropout during training leads to a huge difference in performance between MV_Towers and MV_Dropout.

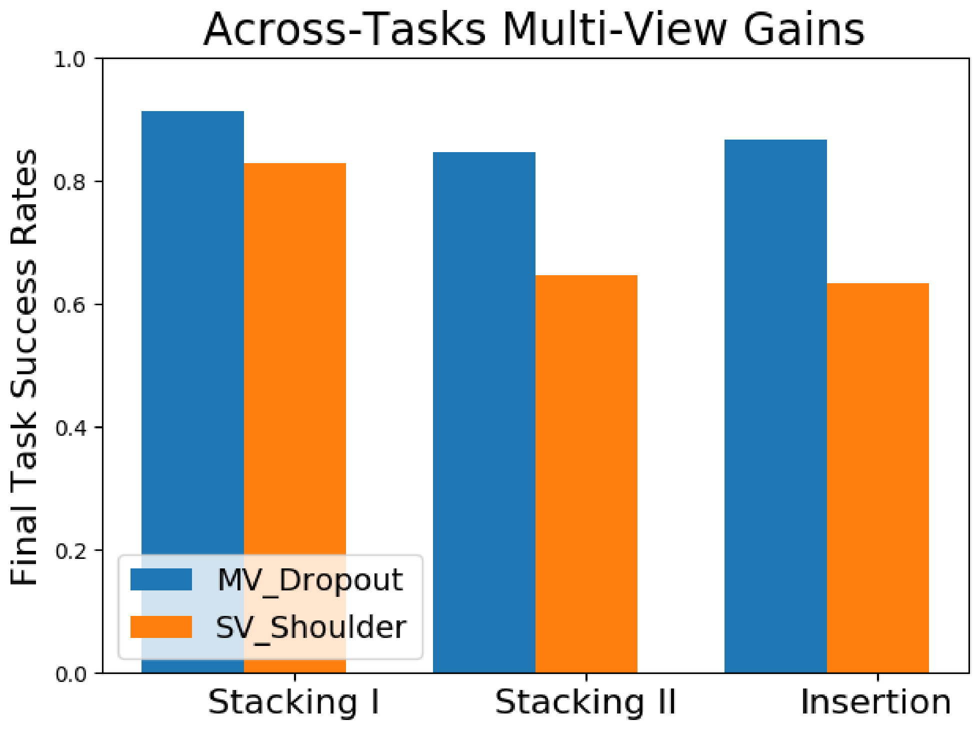


Figure 5.6: **Comparison of best performing SV and MV models across different tasks.** The relative gains from a multi-view approach are dependent on task with harder tasks gaining more. Switching from single to multi-view results in the following absolute performance gains: Stacking I **8.97%**; Stacking II **20.14%**; and Insertion **23.43%**

Table 5.1: **View Dropout Experiment** (% Task Success on Insertion Task): This table overviews how different trained policies perform as the number of views available at runtime is reduced. The Multi-View and Multi-View (Dropout) rows were all trained with observations from 3 views, but evaluated with 3, 2, and 1 view. The single view baselines were trained with 1 view and evaluated with 1 view. Of note is the fact that the Multi-View (Dropout) significantly outperforms the Single View baselines even when only provided a single view at runtime. It also outperforms the Multi-View model trained and evaluated with 3 views implying the dropout procedure has benefits even when all views are available. All numbers in the table come from the average of evaluating the final trained policy for 700 episodes.

# of Views at Runtime:	3 Views	2 Views			1 View		
	All Views	Shoulder + Left	Shoulder + Right	Left + Right	Shoulder	Left	Right
MV_Towers_4M [trained w/ 3 views]	58.0	0.43	3.86	0	0	0	0
MV_Dropout_4M [trained w/ 3 views]	**86.86**	68.0	83.0	70.0	32.14	34.29	14.57
SV_Shoulder_8M [trained w/ 1 view]	N/A	N/A	N/A	N/A	63.43	N/A	N/A

5.5 Experiments

For our experiments, three cameras were placed pointed at the robot's workspace. With respect to the robot, one camera is roughly over the shoulder, one to the left and one to the right. For each episode, 0.01 std-dev uniform noise is added to the camera position, look and up vectors used to define the camera pose, for each camera, to simulate imperfect camera calibration and improve generalization. The bin locations are randomized with x, y, z position noise sampled uniformly from the ranges ($\pm$0.025, $\pm$0.05, $\pm$0.05). To compare various multi-view and single-view approaches, three simulated robotic tasks were used as test-beds:

Stacking I (Figure 5.4a, 5.4d): The right bin starts with a single block (5cm edge length) either blue or orange in a random position. The left bin also starts with a single block either blue or orange. An episode of the task is counted as a success if at the end of the episode there are two blocks in the left bin with one on top of the other.

Stacking II (Figure 5.4b, 5.4e): The right bin starts with 6 small blocks (3.8cm edge length) in random positions, while the left bin starts with a single small cube in a random position. An episode of the task is a success if at the end of the episode there are two blocks in the left bin with one on top of the other. In contrast to Stacking I, the blocks here are smaller which lowers the margin for error.

Insertion (Figure 5.4c, 5.4f): The right bin starts with 3 blocks (5cm edge length) either blue or orange in random positions. The left bin starts with a fixture at a random position, but fixed orientation. An episode of the task is a success if at the end the fixture is in the left bin, and has a block firmly inserted into the middle fixture position. The fixture location has 9mm of clearance for the cube.

Poor exploration is a known issue in sparse-reward settings and previous works [53][123] have shown the importance of providing some demonstration data to aid exploration and bootstrap the reinforcement learning. Similarly, we bootstrap off a simple scripted sub-optimal policy (yielding about 20% for the Insertion task) to collect demonstration data which is included in the replay buffer to address the exploration issue.

Model performance is affected both by the number of training iterations and by the duration of time that training is run for. 180 data collection jobs are run for all training workflows producing ∼5000 episodes per hour for the insertion task with some variability across model performance and task. The 50,000 most recent episodes are kept in an experience replay buffer which takes ∼10 hours to initially reach capacity. Over time, the distribution of episodes in the buffer will shift as the policy learns the task and older episodes are evicted. We used 4 million gradient updates for the multi-view and RGBD architectures, but provided the SV_Shoulder 8 million gradient steps in order to give it a comparable wall clock time and comparable amount of collected data. The models were trained following the QT-Opt setup with 1000 bellman update workers and 10 GPUs.

5.5.1 Single and Multi-View Insertion

Figures 5.4c and 5.4f shows the insertion task. As shown in Figure 5.5, single view approaches to task learning struggle to learn this task where the margin for error in aligning the block to the fixture is very low. This is true even when the single view approaches are provided depth information in the form of a depth image channel (SV_RGBD). Conversely, we see the value of using multiple camera-view as input into the system. In addition, we see that using sensor dropout gives further improvement on the multi-view performance. This boost might be because dropping out camera views during training forces the network to squeeze out as much information from each camera. Note that at inference time, we use all three cameras. The Q-Agg network also achieves comparable high performance although it contains more parameters.

5.5.2 Varying Task Difficulty

From Figure 5.6, notice that the relative performance of the multi-view system compared to single-view varies by task. For tasks that can accommodate a high margin of precision error (such as Stacking I where the blocks are large enough to overlap with little precision), the performance gap between single versus multiple view systems is small. As tasks require more and more precise manipulation, the benefit of multiple views becomes more apparent. By switching from SV_Shoulder to MV_Dropout, task failure rate on Stacking I dropped from 17.14% to 8.71%- a **49.18%** reduction in failure rate i.e. (17.14% - 8.71%) / 17.14%. Higher performance gains were seen on relatively harder tasks: Stacking II failure rate dropped from 35.43% to 15.29% a **56.84%** reduction in failure rate and Insertion dropped from 36.57% to 13.14% a **64.1%** reduction in failure rate.

5.5.3 Camera Dropout Robustness Test

We experimentally demonstrate the robustness that result from the use of sensor dropout during training by randomly taking out one or two of the cameras during evaluation and measure the task performance. As show in Table 5.1, while both multi-view approaches perform better that the single-view baseline, using sensor dropout during training makes a multi-view system robust to loss of camera after deployment; it leverages the inherent redundancy of the multi-view architecture to achieve a more reliable system. Compared to MV_Towers where loss of camera view after training is catastrophic, the performance of MV_Dropout drops 4% to 19% when one view is taken out and 53% to 73% when two viewpoints are taken out depending on which views are removed. The Shoulder and Left views are positioned close to each other so the Shoulder + Right performs much better than Shoulder + Left. From the right view alone it can be difficult to see the fixture so it is the worst performing single view. The MV_Dropout model is able to work half as well as the SV_Shoulder model, when operating off only a Shoulder view, this model is not specialized to that specific image and also has had to learn to operate off other view combinations.

5.6 Summary

This chapter introduces a multi-view approach to robotic learning for precision-based tasks. Our approach directly learns the task without going through an intermediate step of reconstructing the scene. The effective use of multiple views enables a richer observation of the underlying state relevant to the task. Experiments on precision-based stacking and insertion tasks show that our sensor-dropout approach to multi-view task learning achieves superior performance compared to the common single view approach. This improvement can be seen in the asymptotic performance as well as robustness to occlusion and loss of camera views. Our multi-view approach enables 3D tasks from RGB cameras without the need for explicit 3D representations and without camera-camera and camera-robot calibration.

Chapter 6

Conclusion

The journey towards making robots that can handle fast changing, less-structured human-centered environments, and are more amenable to different tasks is still in its early stages. Some of the key challenges we face include: (i) limited use of the workspace where robots are not able to fully utilize 6DOF grasps to pick-up objects in difficult-to-reach regions, (ii) difficulty in picking up moving objects in cluttered scenes, (iii) lack of robustness during the final grasping phase due to the open-loop nature of most algorithms, (iv) difficulty in achieving closed-loop manipulation from vision using the common single-view camera settings.

In this thesis, we draw inspiration from the human ability to seamlessly manipulate objects to address these challenges of robot manipulation. To achieve this, we develop different sensibilities that improve robots' performance and reliability in manipulation tasks. In Chapter 2, we developed a novel representation for reachability awareness that enables robotic grasping of static objects in difficult-to-reach regions around the robot. The reachability space, computed offline, is used at runtime to bias the grasp planner towards accessible regions of the grasping scene. This results in a higher percentage of reachable grasps, a higher percentage of successful grasp executions, and a reduced planning time. Based on the robot's morphology, this ability to reason about regions that can be reached is valuable in both static and dynamic scenarios. In Chapter 3, we showed that reachability awareness is also valuable in dynamic settings where the target object is moving within static obstacles. To achieve reliable dynamic grasping, we also developed a neural network for motion-awareness that predicts and selects robust grasps based on the motion of the object. Our notion of reachability-awareness and motion-awareness help to identify reachable and robust grasps. To smoothly move the arm to the planned reachable grasp, we developed a seeding approach for generating arm motion trajectory that enables fast computation and smooth arm motion. Our extensive experiments demonstrate the importance of each of these components. In Chapter 4, we developed an adaptive tactile-aware grasping algorithm that uses tactile sensing to close the loop during the last phase of grasping. This closed-loop behavior was obtained using policy-gradient reinforcement learning, learned purely in simulation and transferred to the real robot in a high-fidelity manner. The resulting algorithm helps turn common failure cases into successes and make grasping systems significantly more robust to camera-robot calibration errors. Our work can be used as an add-on to existing open-loop grasp planners to improve their robustness. In Chapter 5, we presented a multi-view continuous q-learning approach that uses reinforcement learning to learn object manipulation tasks like insertion that requires a level of 3D understanding. Our use of static but uncalibrated cameras does not require camera-robot or camera-camera calibration making the proposed approach easy to setup, and our use of sensor dropout during training makes it resilient to the loss of camera-views after deployment. The obtained end-to-end policy is reactive– directly mapping from high-dimensional multi-view image observation to robot action – enabling failure recovery behaviors.

System Implementation Details: In this thesis, we reasoned about robotic manipulation along different philosophical themes which manifest themselves in different system implementation details. Following are a summary of some system considerations and decisions made to realize the ideas introduced in this work:

- *model-based grasping:* This entails an initial step of parsing the image of the grasping scene into object models, using object recognition and pose detection modules. The grasping algorithms are then developed and applied to the pre-processed grasping scene to generate pick-up actions for specific target objects while avoiding collision with other *distractor* objects. Chapters 2 and 3 take this approach to develop a reachability-aware grasping algorithm for static and moving objects.

- *model-free grasping:* This approach directly generates grasp poses and actions from raw sensory inputs (images or tactile readings) without recreating the models of the objects in the scene. This approach has no notion of objectness but, as shown in Chapter 4, it is able to generalize well across different scenes with novel objects.

- *End-to-End manipulation:* The inputs and outputs of manipulation algorithms can occur at different levels. For example, the inputs can be raw images or preprocessed object models and poses, while the output can vary from high-level action spaces such as a grasp pose, to a lower-level space such as end-effector displacement, or a much lower-level space such as joint-torques. To execute high-level actions like moving to a grasp pose, the action output still requires further processing such as motion planning. An end-to-end manipulation systems strives to go directly from raw sensor inputs to low-level action-spaces that would require less post-processing of the action which can be executed in a tighter loop. Chapter 5 presents an end-to-end manipulation system that directly predicts end-effector displacement from raw multi-view camera inputs.

- *Closed-loop manipulation:* An open-loop system predicts a final high-level robot action (e.g. a grasp) that is post-processed and realized by another module. On the other hand, a closed-loop system has a feedback component that continuously observes the current robot state and environment and uses the observation to predict a lower level action. This enables robust and reactive behaviors as shown in Chapters 4 and 5.

6.1 Current Limitations and Future Work

There are several avenues for future work that build on the works in this thesis. As a first step, a tighter unification of the methods developed would be very valuable. For example, while reachability and motion awareness was shown to improve dynamic grasping, it would be interesting to investigate the importance of tactile awareness during dynamic grasping. As was demonstrated with static multifingered grasping, tactile sensing is likely to show performance benefits when picking up moving objects especially during the approach-and-grasp phase. This tight integration would bring to the fore how to best combine model-based and model-free methods. While Chapters 2 and 3 are model-based in that object models are first extracted from the camera sensors before passing the models through other modules of the algorithm, Chapter 4 and 5 introduce more model-free approaches that directly map sensor observations such as camera images and tactile sensor readings into robot actions such as grasp types/arm joint motions. Unification of these techniques into one would be valuable in better understanding the boundary between model-based and model-free; while yielding a strong manipulation system.

Other future avenues for research are highlighted as follows:

Sensor Integration for Complex Manipulation: Chapter 4 presents an approach to achieve closed-loop behavior using tactile sensor readings in the final phase of grasping while Chapter 5 derives a closed-loop policy using multiple camera-views for different manipulation tasks. Since these different modalities (vision and tactile) have their advantages, it would be interesting to develop approaches that better integrate these and other sensor measurements to achieve more complex tasks like tool-use e.g. using the hammer or a screwdriver. Tighter integration of multi-modal sensing is important to the future of robotic manipulation.

Stronger Visual Priors from Multi-view Robotic Systems: In Chapter 5, we present an end-to-end multi-view vision-based manipulation policy that directly maps images from multiple views to robot actions. We introduced a sensor dropout approach that improves the training behavior and overall performance on multi-stage precision tasks like stacking and insertion. This direct mapping from pixels to robot joint actions makes this approach run in closed-loop and can be more amenable to dynamic applications. There are a few directions to take this further. By learning from multiple views, strong visual priors can be leveraged to improve training speed, policy performance and task generalization. For example, scene consistency in regions where some camera views overlap can provide strong training signals. In addition, improved quality of encoding representation can aid transferability to new tasks and potentially sim-to-real transfer. For our multi-view work, we arbitrarily placed the cameras at poses that roughly overlook the robot's workspace. An important question is to determine the optimal camera placements. In addition, it would interesting determine how the number of cameras affect task performance and speed and if there is a minimum number of cameras required for different tasks.

6.2 Closed-Loop Sensory Awareness: An important ingredient for Complex Robotic Manipulation

Current robotic manipulation skills remain far from human-level versatility and there are a number of algorithmic and technological ingredients required to get there. These include sensor design, sensor fusion, transfer learning, planning algorithms, feedback control among others. Using robotic grasping, stacking and insertion tasks as test-beds, we have demonstrated in this thesis how advancement in each of these themes can take us closer to achieving human-level robotic manipulation. To improve static grasping and achieve dynamic grasping, we showed the importance of fast planning algorithms. Our reachability-aware grasping approach enables fast grasp planning and grasp selection while our trajectory seeding approach achieves fast generation of smooth arm trajectory in dynamic settings. The advantage of feedback-based/closed-loop engineering systems over open-loop alternatives is well established, although achieving feedback in robotic manipulation is not straight-forward. We demonstrated that a closed-loop grasping behavior can be learned in simulation using reinforcement learning. Using careful choices of observation and action spaces, this reinforcement-learned policy can be transferred from simulation to the real robot in a high fidelity manner. We also demonstrated that a closed-loop vision-based policy from multiple camera views can be learned using continuous q-learning. Our sensor-dropout mechanism makes this multi-view policy robust to loss of camera views.

Extending the ideas of fast, closed-loop manipulation developed in this work to more complex tasks such as manipulating deformable objects and longer-horizon tasks would be an important next step towards more versatile robotic manipulation.

Bibliography

[1] Iretiayo Akinola, Jacob Varley, and Dmitry Kalashnikov. "Learning precise 3D manipulation from multiple uncalibrated cameras". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 4616–4622.

[2] Iretiayo Akinola et al. "Dynamic Grasping with Reachability and Motion Awareness". In: *arXiv preprint arXiv:2103.10562* (2021).

[3] Iretiayo Akinola et al. "Workspace Aware Online Grasp Planning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2917–2924.

[4] Peter K Allen, Matei Ciocarlie, and Corey Goldfeder. "Grasp planning using low dimensional subspaces". In: *The Human Hand as an Inspiration for Robot Hand Development*. Springer, 2014, pp. 531–563.

[5] Peter K Allen et al. "Automated tracking and grasping of a moving object with a robotic hand-eye system". In: *IEEE Transactions on Robotics and Automation* 9.2 (1993), pp. 152–165.

[6] Marcin Andrychowicz et al. "Learning dexterous in-hand manipulation". In: *arXiv preprint arXiv:1808.00177* (2018).

[7] L C Baird. "Reinforcement Learning in Continuous Time: Advantage Updating". In: *IEEE International Conference on Neural Networks (ICNN)*. Vol. 4. June 1994, pp. 2448–2453.

[8] Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. "A robot path planning framework that learns from experience". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3671–3678.

[9] Marten Björkman et al. "Enhancing visual perception of shape through tactile glances". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 3180–3186.

[10] James E Bobrow, Steven Dubowsky, and John S Gibson. “Time-optimal control of robotic manipulators along specified paths”. In: *The international journal of robotics research* 4.3 (1985), pp. 3–17.

[11] Roberto Calandra et al. “More Than a Feeling: Learning to Grasp and Regrasp using Vision and Touch”. In: *arXiv preprint arXiv:1805.11085* (2018).

[12] Roberto Calandra et al. “The feeling of success: Does touch sensing help predict grasp outcomes?” In: *arXiv preprint arXiv:1710.05512* (2017).

[13] Angel X Chang et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012* (2015).

[14] Yevgen Chebotar et al. “Closing the sim-to-real loop: Adapting simulation randomization with real world experience”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8973–8979.

[15] Yevgen Chebotar et al. “Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 1960–1966.

[16] Xiaotong Chen et al. “GRIP: Generative Robust Inference and Perception for Semantic Robot Manipulation in Adversarial Environments”. In: *arXiv preprint arXiv:1903.08352* (2019).

[17] Ricson Cheng, Arpit Agarwal, and Katerina Fragkiadaki. “Reinforcement Learning of Active Vision forManipulating Objects under Occlusions”. In: *arXiv preprint arXiv:1811.08067* (2018).

[18] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. “Dimensionality reduction for hand-independent dexterous robotic grasping”. In: *Intelligent Robots & Systems*. IROS. 2007, pp. 3270–3275.

[19] Matei T Ciocarlie and Peter K Allen. “Hand posture subspaces for dexterous robotic grasping”. In: *IJRR* 28.7 (2009).

[20] Deen Cockbum et al. “Grasp stability assessment through unsupervised feature learning of tactile images”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2238–2244.

[21] David Coleman et al. “Experience-based planning with sparse roadmap spanners”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 900–905.

[22] E Coumans and Y Bai. "Pybullet, a python module for physics simulation for games, robotics and machine learning". In: *GitHub* (2016).

[23] Hao Dang and Peter K Allen. "Stable grasping under pose uncertainty using tactile feedback". In: *Autonomous Robots* 36.4 (2014), pp. 309–330.

[24] Hao Dang, Jonathan Weisz, and Peter K Allen. "Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics." In: *ICRA*. 2011, pp. 5917–5922.

[25] Rosen Diankov and James Kuffner. "Openrave: A planning architecture for autonomous robotics". In: *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34* 79 (2008).

[26] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. "A Billion Ways to Grasp: An Evaluation of Grasp Sampling Schemes on a Dense, Physics-based Grasp Data Set". In: *arXiv preprint arXiv:1912.05604* (2019).

[27] SM Ali Eslami et al. "Neural scene representation and rendering". In: *Science* 360.6394 (2018), pp. 1204–1210.

[28] Javier Felip, Jose Bernabé, and Antonio Morales. "Contact-based blind grasping of unknown objects". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE. 2012, pp. 396–401.

[29] Javier Felip et al. "Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks". In: *Robotics and Autonomous Systems* 61.3 (2013), pp. 283–296.

[30] Carlo Ferrari and John Canny. "Planning optimal grasps". In: *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE. 1992, pp. 2290–2295.

[31] Corey Goldfeder and Peter K Allen. "Data-driven grasping". In: *Autonomous Robots* 31.1 (2011), pp. 1–20.

[32] Corey Goldfeder et al. "The columbia grasp database". In: *2009 IEEE international conference on robotics and automation*. IEEE. 2009, pp. 1710–1716.

[33] Marcus Gualtieri and Robert Platt. "Viewpoint selection for grasp detection". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 258–264.

[34] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).

[35] Kaiyu Hang, Johannes A Stork, and Danica Kragic. “Hierarchical fingertip space for multi-fingered precision grasping”. In: *Intelligent Robots and Systems (IROS)*. IEEE. 2014, pp. 1641–1648.

[36] Kris Hauser. “Robust contact generation for robot simulation with unstructured meshes”. In: *Robotics Research*. Springer, 2016, pp. 357–373.

[37] Francois R Hogan et al. “Tactile regrasp: Grasp adjustments via simulated tactile transformations”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2963–2970.

[38] Nasser Houshangi. “Control of a robotic manipulator to grasp a moving target using vision”. In: *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE. 1990, pp. 604–609.

[39] Emil Hyttinen, Danica Kragic, and Renaud Detry. “Estimating tactile data for adaptive grasping of novel objects”. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE. 2017, pp. 643–648.

[40] Emil Hyttinen, Danica Kragic, and Renaud Detry. “Learning the tactile signatures of prototypical object parts for robust part-based grasping of novel objects”. In: *2015 IEEE international conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 4927–4932.

[41] Jarmo Ilonen, Jeannette Bohg, and Ville Kyrki. “Fusing visual and tactile sensing for 3-d object reconstruction while grasping”. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 3547–3554.

[42] Lester Ingber. “Very fast simulated re-annealing”. In: *Mathematical and computer modelling* 12.8 (1989), pp. 967–973.

[43] Fahad Islam et al. “Provably Constant-time Planning and Replanning for Real-time Grasping Objects off a Conveyor Belt”. In: *arXiv preprint arXiv:2101.07148* (2021).

[44] Jan Issac et al. “Depth-based object tracking using a robust gaussian filter”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 608–615.

[45] Gregory Izatt et al. “Tracking objects with point clouds from vision and touch”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4000–4007.

[46] Divye Jain et al. “Learning Deep Visuomotor Policies for Dexterous Hand Manipulation”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3636–3643.

[47] Stephen James et al. “Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12627–12637.

[48] Leif P Jentoft, Qian Wan, and Robert D Howe. “Limits to compliance and the role of tactile sensing in grasping”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 6394–6399.

[49] Mohsen Kaboli et al. “A tactile-based framework for active object learning and discrimination using multimodal robotic skin”. In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 2143–2150.

[50] Mohsen Kaboli et al. “Tactile-based active object discrimination and target object search in an unknown workspace”. In: *Autonomous Robots* 43.1 (2019), pp. 123–152.

[51] Mrinal Kalakrishnan et al. “Learning force control policies for compliant manipulation”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 4639–4644.

[52] Mrinal Kalakrishnan et al. “STOMP: Stochastic trajectory optimization for motion planning”. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 4569–4574.

[53] Dmitry Kalashnikov et al. “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation”. In: *arXiv preprint arXiv:1806.10293* (2018).

[54] R.E. Kalman. “A new approach to linear filtering and prediction problems”. In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45.

[55] Daniel Kappler et al. “Real-time perception meets reactive motion generation”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1864–1871.

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

[57] Danica Kragic and Henrik I Christensen. “A framework for visual servoing”. In: *International Conference on Computer Vision Systems*. Springer. 2003, pp. 345–354.

[58] Danica Kragic, Henrik I Christensen, et al. “Survey on visual servoing for manipulation”. In: *Computational Vision and Active Perception Laboratory, Fiskartorpsv* 15 (2002), p. 2002.

[59] Oliver Kroemer, Scott Niekum, and George Konidaris. “A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms”. In: *arXiv preprint arXiv:1907.03146* (2019).

[60] James J Kuffner and Steven M LaValle. “RRT-connect: An efficient approach to single-query path planning”. In: *Robotics and Automation, 2000. Proceedings. ICRA’00. IEEE International Conference on.* Vol. 2. IEEE. 2000, pp. 995–1001.

[61] Jennifer Kwiatkowski, Deen Cockburn, and Vincent Duchaine. “Grasp stability assessment through the fusion of proprioception and tactile signals using convolutional neural networks”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. 2017, pp. 286–292.

[62] Friedrich Lange, Michael Suppa, and Gerd Hirzinger. “Control with a compliant force-torque sensor”. In: *ROBOTIK 2012; 7th German Conference on Robotics.* VDE. 2012, pp. 1–6.

[63] Michelle A Lee et al. “Making Sense of Vision and Touch: Learning Multimodal Representations for Contact-Rich Tasks”. In: *arXiv preprint arXiv:1907.13098* (2019).

[64] Ian Lenz, Honglak Lee, and Ashutosh Saxena. “Deep learning for detecting robotic grasps”. In: *The International Journal of Robotics Research* 34.4-5 (2015), pp. 705–724.

[65] Sergey Levine et al. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 421–436.

[66] Miao Li et al. “Learning of grasp adaptation through experience and tactile sensing”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems.* Ieee. 2014, pp. 3339–3346.

[67] Y Litvak, A Biess, and A Bar-Hillel. “Learning Pose Estimation for High-Precision Robotic Assembly Using Simulated Depth Images”. In: *2019 International Conference on Robotics and Automation (ICRA).* IEEE. 2019, pp. 3521–3527.

[68] Qingkai Lu and Tucker Hermans. “Modeling grasp type improves learning-based grasp planning”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 784–791.

[69] Qingkai Lu et al. “Planning multi-fingered grasps as probabilistic inference in a learned deep network”. In: *arXiv preprint arXiv:1804.03289* (2018).

[70] Jianlan Luo et al. “Reinforcement Learning on Variable Impedance Controller for High-Precision Robotic Assembly”. In: *arXiv preprint arXiv:1903.01066* (2019).

[71] Robert Mahony, Peter Corke, and Tarek Hamel. “Dynamic image-based visual servo control using centroid and optic flow features”. In: *Journal of Dynamic Systems, Measurement, and Control* 130.1 (2008), p. 011005.

[72] Lucas Manuelli et al. “kPAM: KeyPoint Affordances for Category-Level Robotic Manipulation”. In: *arXiv preprint arXiv:1903.06684* (2019).

[73] Naresh Marturi et al. “Dynamic grasp and trajectory planning for moving objects”. In: *Autonomous Robots* 43.5 (2019), pp. 1241–1256.

[74] Arjun Menon, Benjamin Cohen, and Maxim Likhachev. “Motion planning for smooth pickup of moving objects”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 453–460.

[75] Hamza Merzić et al. “Leveraging Contact Forces for Learning to Grasp”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3615–3621.

[76] Justinas Mišeikis et al. “Multi 3D camera mapping for predictive and reflexive robot manipulator trajectory estimation”. In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2016, pp. 1–8.

[77] Douglas Morrison, Peter Corke, and Jürgen Leitner. “Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach”. In: *Robotics: Science and Systems (RSS)* (2018).

[78] Adithyavairavan Murali et al. “Learning to grasp without seeing”. In: *arXiv preprint arXiv:1805.04201* (2018).

[79] Ashvin Nair et al. “Overcoming exploration in reinforcement learning with demonstrations”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6292–6299.

[80] Helen Oleynikova et al. “Signed distance fields: A natural representation for both mapping and planning”. In: *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan. 2016.

[81] Chavdar Papazov and Darius Burschka. “An efficient ransac for 3d object recognition in noisy and occluded scenes”. In: *Asian Conference on Computer Vision*. Springer. 2010, pp. 135–148.

[82] Andreas ten Pas et al. “Grasp pose detection in point clouds”. In: *The International Journal of Robotics Research* 36.13-14 (2017), pp. 1455–1473.

[83] Lerrel Pinto and Abhinav Gupta. “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours”. In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 3406–3413.

[84] Ivaylo Popov et al. “Data-efficient deep reinforcement learning for dexterous manipulation”. In: *arXiv preprint arXiv:1704.03073* (2017).

[85] Oliver Porges et al. “Reachability and capability analysis for manipulation tasks”. In: *ROBOT2013: First Iberian Robotics Conference*. Springer. 2014, pp. 703–718.

[86] Nathan Ratliff et al. “CHOMP: Gradient optimization techniques for efficient motion planning”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 489–494.

[87] Máximo A Roa and Raúl Suárez. “Grasp quality measures: review and performance”. In: *Autonomous Robots* 38.1 (2015), pp. 65–88.

[88] Tanner Schmidt, Richard A Newcombe, and Dieter Fox. “DART: Dense Articulated Real-Time Tracking.” In: *Robotics: Science and Systems*. Vol. 2. Berkeley, CA. 2014.

[89] Philipp S Schmitt et al. “Modeling and planning manipulation in dynamic environments”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 176–182.

[90] Philipp S Schmitt et al. “Planning Reactive Manipulation in Dynamic Environments”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 136–143.

[91] John Schulman et al. “High-dimensional Continuous Control Using Generalized Advantage Estimation”. In: *CoRR* abs/1506.02438 (June 2015). arXiv: `1506.02438 [cs.LG]`.

[92] John Schulman et al. “Motion planning with sequential convex optimization and convex collision checking”. In: *IJRR* 33.9 (2014), pp. 1251–1270.

[93] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[94] Pierre Sermanet et al. “Time-contrastive networks: Self-supervised learning from video”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[95] Zhe Su et al. “Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor”. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2015, pp. 297–303.

[96] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. “The open motion planning library”. In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[97] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT Press, 1998.

[98] Garrett Thomas et al. “Learning robotic assembly from CAD”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE. 2018, pp. 1–9.

[99] Jonathan Tremblay et al. “Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects”. In: *arXiv preprint arXiv:1809.10790* (2018).

[100] Nikolaus Vahrenkamp et al. “Humanoid motion planning for dual-arm manipulation and re-grasping tasks”. In: *IROS*. IEEE. 2009.

[101] Nikolaus Vahrenkamp et al. “Visual servoing for dual arm motions on a humanoid robot”. In: *2009 9th IEEE-RAS International Conference on Humanoid Robots.* IEEE. 2009, pp. 208–214.

[102] Nikolaus Vahrenkamp et al. “Visual servoing for humanoid grasping and manipulation tasks”. In: *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2008, pp. 406–412.

[103] Jacob Varley et al. “Shape completion enabled robotic grasping”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. 2017, pp. 2442–2447.

[104] Mel Vecerik et al. “A practical approach to insertion with variable socket position using deep reinforcement learning”. In: *2019 International Conference on Robotics and Automation (ICRA).* IEEE. 2019, pp. 754–760.

[105] Matej Večerık et al. “Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards”. In: *arXiv preprint arXiv:1707.08817* (2017).

[106] Ulrich Viereck et al. “Learning a visuomotor controller for real world robotic grasping using simulated depth images”. In: *arXiv preprint arXiv:1706.04652* (2017).

[107] Rick Wagner. “Multi-linear interpolation”. In: *Beach Cities Robotics* (2008).

[108] Chen Wang et al. “DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion”. In: *arXiv preprint arXiv:1901.04780* (2019).

[109] Shaoxiong Wang et al. “3d shape perception from monocular vision, touch, and shape priors”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1606–1613.

[110] David Watkins-Valls, Jacob Varley, and Peter Allen. “Multi-modal geometric learning for grasping and manipulation”. In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2019.

[111] David Watkins-Valls, Jacob Varley, and Peter Allen. “Multi-modal geometric learning for grasping and manipulation”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7339–7345.

[112] Jonathan Weisz and Peter K Allen. “Pose error robust grasping from contact wrench space metrics”. In: *2012 IEEE international conference on robotics and automation*. IEEE. 2012, pp. 557–562.

[113] William J Wilson, CC Williams Hulls, and Graham S Bell. “Relative end-effector control using cartesian position based visual servoing”. In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 684–696.

[114] Bohan Wu, Iretiayo Akinola, and Peter K Allen. “Pixel-Attentive Policy Gradient for Multi-Fingered Grasping in Cluttered Scenes”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019.

[115] Bohan Wu et al. “MAT: Multi-fingered adaptive tactile grasping via deep reinforcement learning”. In: *arXiv preprint arXiv:1909.04787* (2019).

[116] Manuel Wüthrich et al. “Probabilistic object tracking using a range camera”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 3195–3202.

[117] Huangsheng Xie et al. “Research on visual servo grasping of household objects for nonholonomic mobile manipulator”. In: *Journal of Control Science and Engineering* 2014 (2014), p. 16.

[118] Xinchen Yan et al. “Learning 6-DoF grasping interaction via deep geometry-aware 3D representations”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–9.

[119] Xinyu Ye and Shan Liu. “Velocity Decomposition Based Planning Algorithm for Grasping Moving Object”. In: *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*. IEEE. 2018, pp. 644–649.

[120] Brayan S Zapata-Impata, Pablo Gil, and Fernando Torres. “Non-Matrix Tactile Sensors: How Can Be Exploited Their Local Connectivity For Predicting Grasp Stability?” In: *arXiv preprint arXiv:1809.05551* (2018).

[121] Andy Zeng et al. “Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning”. In: *arXiv preprint arXiv:1803.09956* (2018).

[122] Andy Zeng et al. “Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[123] Yuke Zhu et al. “Reinforcement and imitation learning for diverse visuomotor skills”. In: *arXiv preprint arXiv:1802.09564* (2018).