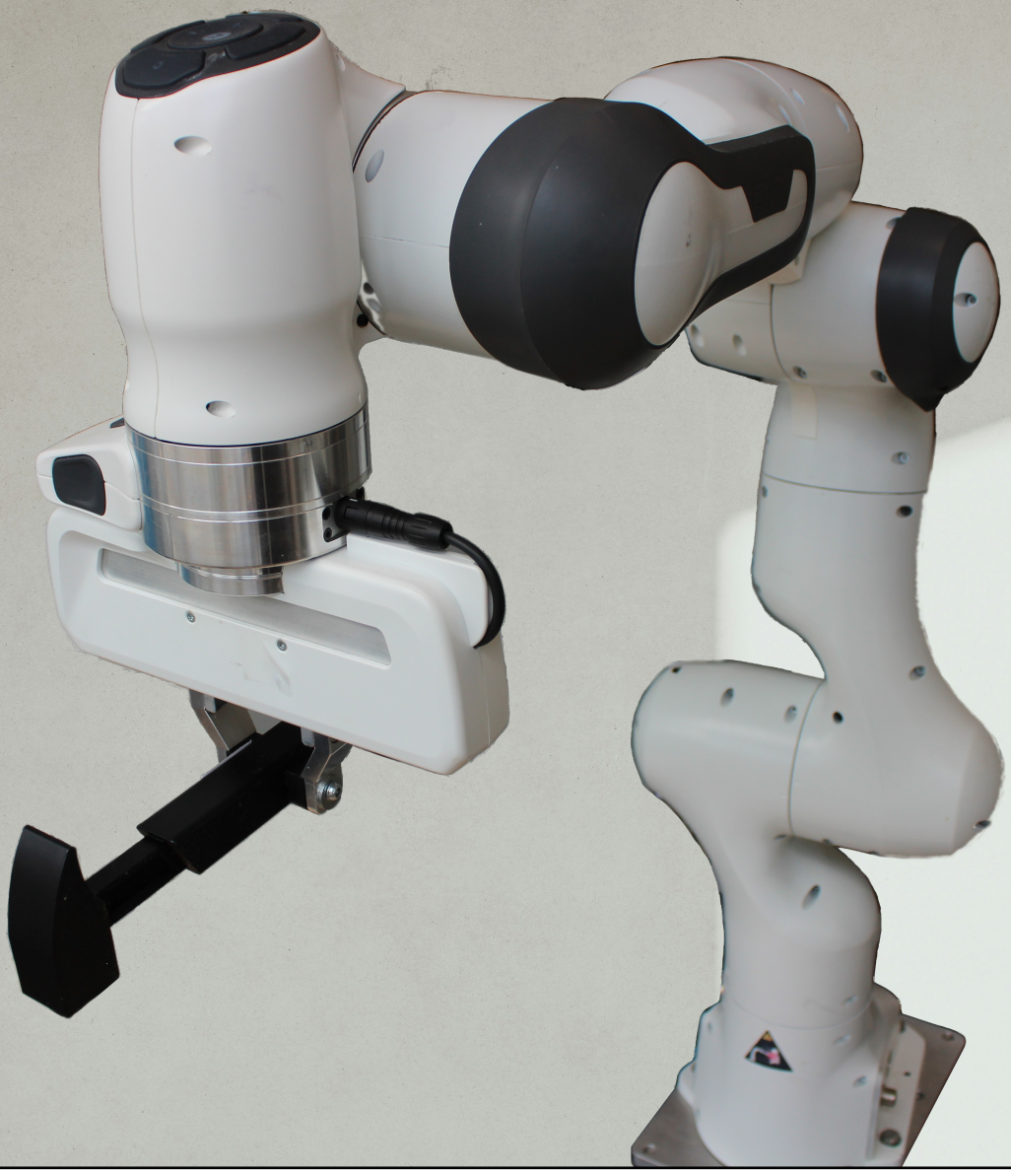
Increasing Robot Effective Workspace through Tool-Use

Minh Tâm Kevin Tran

MSc Thesis



TU Delft Delft University of Technology

Cognitive Robotics

Increasing Robot Effective Workspace through Tool-Use

Master of Science Thesis

For the degree of Master of Science in Cognitive Robotics at Delft University of Technology

Minh Tâm Kevin Tran

August 12, 2026

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Delft University of Technology
Department of
Mechanical Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical Engineering (ME) for acceptance a thesis entitled

Increasing Robot Effective Workspace through Tool-Use

by

Minh Tâm Kevin Tran

in partial fulfillment of the requirements for the degree of

Master of Science.

Dated: August 12, 2026

Supervisors:

Dr. Carlos Hernandez Corbato

Dr. Burak Şişman

Readers:

Dr. Sebastijan Dumančić

Abstract

A fixed-base robot arm can only reach objects within its physical workspace. When an object lies beyond this reach, the robot can use a held tool to pull the object closer before grasping it. This thesis studies how much a tool extends the reachable workspace of a Franka Panda robot arm (7 degrees of freedom).

A symbolic task and motion planning (TAMP) system, built from PDDL, a behaviour tree, and MoveIt2, plans and executes a drag action in which the robot uses a tool to pull an out-of-reach object toward itself. Two tool shapes, a hammer and a T-shape, are compared against a no-tool baseline. All trials run in MuJoCo simulation. Objects are placed on a systematic polar grid, spanning a distance of 0.35 to 0.90 m from the robot and an angle of 0 to 60 degrees, giving 1326 trials in total. The main measures are whether a plan can be found and executed, whether the object actually reaches the target, and the robot's mean manipulability (a measure of how far the arm's joints are from a singular configuration) during each trial.

The results show a clear trade-off. Without a tool, the robot succeeds almost every time at close range but fails completely beyond about 0.80 m. With a tool, success is lower at close range, because grasping and dragging add extra steps that can fail, but success increases beyond about 0.60 m, a distance the arm cannot reach directly. This crossover, where tool use becomes worthwhile, is the central finding of the thesis. A logistic regression model confirms this pattern statistically and shows that the two tool shapes extend reach by a similar amount, but differ in how much their success depends on the arm's manipulability: hammer performance is strongly tied to manipulability, while T-shape performance is not. This difference is due to how each tool interacts with the environment during the drag.

A smaller set of real-world experiments with the physical robot and the hammer tool supports the simulation results and shows that a collision-related failure mode seen with the T-shape in simulation does not appear in practice. Overall, the thesis shows that even a simple, prescribed tool-use action, planned by a symbolic planner, can meaningfully extend what a fixed-base manipulator can reach, and that the geometry of the tool shapes how reliably this works.

Table of Contents

1 Introduction 1
- 1.1 Research Questions . . . 2
- 1.2 Contributions . . . 2

2 Background 5
- 2.1 Task and Motion Planning . . . 5
- 2.2 Affordances . . . 6
- 2.3 Robot manipulability . . . 7

3 Related Work 9
- 3.1 Affordance-Based Reasoning in TAMP . . . 9
- 3.2 Tool-Use Planning . . . 9
- 3.3 Extending the Reachable Workspace Through Tool Use . . . 10

4 TAMP Architecture for Tool Use 13
- 4.1 Design Rationale . . . 13
- 4.2 System Overview . . . 15
- 4.3 Symbolic Planning Layer . . . 15
  - 4.3.1 PDDL Domain . . . 15
  - 4.3.2 Problem Specification and Tool Conditions . . . 16
  - 4.3.3 Plan Generation . . . 17
- 4.4 Plan Execution Layer . . . 17
  - 4.4.1 Behaviour Tree Executor . . . 17
  - 4.4.2 Behaviour Tree Leaf Nodes . . . 17
  - 4.4.3 move_and_pick and move_and_place . . . 18
  - 4.4.4 drag_closer . . . 19
- 4.5 Motion Planning and Execution . . . 20

4.6 Robot Manipulability . . . 21
4.7 Outcome and Manipulability Logging . . . 21
4.8 Deployment: Simulation and Real Robot . . . 22

5 Methodology 23
5.1 Assumptions . . . 23
5.2 Evaluation Metrics . . . 24
5.3 Experimental Design . . . 25
5.3.1 Tools and dimensions . . . 25
5.4 Configuration Space and Trial Design . . . 25
5.5 Statistical Analysis . . . 26
5.6 Sample size determination . . . 27

6 Results 29
6.1 Task Success Rate . . . 29
6.1.1 Task success vs. Object Orientation . . . 31
6.2 Logistic Regression Analysis . . . 31
6.2.1 Task success vs. Tool . . . 32
6.2.2 Manipulability Analysis . . . 35
6.3 Manipulability Across the Workspace . . . 38
6.4 Correlation Matrix . . . 38

7 Discussion 43
7.1 Effective Workspace Extension . . . 43
7.2 Tool-Shape influence . . . 44
7.3 Manipulability as a predictor of Task Success . . . 44
7.4 Manipulability across the Workspace . . . 45
7.5 Comparison with Related Work . . . 45
7.6 Limitations and Future Work . . . 46
7.7 Summary . . . 46

8 Conclusion 49

A Real-World Experiments 51
A.1 Experimental Setup . . . 51
A.2 Task Success Rate . . . 51
A.3 Manipulability Analysis . . . 53
A.4 Failure Modes . . . 53
A.5 Comparison with Simulation . . . 54

Introduction

Tool use has enabled an unprecedented expansion of human capabilities and has been observed across species, from chimpanzees extracting termites with sticks to birds cracking nuts with stones [45]. In robotics, tool use offers can offer a similar opportunity. By using objects in the environment, a robot can extend its capabilities beyond its inherent constraints, enabling tasks that would otherwise be infeasible.

Industrial robotics has historically addressed this challenge through specialization, designing machines and end-effectors for narrowly defined tasks. While effective in structured environments, this approach limits flexibility. Service robotics, by contrast, demands that robots operate in open-ended, partially observable environments, reasoning about their own capabilities and how to interact with a dynamic world. Achieving this requires combining high-level task reasoning with continuous motion planning, this problem is formalized as Task and Motion Planning (TAMP) [21].

Classical TAMP approaches, whether sampling-based, optimization-based, or hybrid, provide strong theoretical guarantees but often suffer from combinatorial explosion when enumerating actions and states in complex environments [21].

One way to keep this search manageable is affordance-based reasoning. Clasically, Gibson [24] defined affordances as the action possibilities that an environment offers to an agent, given the agent's own capabilities. An affordance is therefore not a property of an object on its own, but a relationship between the agent and the environment. Barth-Maron et al. [4] give a concrete example of this benefit: incorporating affordances as supplementary knowledge within a Markov Decision Process allows irrelevant actions to be pruned, substantially reducing the state-space explored during planning. Different formal definitions of affordances have been explored for robotics and will be discussed in section 2.2.

This thesis builds a TAMP pipeline that combines PDDL-based symbolic planning (PlanSys2), behaviour trees, and motion planning with MoveIt2, running on a Franka Emika Panda arm (7-DOF) in Mujoco simulation as well as in real life. In this pipeline, affordances play a supporting role: they are encoded as binary preconditions in the PDDL domain, used to check whether a tool-use action can be applied to a given object. The pipeline plans and executes a `drag_closer` action, in which the robot picks up a tool and uses it to drag an object that is otherwise out of reach back into its

workspace. Two tool shapes, a hammer and a T-Shape, are compared against a no-tool baseline in which the robot tries to reach the object directly.

Goal. The goal of this thesis is to characterise how tool use extends the robot's effective reachable workspace, and to quantify where, in that workspace, using a tool starts to outperform direct reach for retrieving an object. To do this, object positions are sampled systematically over a polar grid: a radial distance from $r \in [0.35, 0.90]$ m and an angle from $\theta \in [0, \pi/3]$ measured from the robot's base. Radial distance is used because the robot studied here, the Franka Emika Panda, is a fixed-base arm whose reach is naturally described as a distance from its base. The thesis's contributions are based on this type of manipulator and do not directly extend to mobile robots or to arms without rotating joints. Sampling this grid gives 1326 trials in total. For each trial, two metrics are measured.
Task success rate is the fraction of trials in which the robot completes its full planned action sequence and successfully retrieves the object.
Manipulability is a measure of how freely the arm can move at a given joint configuration: it drops toward zero near a kinematic singularity, for example when the arm is fully extended, and is high when the arm can move easily in any direction. It is quantified using Yoshikawa's index $\bar{w}$ (introduced in section 2.3), averaged over each trial.

1.1 Research Questions

To analyze how tool-use changes the effective reachable workspace for the robot, the following research questions were formulated.

Main research question:

RQ.1: How can tool-use be integrated into a PDDL-based TAMP pipeline for robot manipulation to extend its effective workspace?

Sub-question:

SQ.1: How does tool-use affect the effective workspace of the robot?
SQ.2: What is the relationship between manipulability and success rate, and how does tool use affect this?
SQ.3: How does tool use affect the relationship between radial distance and success rate?

1.2 Contributions

This thesis makes the following contributions:

- A TAMP pipeline, combining PDDL-based symbolic planning, behaviour tree execution, and MoveIt2 motion planning, that plans and executes a tool-mediated

`drag_closer` action on a Franka Emika Panda arm, in simulation and on the physical robot.

- A systematic, characterisation of task success across 1326 trials, identifying the kinematic transition zone (~0.55–0.65 m) at which tool use starts to outperform direct reach for a pick-and-place task, and a logistic regression model that gives an algebraic crossover radius for each tool condition (SQ.1, SQ.3).
- A quantified relationship between manipulability and task success, including a per-phase analysis showing that tool-condition failures occur in the drag phase (SQ.2).
- A comparison between the two tool geometries (hammer and T-Shape), showing differences in how they extend reach and how sensitive they are to arm manipulability.

Background

This chapter introduces three topics: task and motion planning (TAMP), affordances, and robot manipulability. Each plays a distinct role in this thesis. TAMP is the planning framework used to decide when and how the robot should use a tool to retrieve an out-of-reach object, combining a symbolic planner that chooses the sequence of actions with a motion planner that executes them on the arm. Affordances are the mechanism inside this framework that determines whether a tool-use action is applicable in the first place: a tool can only be used to drag an object closer if it has the right affordance. Manipulability is the metric used to explain, once a plan is executed, why some trials succeed and others fail: as the arm approaches its kinematic limits, its ability to move reliably degrades, which is directly relevant to a task where the robot must control a tool over an extended trajectory. Together, these three topics form the main parts of this thesis: TAMP plans and executes tool use, affordances gate when tool-use actions are available to the planner, and manipulability explains the quality of the resulting motion. The remainder of this chapter covers each topic in turn.

2.1 Task and Motion Planning

Task planning determines a sequence of actions that transforms an initial world state into a desired goal state. The task domain defines available actions, their preconditions, and their effects [12]. The dominant formalism is the Planning Domain Definition Language (PDDL) [18], a planning language based on first-order logic and rooted in the foundational STRIPS representation [17]. Other languages used for task planning include Answer Set Programming (ASP) [30] and Hierarachical Task Networks (HTN) [22], which are also based on first-order logic but have different semantics and expressiveness compared to PDDL. Extensions to PDDL exist to support these approaches including HTNs [37] and temporal planning [18].

The objective of motion planning is to compute a continuous, collision-free path of valid configurations for a robot from an initial configuration to a goal. The two dominant approaches are sampling-based methods [19] and optimization-based methods [39]. Sampling-based planners are probabilistically complete, in ther words guaranteed to find a solution if one exists given sufficient time, but may run indefinitely if none

exists. Optimization-based planners are often more efficient in practice but provide no completeness guarantees, and may converge to local minima [12].

TAMP integrates te symbolic task planning and continuous motion planning into a single framework. This is necessary because the two are tightly coupled: a motion planner may find a task planner's action infeasible due to kinematic constraints, requiring the task planner to revise its high-level plan. A TAMP solution is a sequence of actions $\pi = [a_1, \ldots, a_k]$, where each a_i carries preconditions, constraints τ_i, effects, and a state s_i.

A plan skeleton is a partial plan that contains a sequence of actions, without all parameters being specified. Determining the correct values for the parameters of a plan skeleton is called a hybrid constraint satisfaction problem (H-CSP) [21]. This H-CSP spans both continuous domains *(robot configurations, object poses, trajectories)* and discrete domains *(symbolic states)*.

Strategies for solving TAMP problems fall into three categories [21]:

Sequencing-first methods establish an action sequence before assigning motion paths [33];
Satisfaction-first methods sample parameter values and check constraint satisfaction [20];
Interleaved methods alternate between task-level decisions and motion-level feasibility checks in an integrated fashion [43].

2.2 Affordances

The concept of affordances originates in Gibson's cognitive psychology [24], describing the action possibilities that an environment offers an agent. These action possibilities fundamentally describe a relation between an object and the agent performing the action. Norman later distinguished *perceived affordances*, shaped by an agent's goals and experience, from *real affordances*, the action possibilities actually supported by the environment [34].

For robotics, Şahin et al. [41] defined an affordance as an acquired relation between an effect and an (entity, behavior) pair:

$$(\textit{effect},\ (\textit{entity},\ \textit{behavior}))$$

where *entity* is the relevant part of the environment, *behavior* is the robot's action and perception, and *effect* is the resulting change. This makes affordances practically computable for robot control. More recent formalisms further incorporate the agent's internal state: Cruz et al. [11] note that an affordance is only exploitable if the required conditions are met. For example, a robot fit with two grippers already holding objects in both grippers cannot grasp a third, rendering the *'grasping'* affordance temporarily unavailable.

In this thesis, affordances are represented as binary predicates in the PDDL domain, used as preconditions that determine whether a tool-use action is applicable to a given

object. This follows the broad approach of grounding planning operators in affordance predicates [31], and the use of functional affordances as planning preconditions for tool substitution [3]. Compared to the richer representations described above, this is a coarse simplification: a predicate such as `(has-affordance ?tool pull-affordance)` either holds or does not. The affordance representation is used to determines whether a tool is available to be used for a `drag_closer` action.

2.3 Robot manipulability

A robot manipulator operating near the boundary of its workspace does not simply reach a point or fail to reach it, there is a continuous degradation in the quality of motion as the robot approaches kinematic limits. A configuration $\mathbf{q}$ where $\det(\mathbf{J}(\mathbf{q})) = 0$ is a called a singularity. At such points, the robot loses the ability to produce motion in one or more directions. These singularities commonly arise when the manipulator is fully extended or when multiple joint axes align, leading to a reduction in degrees of freedom, similar to gimbal lock [10].
When a robot approaches a singular configuration, issues occur where the robot requires very high joint velocities to achieve a small displacement at its end-effector. At singularity these required joint velocities approach infinity.
In the context of tool-use, this is particularly consequential. A drag action requires the robot to maintain a controlled motion over its trajectory while holding a tool. If the robot approaches singular configuration during the motion, it becomes unreliable even if the target pose is technically within the robot's workspace. This motivates the need for a metric that captures not just reachability (measured in mm), but the quality of manipulation at a given configuration.

Yoshikawa's manipulability index [47] is the most widely accepted performance metric to measure manipulability in manipulators. To obtain the set of available end-effector cartesian velocities, one can make a mapping from the N-dimensional joint velocity space to end-effector cartesian velocity space. This mapping spans an elipsoid around the end-effector and thus maps all possible end-effector velocities. If this elipsoid is very spherical, meaning its radii are close in magnitude, the end effector can reach any velocity. On the other hand, if one radius is near zero, this indicates that the end effector is not able to reach any velocities along that axis [10].
Yoshikawa's manipulability index ($m(\mathbf{q})$) is a dimensionless measure that describes how spherical this elipsoid is, for example as a ratio between the largest and smallest radius and is defined as

$$w(\mathbf{q}) = \sqrt{det[J(\mathbf{q})J(\mathbf{q})^T]}$$

When $m(\mathbf{q}) = 0$, the robot is in a singular configuration. This method is used to calculate both the translational velocity elipsoid as the rotational velocity elipsoid. A robot configuration can be in a state with low translational manipulability, but higher rotational manipulability.

Other manipulability measures. Yoshikawa's ellipsoid is not the only way to describe manipulability. Lee [26] points out that the ellipsoid is derived assuming joint velocity limits form a sphere in joint space. In practice, each joint can mov separately and has its own separate velocity limit, so the true task-space region is a box rather than a sphere. This means the ellipsoid does not transform the exact joint limits into task space, and can therefore give an inaccurate measure of dexterousness, including identifying the wrong direction of best motion.

To address this, Lee [26] describes the *manipulability polytope*, which transforms the actual joint velocity limits directly into task space, instead of approximating them with a sphere. This produces an exact representation of the reachable end-effector velocities, at the cost of a more complex calculation, especially for redundant manipulators with many joints.

Despite this limitation, the manipulability ellipsoid remains the most widely used measure of manipulability, due to its simple, closed-form calculation [47]. For this reason, Yoshikawa's manipulability index is adopted in this thesis as the metric for robot manipulability.

Related Work

This chapter reviews work related to the three parts of this thesis: how affordances are used inside TAMP, how tool use is planned, and how tool use has been shown to extend a robot's reach. It first covers affordance-based reasoning in TAMP, showing how affordances have been used to keep planning manageable and to decide which actions or tools apply to a given task. It then covers tool-use planning more broadly, comparing systems that plan and execute tool-use actions in different ways. Finally, it covers work that specifically studies how tool use extends a robot's reachable workspace, which is the setting closest to this thesis, and against which the results in chapter 6 are later compared.

3.1 Affordance-Based Reasoning in TAMP

Early integration of affordances with PDDL-based planning was demonstrated by Lörken and Hertzberg [31], who used affordances to ground symbolic actions for a crane-like robot activating switches using toy blocks. Their approach focuses solely on affordances without differentiating between specific objects, which limits object-specific selection.

Awaad et al. [3] identified three key applications of affordance-based reasoning: object substitution via tool use, object substitution to enhance performance, and action substitution based on the robot's morphology. They specifically distinguish between **functional affordances** and **affordance cues**. Functional affordances, a subset of Norman's perceived affordances [34], allow tasks to be described by their function rather than requiring exhaustive low-level specification, enabling significant reduction of the planning action space. This distinction is relevant to tool-use planning, where affordance conditions can serve as preconditions for determining which tools are eligible for a given action.

3.2 Tool-Use Planning

Car et al. [7] evaluated affordance-aware versus affordance-agnostic planning on single- and multi-task tool-use scenarios, demonstrating a significant improvement in task success rate when an affordance model was included. Their architecture used LLMs for

both high-level planning and step-level action translation. The present work differs in that affordances are grounded directly within a PDDL-based TAMP pipeline executed on a physical robot manipulator, without relying on LLMs for plan generation, which isolates the contribution of the affordance representation from the planner itself.

Ti et al. [42] incorporated a directional manipulability index into the cost function of an optimization-based motion planner (ADMM-iLQR) for robot hammering tasks. Their work highlights how tool orientation and end-effector configuration affect trajectory quality in tool-use scenarios.

Xu et al. [46] proposed RoboTool, a system that accepts natural language task descriptions and generates executable robot control code through four stages: task analysis, strategy planning, parameter calculation, and code generation. The system explicitly handles physical constraints such as workspace limits and demonstrates behaviors including sequential tool use and tool improvisation to retrieve objects beyond the arm's standard reach. While the motivation is closely related to the present work, RoboTool relies on an LLM planner and validates performance through individual demonstrations rather than systematic evaluation across sampled workspace positions.

3.3 Extending the Reachable Workspace Through Tool Use

The use of tools to compensate for kinematic reach limitations has been explored in several works.

Nguyen and Choe [32] trained a deep reinforcement learning controller on a simulated jointed arm that must grasp a T-, I-, or L-shaped tool and drag an object to a target location. The controller operates on raw visual and proprioceptive input with minimal reward shaping, and a range of behaviors emerged from training, including wide sweeps, throwing, and contact correction. This work is the most directly comparable to the present study in terms of task structure, as it shares the same core setup of tool-mediated retrieval of an object placed outside direct reach. However, it operates in 2D simulation using a reinforcement learning approach and does not characterise how the choice of tool affects task success as a function of spatial position across the workspace.

Dong et al. [13] proposed a method for jointly selecting candidate tools and planning contact-rich manipulation trajectories on a Franka Panda arm, with the objective of maximising robustness against execution disturbances. Their approach uses an energy-based robustness metric to evaluate tools such as umbrellas and hangers for pulling objects from outside the arm's standard workspace into a reachable region, which is geometrically the same problem as the drag-closer action considered in this thesis. Their contribution lies in the joint tool-selection and motion planning formulation optimised for disturbance tolerance, whereas the present work focuses on characterising how different tool geometries alter the spatial distribution of task success, evaluated systematically over the workspace.

Lee et al. [28] built a real-world, dual-arm system that retrieves out-of-reach objects by pushing or dragging them with a stick, an L-hook, or a Y-hook. A fine-tuned LLM generates the high-level symbolic plan, and a geometric affordance model computes

the best contact point on the tool before a manoeuvrability-driven stepping controller executes the motion. Their system also handles environments containing wall obstacles, where the object must first be stepped around an obstacle before it can be dragged into reach. Their notion of manoeuvrability describes the quality of the contact point between tool and object, modelled as a Gaussian that peaks at the midpoint of a tool segment. This differs from the manipulability measure used in this thesis, which describes the arm's kinematic distance from a singular configuration rather than the tool-object contact geometry. This thesis however uses Yoshikawa's manipulability to capture how well the arm's configuration can reach different points around the end effector. A further difference lies in how affordances are represented. Lee et al.'s affordance model is part of the planner, it actively computes where on the tool to make contact and derives the motion from this. In this thesis, affordances are only used as binary preconditions to decide whether a tool-use action may be applied.

Lee et al. also compare their LLM-based planner against a PDDL-based baseline [25]. The PDDL baseline achieves 99% success when tested within the distribution it was designed for, but drops to 42% and 6% under two zero-shot, out-of-distribution workspace changes, while their own LLM-based planner degrades far less under the same changes. This result is direct evidence that symbolic planning does not generalise well outside the conditions it was built for, a limitation that is relevant to the PDDL-based planner used in this thesis as well.

TAMP Architecture for Tool Use

This chapter describes the robotic system developed for this thesis: a task and motion planning (TAMP) pipeline that combines PDDL-based symbolic planning with PlanSys2, behaviour tree execution, and motion planning with MoveIt2, running on a Franka Emika Panda arm (7-DOF) in MuJoCo simulation and on the physical robot. The system accepts a tool condition and an object pose as input, plans a sequence of high-level actions, executes that sequence on the arm, and records the outcome. Affordance predicates in PDDL determine when `drag_closer` is applicable, integrating tool-use actions into the TAMP pipeline; this system is what generates the trial data used to answer SQ.1–SQ.3 in Chapters 6 and 7. Chapter 5 (Methodology) describes how this system is ran across multiple trials. This chapter describes the system architecture.

4.1 Design Rationale

This task, retrieving an object that lies beyond direct reach, was chosen because it is the simplest task in which tool use can extend a robot's effective workspace, and it is the task studied in the closest related work [32, 28, 13]. It isolates reach as the variable under study, without additional parameters as clutter, occlusion or multi-object planning. It also reduces to a single continuous parameter, the object's radial distance, which can be sampled systematically as required by SQ.1–SQ.3.

The task requires a robot arm to retrieve a box that may lie beyond its direct reach, optionally with the help of a tool that extends its effective reach. This task has two different parts. The first part is discrete: decide whether a tool is needed, and if so, decide the order of picking up the tool, dragging the box closer, and then picking up the box. The second part is continuous: compute joint trajectories that move the end effector through free space without collision, while satisfying the kinematic limits of the arm.

TAMP architectures separate these two concerns into a symbolic planning layer and a motion planning layer, and maps each symbolic action to one or more motion primitives. This separation lets the symbolic planner reason about the structure of the task (which actions, in which order) without modelling continuous kinematics, while

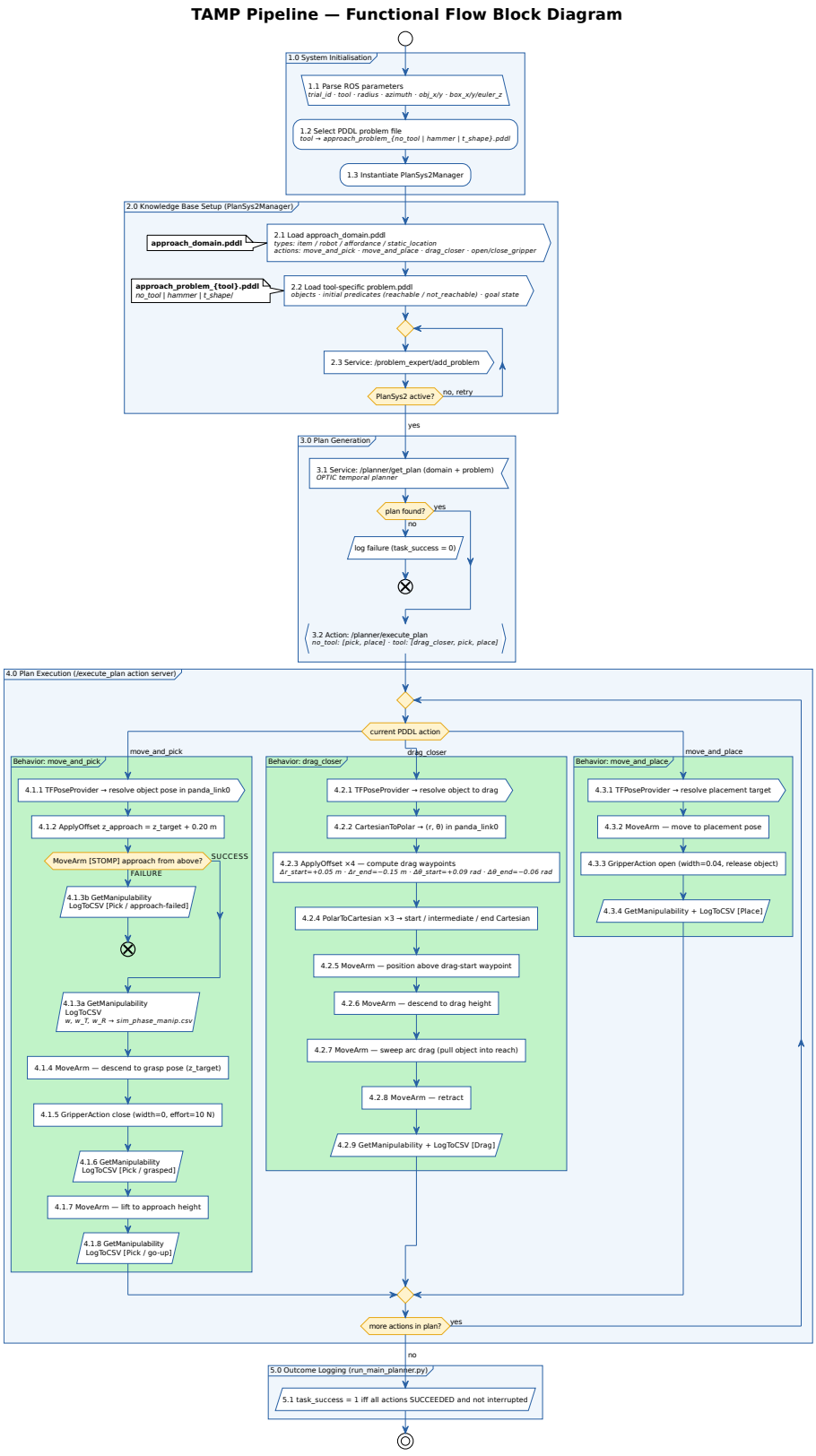


Figure 4.1: Functional flow of the TAMP pipeline, from trial configuration to outcome logging.

the motion planner is invoked only with concrete, fully specified goals. The architecture adopted here follows this pattern with three layers: a PDDL domain and problem solved by PlanSys2, a behaviour tree (BT) executor that expands each planned action into a sequence of motion primitives, and MoveIt2, which performs collision-free motion planning and executes the resulting trajectory on the arm.

4.2 System Overview

The pipeline is organised into three layers, summarised in Table 4.1 and described in the following sections. Figure 4.1 shows the functional flow of the system, from trial configuration to outcome logging.

Table 4.1: Layers of the TAMP architecture

Layer	Implementation	Responsibility
Symbolic planning	PlanSys2 + OPTIC	Decide which actions, in which order
Plan execution	BehaviorTree.CPP v3	Expand each action into motion primitives
Motion planning & control	MoveIt2 + STOMP	Compute and execute joint trajectories

A Python orchestration node, `PlanSys2Manager`, sits above all three layers. It loads the PDDL domain and problem into PlanSys2, requests a plan, and forwards the plan to an executor action server for execution. A second script, `run_main_planner.py`, wraps this manager with trial-specific configuration, failure detection, and CSV logging used in the experiments of Chapter 5.

4.3 Symbolic Planning Layer

The symbolic planning layer decides which actions the robot should take, and in what order, to retrieve the object. It does not deal with motion at all: it works only with the PDDL domain and problem described below, and outputs a symbolic action sequence. This section covers the domain (what actions and predicates exist), the problem (how each tool condition is set up), and how PlanSys2 turns these into a plan.

4.3.1 PDDL Domain

The domain, `robot_arm` (defined in `approach_domain.pddl`), declares four types: `item` (movable objects, including tools), `static_location`, `robot`, and `affordance`. State is expressed with eight predicates: object location (`at`), whether the robot is holding an object (`holding`), whether an object is within direct reach (`reachable` and its negation `not_reachable`), gripper state (`gripper_opened`, `gripper_closed`), robot availability (`idle`), and tool capability (`has-affordance`). The `not_reachable` predicate exists because the OPTIC planner does not support negative preconditions or derived predicates; reachability and its negation are tracked as two separate predicates that are kept

consistent by the actions that change them, rather than being derived from a single `reachable` predicate at query time.

The domain defines five durative actions:

- **`move_and_pick(?r - robot, ?obj - item, ?loc - static_location)`**: picks up `?obj` at `?loc`, provided the gripper is open, the robot is idle, and `?obj` is reachable. After execution, the robot holds `?obj`, the gripper is closed, and `?obj` is no longer at `?loc`.
- **`move_and_place(?r, ?obj, ?loc)`**: the inverse of `move_and_pick`. Requires the robot to be holding `?obj` with the gripper closed; places `?obj` at `?loc` and opens the gripper.
- **`drag_closer(?r - robot, ?obj - item, ?tool - item)`**: the drag action. Requires `?obj` to be `not_reachable`, `?tool` to be reachable and to hold the `pull-affordance` for the whole action (an `over all` condition), and the gripper to be open at the start. Its effect is purely symbolic: `?obj` becomes `reachable` (and is no longer `not_reachable`). The action does not change `?obj`'s `at` predicate; it only changes whether it can subsequently be picked up directly.
- **`open_gripper`** and **`close_gripper(?r - robot)`**: toggle gripper state independently of any object, used where the plan requires an explicit gripper action that is not bundled into a pick or place.

4.3.2 Problem Specification and Tool Conditions

Three tool conditions are evaluated in this thesis: `no_tool`, `hammer`, and `t_shape`. Each condition has its own PDDL problem file (`approach_problem_no_tool.pddl`, `approach_problem_hammer.pddl`, `approach_problem_t_shape.pddl`), selected at runtime from the `tool` ROS parameter. All three problems share the same object instances, a robot `panda`, a box `simple_box`, and a tool object, but differ in their initial state:

- In `no_tool`, `simple_box` is declared `reachable` and no tool predicate is asserted. The resulting plan consists of `move_and_pick` followed by `move_and_place`: a direct, three-action retrieval.
- In `hammer` and `t_shape`, `simple_box` is declared `not_reachable`, and the tool object is declared `reachable` with `has-affordance` for `pull-affordance`. The resulting plan inserts `drag_closer` before `move_and_pick`: a four-action retrieval in which the tool is first used to bring the box within reach.

This means the choice between a direct and a tool-mediated plan is made by the trial configuration, not discovered by the planner from continuous reachability information; reachability is asserted as a boolean fact per tool condition rather than computed from the sampled box pose.

4.3.3 Plan Generation

PlanSys2 exposes domain and problem expert services that hold the loaded domain and the current problem state, and a planner service that generates a plan on request. The `PlanSys2Manager` component (Section 4.2) calls `/planner/get_plan` with the domain and the current problem, and PlanSys2 invokes the OPTIC temporal planner Benton, Coles, and Coles [6] over them. If no valid plan is found, the request returns unsuccessfully, the trial is recorded as a failure, and execution does not proceed. Otherwise, the resulting ordered list of grounded actions is forwarded to the plan execution layer.

4.4 Plan Execution Layer

The plan execution layer takes the symbolic action sequence from the previous layer and carries it out on the arm. It does this by expanding each planned action into a behaviour tree of concrete steps, such as moving to a pose or opening the gripper. This section covers how PlanSys2 turns actions into behaviour trees and the specific trees for each of the five actions defined in section 4.3.1.

4.4.1 Behaviour Tree Executor

PlanSys2 executes plans through a BT.CPP v3-based executor: each grounded PDDL action in the plan is dispatched to a corresponding action server, implemented as a `plansys2_bt_actions::bt_action_node` process running a fixed behaviour tree XML file for that action. Five such processes are launched alongside PlanSys2, one per domain action (`move_and_pick`, `move_and_place`, `drag_closer`, `open_gripper`, `close_gripper`), each with its own BT XML tree under `behavior_trees_ompl/`. The `drag_closer` process loads a different tree depending on the tool, `drag_closer_polar.xml` for `hammer` and `drag_closer_polar_t_shape.xml` for `t_shape`, since the two tools differ in the TF frame names used for tool-offset compensation (Section 4.4.4). Each process also exposes its own ZMQ publisher and server ports for Groot tree-state monitoring [5]; because these ports are plain TCP sockets and not scoped by the ROS domain ID, parallel instances of the pipeline (used to run trials concurrently, Chapter 6) are each assigned a disjoint port block.

This per-action-server structure means the symbolic plan, a flat sequence of action names and parameters, is expanded into five fixed, hand-authored behaviour trees, one of which is selected and parameterised at execution time for each planned action. The behaviour trees themselves, not the symbolic planner, contain the procedural detail of how each action is carried out.

4.4.2 Behaviour Tree Leaf Nodes

The behaviour trees are built from a set of custom BT.CPP leaf nodes, registered centrally in `bt_node_registration.cpp`. Each node is implemented as a single C++ class:

- **TFPoseProvider**: takes in an object frame and returns a TF pose relative to panda_link0 by TF lookup. This is node returns the position and is used in all behaviour trees evaluated in this thesis.
- **ArucoPoseProvider**: an alternative pose node that detects an object's pose directly from ArUco markers rather than via TF, registered alongside TFPoseProvider. This was not used in the simulation experiments in chapter 5, however is used in the real-world experiments in appendix A.
- **MoveArm**: requests a motion plan and execution through MoveIt2's MoveGroup action (Section 4.5). This is the node responsible for actually moving the arm to a target pose.
- **GripperAction**: sends an open or close command, with a target width and maximum effort, to the gripper action server.
- **ApplyOffset**: adds a scalar offset to a numeric blackboard value, used to compute hover heights and drag-trajectory waypoints from a base pose.
- **CartesianToPolar** and **PolarToCartesian**: convert between the Cartesian frame and the polar frame centred on the robot base (panda_link0), used by drag_closer (Section 4.4.4).
- **SetTCPOffset**: compensates a target pose for the offset between the gripper's tool centre point (TCP) and a tool frame, so the tool tip, not the gripper, reaches the intended pose (Section 4.4.4).
- **CaptureOffset**: records the current offset between the TCP and a tip frame, for later reuse.
- **GetManipulability**: computes the Yoshikawa manipulability index at the current arm configuration (Section 4.6).
- **LogToCSV**: writes the current manipulability readings, with trial metadata and a phase label, to a CSV file (Section 4.7).

Most of these nodes are synchronous (BT::SyncActionNode); MoveArm and GripperAction are asynchronous action-client nodes (plansys2::BtActionNode) that report RUNNING until the underlying MoveIt2 or gripper action server returns a result.

4.4.3 move_and_pick and move_and_place

The move_and_pick tree resolves the target object pose with TFPoseProvider, then attempts an approach move with MoveArm to a position offset 0.2 m above the object. This approach attempt is wrapped in a Fallback node so that manipulability is logged and the action still proceeds to a controlled failure state if the approach move itself fails, rather than aborting the tree without a record. After this, the arm moves down to the grasp height, the gripper closes, and the arm retracts to the same hover offset,

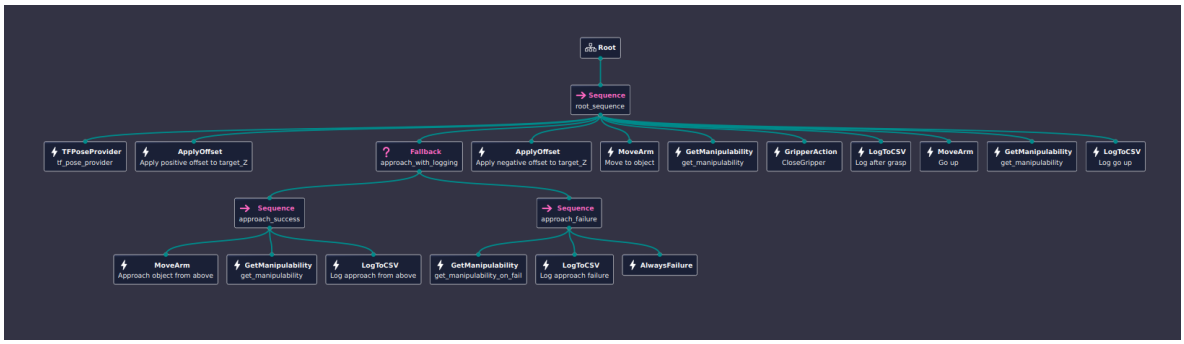


Figure 4.2: Behavior tree for `move_and_pick` action. Contains leaf nodes for obtaining object poses, moving the arm and logging success and manipulability.

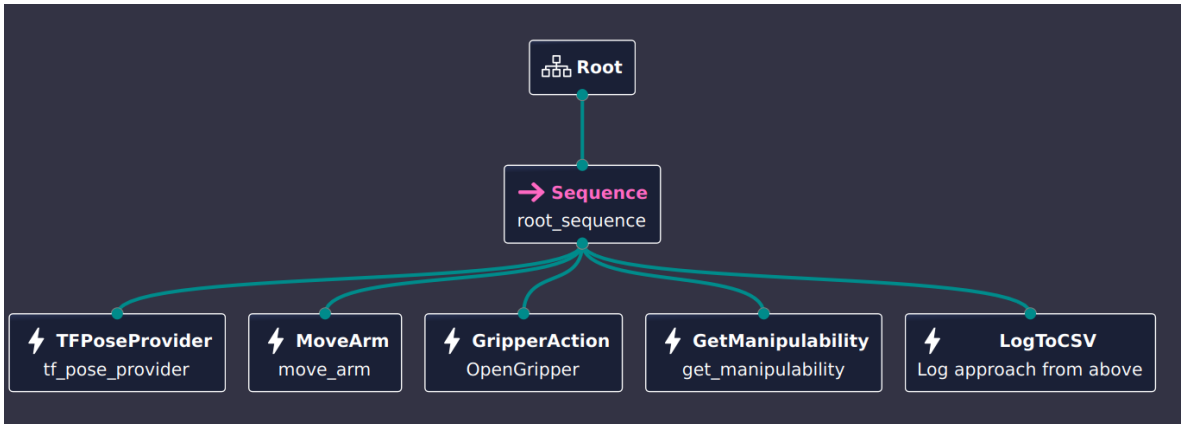


Figure 4.3: Behavior tree for `move_and_place` action. Contains leaf nodes for object poses, descending down and opening the gripper.

with manipulability logged with `GetManipulability` and `LogToCSV` after each motion segment. The full behavior tree for this action is shown in fig. 4.2. `move_and_place` follows the same pattern in reverse: approach above the placement target, descend, open the gripper, and retract, see fig. 4.3.

4.4.4 drag_closer

The `drag_closer` tree implements a drag action to bring objects closer using tools and is the most elaborate of the five trees. Figure 4.4 shows the `drag_closer` action in the

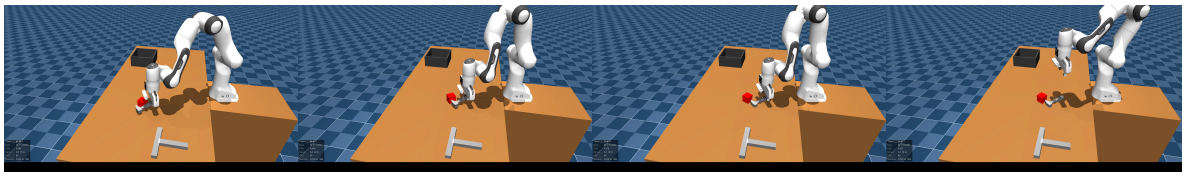


Figure 4.4: `drag_closer` action using the hammer as a tool. The red box is to be picked and placed inside the grey bin.

MuJoCo simulation using the hammer as a tool. It contains two stages: grasping the tool, then using the grasped tool to drag the box within reach. A simplified version of the behavior tree is shown in fig. 4.5.

The drag geometry is computed in the polar frame centred on `panda_link0` (the same frame used for trial sampling, Chapter 5). The tree resolves the box pose, converts it to polar coordinates (r, θ) with `CartesianToPolar`, and derives three waypoints along the line from the tool to the box: a start point at $r + 0.05\,\text{m}$ and $\theta + 0.09\,\text{rad}$, an intermediate point at the same radius and the unmodified angle, and an end point at $r - 0.15\,\text{m}$ and $\theta - 0.06\,\text{rad}$. These three points, converted back to Cartesian coordinates with `PolarToCartesian`, define a short sweeping arc that starts beyond the box, passes under it, and ends with the box pulled toward the robot.

Because the tool tip, not the gripper, must trace this arc, every waypoint is passed through `SetTCPOffset` before being sent to `MoveArm`. `SetTCPOffset` looks up the live transform between the TCP frame and the tool tip frame (`hammer_head` for the hammer condition) and computes the gripper pose $T_{\text{world,tcp}} = T_{\text{world,toolgoal}} \cdot T_{\text{tcp,tool}}^{-1}$ that places the tool tip at the intended waypoint. Looking this transform up live, rather than assuming a fixed grasp offset, means the compensation is correct regardless of small variation in how the tool was grasped.

After the tool is grasped and the TCP-to-tool offset is captured with `CaptureOffset`, the tree moves the arm through a hover pose above the drag start, then down to a working height under the box, then through the intermediate point to the end point, opens the gripper, and retracts to a hover pose above the end point. Manipulability is logged with `GetManipulability` and `LogToCSV` after each of these segments, using the tool tip frame (rather than the TCP) as the reference, so that the recorded manipulability reflects the kinematic configuration relevant to the tool, not the gripper alone.

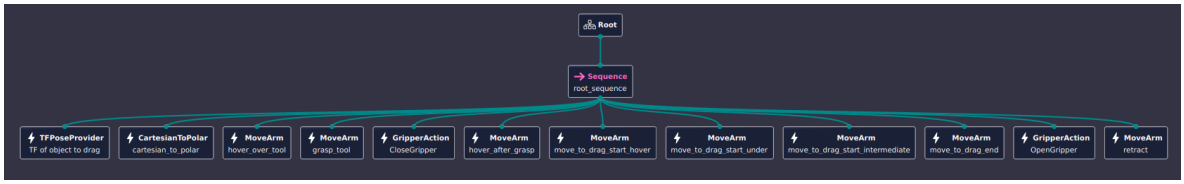


Figure 4.5: Simplified behavior tree for `drag_closer` action. Nodes that are not visualized due to space: `ApplyOffset`, `TFPoseProvider`, `LogToCSV`, `getManipulability`

4.5 Motion Planning and Execution

All Cartesian motion requests pass through the `MoveArm` behaviour tree node, which targets the `panda_manipulator` MoveIt2 planning group and plans with the STOMP pipeline. Two design choices in `MoveArm` are specific to this system and are described here because they affect what fraction of requested motions succeed, and therefore the results reported in Chapter 6.

Cartesian seeding of STOMP. Before invoking STOMP, `MoveArm` calls `/compute_cartesian_path` toward the target pose. This serves two purposes: firstly, the STOMP motion planner requires a goal in joint-space, while the `TFPoseProvider` provides a goal in cartesian space. Therefore we call `/compute_cartesian_path` to get a goal in joint-space. Secondly it supplies a joint trajectory to STOMP as a seed, which STOMP uses as a warm start towards the target rather than from an uninformed start. If the Cartesian path computation reaches a fraction of zero of the requested path, no joint goal can be formed and the node fails immediately, without invoking STOMP.

Orientation constraint. A path constraint keeps the end-effector's Z-axis pointing downward throughout each motion, with a tolerance of $\pi/3$ (60 °) around the X and Y axes and free rotation about Z. This reflects the task: the gripper must remain oriented for a top-down grasp or release at all times, while the wrist is free to rotate about the vertical axis to reach a given approach direction.

The planning request also fixes a workspace volume ($x \in [0.1, 1.0]$, $y \in [-1.0, 1.0]$, $z \in [0.028, 0.6]$ m in the `panda_link0` frame), allows up to 20 planning attempts and 8 replanning attempts, and scales velocity and acceleration to 50% and 10% of maximum, respectively, to keep motion safe for execution on the physical arm as well as in simulation.

4.6 Robot Manipulability

The `GetManipulability` node computes the Yoshikawa manipulability index (w) [47] at the arm's current joint configuration, read from `/joint_states`. It builds the geometric Jacobian J of the `panda_arm` joint group at the `panda_hand_tcp` link through MoveIt2's robot model, and computes

$$w = \sqrt{\det\left(JJ^{\top}\right)}.$$

When a node further from the TCP is the relevant reference point, for example the tip of a grasped tool, the node accepts a fixed offset r from the TCP and extends the Jacobian's translational rows accordingly, $J'_{\text{lin}} = J_{\text{lin}} - [r]_{\times} J_{\text{ang}}$, before computing w. This is the mechanism used by `drag_closer` to report manipulability at the tool tip rather than at the gripper. The same computation is repeated using only the translational or only the rotational rows of J to report a translational and a rotational manipulability value alongside the full index.

4.7 Outcome and Manipulability Logging

Each trial produces two datasets at different levels of detail: one row per trial, and one row per waypoint within a trial.

The trial-level dataset, `executability_results.csv`, has one row per trial. It records the trial's configuration (tool, box pose, object pose), a binary *task_success*

flag set to one only if the plan completed in full, and, for failed trials, a short failure reason obtained by matching reported error messages against a set of patterns (MoveIt error codes, IK failures, joint limit violations, timeouts). This dataset is used for the executability and task-success analysis in Chapter 6.

The waypoint-level dataset, `manipulability.csv`, has one row per waypoint reached during execution. Each row stores the manipulability values at that waypoint and a phase label (for example `move_and_pick_after_approach_from_above` or `drag_closer_drag_end_under`) showing where in the action the reading was taken. This detail is needed for the phase-level manipulability analysis in Chapter 6: the trial-level outcome alone cannot show which part of an action had a failure or a low-manipulability configuration, only a per-waypoint record can.

The two datasets are written separately because they come from different parts of the pipeline, at different times. `executability_results.csv` is written once by `run_main_planner.py` after the trial ends, while `manipulability.csv` is written throughout execution by the `LogToCSV` behaviour tree node. The two are joined afterwards on trial ID for the combined analyses in Chapter 6.

4.8 Deployment: Simulation and Real Robot

The same TAMP pipeline, PDDL domain, behaviour trees, and orchestration code, runs against two different lower layers. In simulation, the arm is modelled in MuJoCo and connected to ROS2 through `ros2_control`, with `franka_moveit_config`'s simulation launch file bringing up MoveIt2 against the simulated hardware interface. On the physical robot, the same MoveIt2 and behaviour tree layers connect instead to the real `franka_hardware` interface over the Franka FCI. The simulation obtains object poses directly through TFs published by MuJoCo with the `TFPoseProvider` node (Section 4.5); on the real robot, the published TF frames are published by the `aruco_pose_provider` node (Chapter 5). Recovery from a hardware error or reflex state on the physical robot is handled separately, outside the TAMP pipeline, by a recovery script (`panda_recover.py`) that resets the Franka control interface before a trial is retried, this is used in the real-world experiments in appendix A.

To support running multiple trials concurrently, the pipeline can be launched with an `instance_id` that shifts the BT executor's Groot monitoring ports by a fixed block per instance, and a separate ROS domain ID per instance keeps each pipeline's ROS graph isolated from the others. Apart from this port and domain allocation, each running instance is otherwise a complete, independent copy of the architecture described in this chapter.

Methodology

This section describes the experiments conducted to evaluate the TAMP system, described in chapter 4, on effective workspace.

5.1 Assumptions

The following assumptions are made:

- **ASM.1**: A PDDL domain file and problem file are available.
- **ASM.2**: The robot's action capabilities are known a priori and defined as primitive actions in PDDL (e.g., `open_gripper`, `move_arm`).
- **ASM.3**: Object affordances within the scene are known and provided in PDDL.
- **ASM.4**: The grasp poses of objects within the scene are known via TFs.
- **ASM.5**: The tool tips of objects within the scene are known via TFs.
- **ASM.6**: The motion planner's collision scene contains only the robot model; surrounding objects such as the table and the manipulated box are not modelled as collision geometry.

ASM.1 and **ASM.2** assume that the robot's capabilities are predefined as a set of primitive actions, from which more complex behaviours can be composed. This is a common assumption in TAMP frameworks [46, 38] and allows the focus to remain on plan generation.

ASM.3 assumes that object affordances are available prior to planning. Affordance detection and learning have been extensively studied and have demonstrated varying levels of robustness and generalisation [8, 9, 29, 27, 35]. To reduce computational complexity and study the impact of affordance-aware planning on tool-use behaviour, affordances are treated as known in this work.

Similarly, **ASM.4** assumes that grasp poses are known via TFs. In practice, the grasp pose of an object is set equal to its detected object pose. While grasp pose

estimation is a well-established research area [44, 16, 14], reliable grasp execution is required to meaningfully evaluate TAMP performance. Grasp poses are therefore provided to ensure robustness during experimental evaluation, at the cost of not accounting for variation in object geometry that a dedicated grasp pose estimator would handle.

ASM.5 considers the location and pose of the tool tips w.r.t. the tool's grasping point to also be fixed and known. As the aim of this research is to systematically evaluate the TAMP system on effective workspace rather than affordance detection.

ASM.6 restricts the motion planner's collision scene to the robot model itself; the table and the box being manipulated are not added as collision objects. This keeps the collision check inexpensive and avoids requiring an accurate, continuously updated scene reconstruction, but means the planner cannot detect or avoid contact between a held tool and the table during the `drag_closer` action.

5.2 Evaluation Metrics

The performance of the proposed affordance-aware TAMP system is evaluated using the following metrics:

- Task Success Rate (%)
- Yoshikawa's Manipulability Index $w(\mathbf{q})$ (–)
  - This is a unitless metric defined by the kinematic model of the manipulator. It is not suitable for comparing manipulability across different robot platforms [10].

Task Success Rate is defined as the ratio of trials in which the robot successfully executes the complete task plan from start to finish, satisfying all task objectives. A trial is considered successful only if all required subtasks are completed without failure.

Yoshikawa's Manipulability Index as described in section 2.3, is proportional to the volume of the ellipsoid spanned by the end-effector velocity at a given configuration. It is used to determine proximity to kinematic singularity, with $w(\mathbf{q}) = 0$ indicating a singular configuration.

Because $w(\mathbf{q})$ is logged at every waypoint of a trial rather than once, a single trial-level value is needed for the statistical analysis in section 5.5. The trial mean $\bar{w}(q)$ is used for this purpose, rather than the trial minimum. The minimum is sensitive to single-waypoint outliers, for example a brief, atypical configuration produced by the motion planner, and does not necessarily reflect the manipulability of the trial as a whole. The mean is more robust to such outliers and better represents the overall manipulability the arm sustained throughout the trial.

5.3 Experimental Design

The tools selected for this experiments are chosen to match previous work by Nguyen and Choe [32] who used T-, I- and L-shapes to drag an out-of-reach object towards itself. Three conditions are evaluated to characterise the effect of tool use on workspace coverage:

1. **No-Tool**: The robot performs a direct pick-and-place task without any tool. This condition serves as the baseline for direct kinematic reach.
2. **Hammer**: The robot first grasps a hammer-shaped tool and uses it to execute the `drag_closer` action before performing the pick-and-place step.
3. **T-Shape**: As above, using a T-shaped tool. This condition tests whether tool geometry affects performance given a shared `drag_closer` PDDL abstraction.

All three conditions use the same PDDL domain. The No-Tool condition executes a PDDL plan containing three actions (`open_gripper`, `move_and_pick`, `move_and_place`); the Hammer and T-Shape conditions execute a PDDL plan with four actions that also contains a `drag_closer` action. The same environment, evaluation metrics, and object placement distribution are used across all conditions.

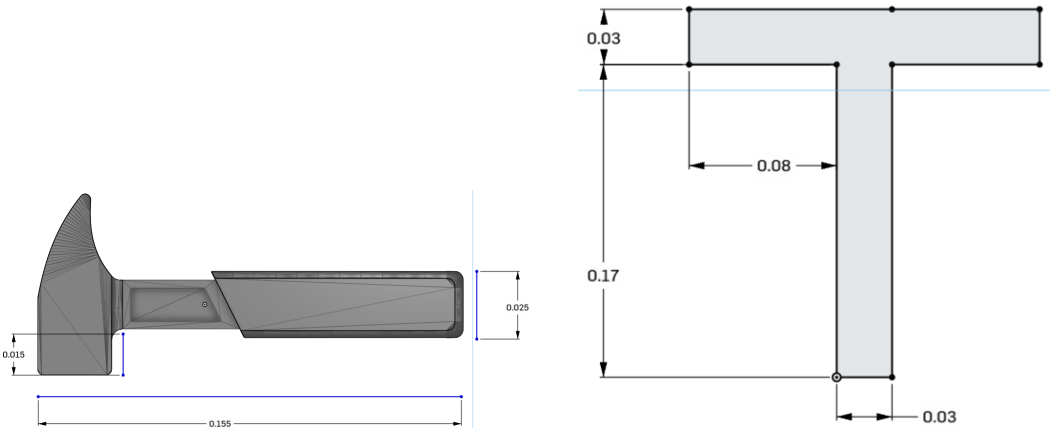
5.3.1 Tools and dimensions

In this work we use a T-shape and hammer (L-shape) to drag out-of-reach objects closer towards the robot for pick-and-place tasks. The dimensions of these tools are shows in fig. 5.1. **ASM.5** assumes that the pose of the tool tip is known with respect to the tool's grasping point. For the hammer, the tool tip adds a total extension of 11cm. For the T-shape, the total extension is 12.7cm. These values are chosen to ensure the tools stay balanced in the robot hand and do not move/slip out of the gripper during motion.

5.4 Configuration Space and Trial Design

Experimental configurations are designed to cover the robot's workspace across three spatial parameters: radial distance from the robot base, azimuth angle, and object orientation. The design is partly inspired by Nguyen and Choe [32], who vary object position and tool shape in a reaching and dragging task. This work follows a similar approach, setting radial distance and tool condition as the primary variables of interest, with object orientation and azimuth angle as secondary factors.

Radial distance is sampled uniformly from $r \in [0.35, 0.90]\,\mathrm{m}$. This range includes both the comfortable reachable zone, as well as regions beyond its reach and regions too close to the Franka Emika Panda [15]. Therefore, the transition zone in which tool use shows benefit is included in this range. The radii are sampled continuously,



(a) Hammer dimensions in metres (b) T-shape dimensions in metres

Figure 5.1: Dimensions of each tool used in the experiment

and thus radius is treated as a continuous variable in the statistical model described in section 5.5.

Azimuth angle is sampled uniformly from $\theta \in [0, \pi/3]$ rad. This range covers the portion of the workspace directly in front of the robot and everything to the side of it. Like radius, azimuth is treated as a continuous covariate in the model. As the Franka Emika Panda's joints can rotate in both directions, it is assumed that the robot performs similarly on both halves of the table.

Similarly, Object orientation refers to the rotation of the box object around its own axis, where 0° represents the same orientation as the world frame. Object orientation is sampled uniformly from $\psi \in [-\pi, \pi]$ and is also set as a continuous variable.

5.5 Statistical Analysis

To evaluate the relationship between manipulability, tool condition, object radius, and task success, a binary logistic regression model is employed. Task success is treated as Bernoulli-distributed, and the log-odds of success are modelled as:

$$\begin{aligned} \ln \frac{p}{1-p} = \beta_0 &+ \beta_1 \text{Hammer} + \beta_2 \text{T-Shape} + \beta_3 r + \beta_4 \theta + \beta_5 \psi \\ &+ \beta_6 \bar{w}(q) + \beta_7 (r \times \text{Hammer}) + \beta_8 (r \times \text{T-Shape}) \\ &+ \beta_9 (\bar{w}(q) \times \text{Hammer}) + \beta_{10} (\bar{w}(q) \times \text{T-Shape}), \end{aligned} \tag{5.1}$$

where p is the probability of task success; *Hammer* and *T-Shape* are binary values with No-Tool as the reference. Meaning that if Hammer $= 0$ and T-Shape $= 0$, the condition is *No-Tool*.
r Is the object's radial distance from the robot base; θ is the azimuth angle; ψ is

the object orientation in radians; and $\bar{w}(q)$ is the mean manipulability index recorded during the trial (see section 2.3).

The main effects β_1 and β_2 capture the overall log-odds change due to each tool condition relative to No-Tool (at the intercept, $r = 0$). β_3 captures how task success is influenced by radial distance for the No-Tool baseline.

The coefficients β_4 and β_5 represent the effect of azimuth (θ) and object orientation (ψ) on task success. While β_6 represents the effect of mean manipulability $\bar{w}(q)$ on task success.

The following terms are interaction terms and are formed by predictors, following standard practice in regression modelling [1]. In logistic regression, an *interaction* can be described as such: a model with outcome variable Y has and one predictor X. Predictor X may depend on another independent variable Z. By multiplying these variable ($Z \times X$) and adding them to the logistic regression model reveals how Z *moderates* X.

The interaction terms β_7 and β_8 are of interest and will test *"SQ.3: How does tool use affect the relationship between radial distance and success rate?."* They describe how the effect of radius r on task success is moderated by tool conditions (Hammer vs. T-shape vs No-Tool.)

Here the expectation is that the interaction between tool conditions and radius will have a positive effect on succes rate. Similarly, the terms β_9 and β_{10} test whether the effect of manipulability on task success is moderated differently across tool conditions.

5.6 Sample size determination

The number of trials is determined by the events-per-variable (EPV) guideline for logistic regression [36], which recommends a minimum of 10 outcome events per predictor term. An event is defined as the minority class of the outcomes (Failure/Success). In this thesis, the minority class corresponds to task failure.
To remain conservative, an EPV of 20 is adopted. The model in eq. (5.1) contains 10 predictor terms, requiring a minimum of $10 \times 20 = 200$ failure events across the full dataset.

In preliminary tests conducted on the real robot, a failure rate of 15.1% was observed. These preliminary tests were done at much sparser radial distances than $r \in [0.35, 0.9]$, namely at 655 mm, 755mm and 855mm from the robot base. The expectation is that the simulated experiments will have a higher failure rate at distances beyond the maximum reach 855 mm as well as at very close distances (<500mm). Therefore, the preliminary failure rate of 15.1% is a conservative estimate for determining the minimum sample size. The minimum total number of trials is:

$$n_{\text{total}} \geq \frac{200}{0.151} \approx 1325 \text{ trials.} \tag{5.2}$$

A total of 1326 trials are conducted: 442 per condition (No-Tool, Hammer, T-Shape). Important to note is that the EPV guideline is a rule of thumb rather than a formally derived bound [36], and the estimates above assume approximate independence between trials, which is reasonable given that each trial uses an independently sampled object pose.

Results

As described in section 5.6, results are drawn from 1326 simulation trials distributed across a polar grid of object positions ($r \in [0.35, 0.90]\,\text{m}$, $\theta \in [0, \pi/3]\,\text{rad}$), with 442 trials per condition (No-Tool, Hammer, T-Shape).

6.1 Task Success Rate

Success rates differ significatly across conditions. The success rates for the conditions are: No-Tool 83.0%, Hammer 57.5%, T-Shape 48.0%.

We can use Pearson's χ^2-test of independence eq. (6.1) to determine whether there is a statistically significant relation between condition and success rate.

$$\chi^2 = \sum_{i=1}^{2} \frac{(O_i - E_i)^2}{E_i} \tag{6.1}$$

where

- χ^2 = Pearson's test statistic
- O_i = The number of observations of type i
- N = The total number of observations
- E_i = expected count of type i, where i is the column of the contingency table in table 6.1.

Calculating Pearson's chi-squared with the contingency table in table 6.1 gives $\chi^2(2) = 124.5$, $p < 0.001$, indicating that there is a significant difference in task success rate across conditions, also shown in fig. 6.1.

Table 6.1: Contingency table for tool condition and task success. Calculating Pearson's chi-squared with this gives $\chi^2(2) = 124.5$, $p < 0.001$.

	success=0	**success=1**
No-Tool	75	367
Hammer	188	254
T-shape	230	212

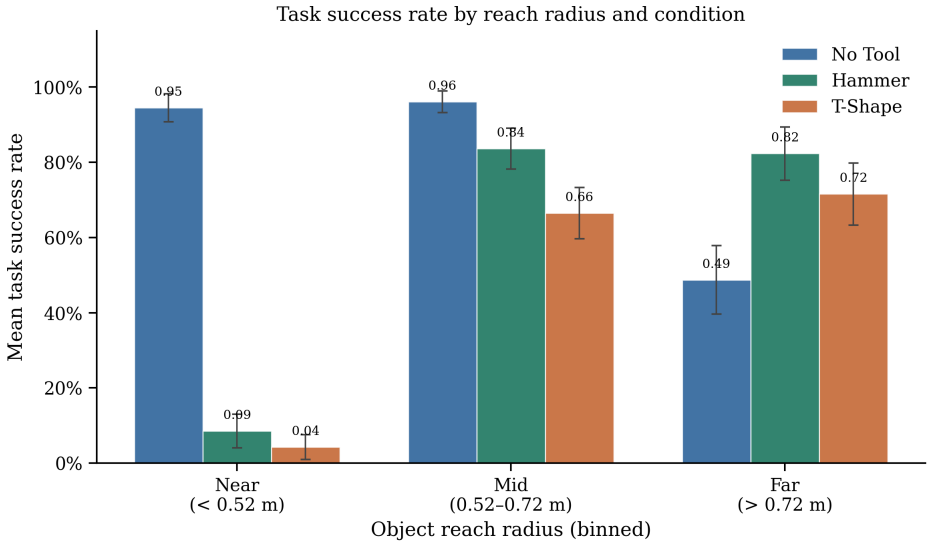


Figure 6.1: Task success by condition. No-Tool achieves the highest success at near radii. Tool conditions perform worse at close ranges and outperform the No-Tool conditions at far radii. ($\chi^2(2) = 124.5$, $p < 0.001$).

Figure 6.1 shows thath the No-Tool condition performs extremely well at close ranges, where the Hammer and T-Shape have a success rate of 9% and 4% respectively. It must be noted that these tool conditions are additionally disadvantaged by the extra `drag_closer` action at radii close to the robot base. This is also shown in fig. 6.2, where the No Tool condition performs well at close radii and falls off at further radii, while both tool conditions Hammer and T-shape have an inverse relation with task success. Both tools have a low task success rate at close radii, and perform better the further out the object is placed.

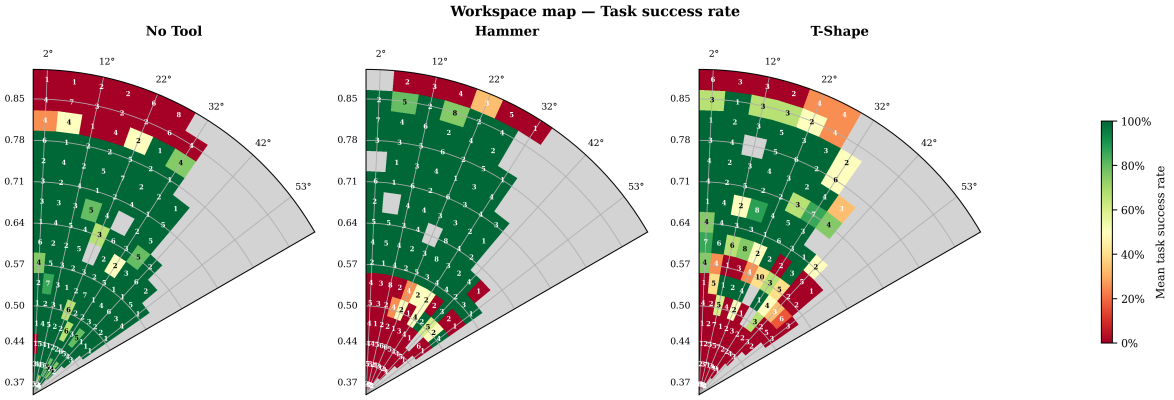


Figure 6.2: Heat map depicting task success at different radii and azimuths across the workspace. The No Tool condition performs better at close radii and falls off at further ranges. While both tool conditions have an inverse relation with task success.

Figure 6.3 shows the sliding-window success rate as a function of object radius. At $r = 0.35$–$0.55\,\text{m}$, No-Tool succeeds in approximately 95% of trials while both tool con-

ditions are near 0%. Both No-Tool and Hammer conditions converge at approximately 95% near $r = 0.60\,\text{m}$, while the T-shape hangs around 60%. Beyond $r = 0.65\,\text{m}$, tool conditions hold at 90–100% success while No-Tool collapses to approximately 29% at $r = 0.82\,\text{m}$ and 0% at $r = 0.87\,\text{m}$.

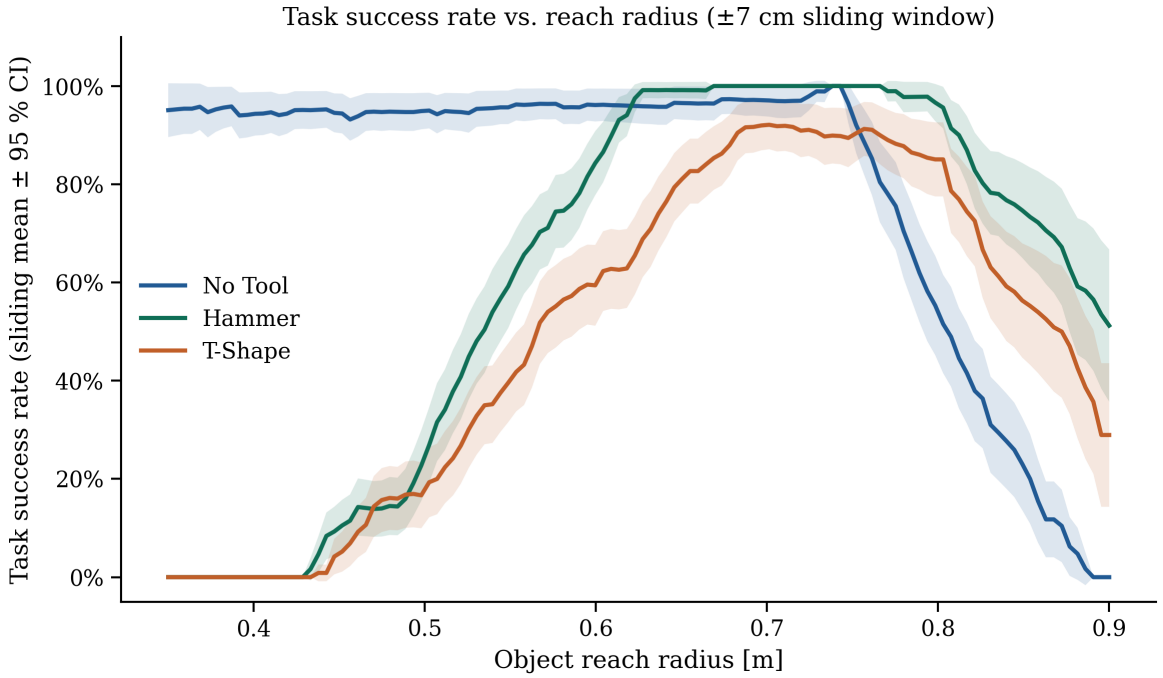


Figure 6.3: Sliding-window task success rate as a function of object radius r for all three conditions. The transition zone at $r \approx 0.7$–$0.8\,\text{m}$ marks the crossover from No-Tool dominance to tool-condition dominance.

6.1.1 Task success vs. Object Orientation

Task success rate showed largely invariant to object orientations. Figure 6.4 plots the success rate across different object orientations for each condition.

6.2 Logistic Regression Analysis

A logistic regression model was fit to binary task success with predictors: condition (relative to No-Tool), object radius r, azimuth θ, orientation ψ, mean manipulability $\bar{w}$, and interaction terms radius×condition and $\bar{w}$×condition.

To determine the goodness-of-fit we use McFadden pseudo-R^2_L:

$$R^2_L = 1 - \frac{ln(L_m)}{ln(L_0)} \tag{6.2}$$

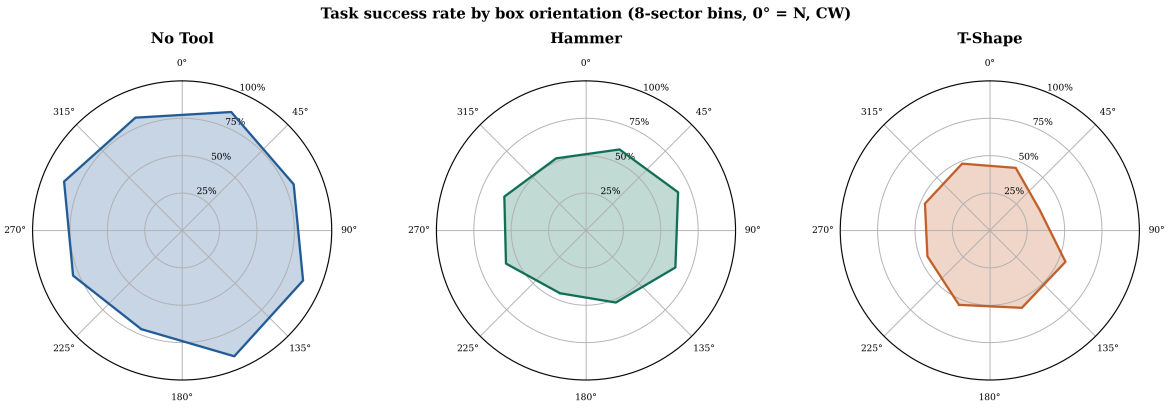


Figure 6.4: Success rate as a function of object orientation per conditions, showing no significant difference in success rate across orientations.

Table 6.2: Logistic regression coefficients for binary task success ($n = 1326$, McFadden $R^2 = 0.483$). Condition is relative to No-Tool. Intercept corresponds to No-Tool at $r = 0$, $\theta = 0$, $\psi = 0$, $\bar{w} = 0$.

Predictor	β	**SE**	z	p
Intercept	+5.79	1.25	+4.65	< 0.001
Hammer	−20.33	2.01	−10.12	< 0.001
T-Shape	−10.81	1.39	−7.79	< 0.001
r (No-Tool)	−8.92	1.40	−6.37	< 0.001
$r\times$ Hammer	+14.72	2.07	+7.11	< 0.001
$r\times$ T-Shape	+18.08	1.92	+9.44	< 0.001
θ (azimuth)	−0.19	0.37	−0.50	0.618
ψ (orientation)	+0.15	0.05	+3.18	0.001
$\bar{w}$	+38.72	8.33	+4.65	< 0.001
$\bar{w}\times$ Hammer	+247.08	32.90	+7.51	< 0.001
$\bar{w}\times$ T-Shape	−48.62	18.46	−2.63	0.008

Where L_M is the log-likelihood of the fitted model, and L_0 is the log-likelihood of the null model (where it only contains an intercept β_0.) The pseudo-$R^2 = 0.483$, Likelihood ratio test (LRT) gives a p-value of $p < 10^{-175}$, and the accuracy is 88%: 82 false negatives, 73 false positives. Table 6.2 reports all coefficients.

6.2.1 Task success vs. Tool

The central finding is in the interaction terms between tools and radius. The tool main effects (Hammer $\beta = -20.33$, T-Shape $\beta = -10.81$, both $p < 0.001$) represent the log-odds penalty versus No-Tool at $r = 0$ and $\bar{w} = 0$. These terms are meaningful only in combination with the radius and manipulability interactions, because $r = 0$ lies outside the sampled workspace.

The radius×condition interactions reverse the sign of the radius slope completely. The net radius slope per condition is:

- No-Tool: $\beta_r = -8.92$ (farther → lower success)
- Hammer: $\beta_r = -8.92 + 14.72 = +5.80$ (farther → higher success)
- T-Shape: $\beta_r = -8.92 + 18.08 = +9.16$ (farther → higher success)

Tool conditions require distance to outperform direct reach. The crossover radius now depends on $\bar{w}$. To obtain it we set compare the log-odds of the No-Tool condition with that of the Hammer and T-shape condition. By setting these equal to each other and solving for r we can determine the crossover radius for each condition.

The log-odds of task success under each condition are:

$$\ln\left(\frac{p}{1-p}\right)_{\text{NoTool}} = \beta_0 - 8.92\,r + 38.72\,\bar{w} \tag{6.3}$$

$$\ln\left(\frac{p}{1-p}\right)_{\text{Hammer}} = \beta_0 - 8.92\,r - 20.33 + 14.72\,r + 38.72\,\bar{w} + 247.08\,\bar{w} \tag{6.4}$$

$$\ln\left(\frac{p}{1-p}\right)_{\text{T-shape}} = \beta_0 - 8.92\,r - 10.81 + 18.08\,r + 38.72\,\bar{w} - 48.62\,\bar{w} \tag{6.5}$$

where β_0 is the intercept and all covariates cancel out. Setting $\ln\left(\frac{p}{1-p}\right)_{\text{NoTool}} = \ln\left(\frac{p}{1-p}\right)_{\text{Hammer}}$, the terms β_0, $-8.92\,r$, and $38.72\,\bar{w}$ cancel, leaving:

$$0 = -20.33 + 14.72\,r + 247.08\,\bar{w} \tag{6.6}$$

Setting $\ln\left(\frac{p}{1-p}\right)_{\text{NoTool}} = \ln\left(\frac{p}{1-p}\right)_{\text{T-shape}}$, the same terms cancel:

$$0 = -10.81 + 18.08\,r - 48.62\,\bar{w} \tag{6.7}$$

Solving for r in each case gives the crossover radius as a function of $\bar{w}$:

$$\begin{aligned} r^*(\bar{w}) &= \frac{20.33 - 247.08\,\bar{w}}{14.72} \quad \text{(Hammer)}, \\ r^*(\bar{w}) &= \frac{10.81 + 48.62\,\bar{w}}{18.08} \quad \text{(T-Shape)}. \end{aligned} \tag{6.8}$$

Evaluated at the sample mean $\bar{w} \approx 0.051$, this gives $r^* \approx 0.53\,\text{m}$ (Hammer) and $r^* \approx 0.74\,\text{m}$ (T-Shape).

Figure 6.5 shows the predicted probability curves; the regression intersects at $r \approx 0.53\,\text{m}$ (Hammer) and $r \approx 0.74\,\text{m}$ (T-Shape). This indicates a transition zone at $r \approx 0.55$–$0.65\,\text{m}$, consistent with the physical reach extensions provided by each tool: 11 cm for the Hammer and 12.7 cm for the T-Shape (see section 5.3.1).

Hammer and T-Shape provide statistically similar reach extension. The difference in radius interaction is $\Delta\beta = 3.36$ (Hammer +14.72, T-Shape +18.08), with heavily overlapping confidence intervals (Hammer CI: $[10.66, 18.78]$; T-Shape CI: $[14.32, 21.83]$).

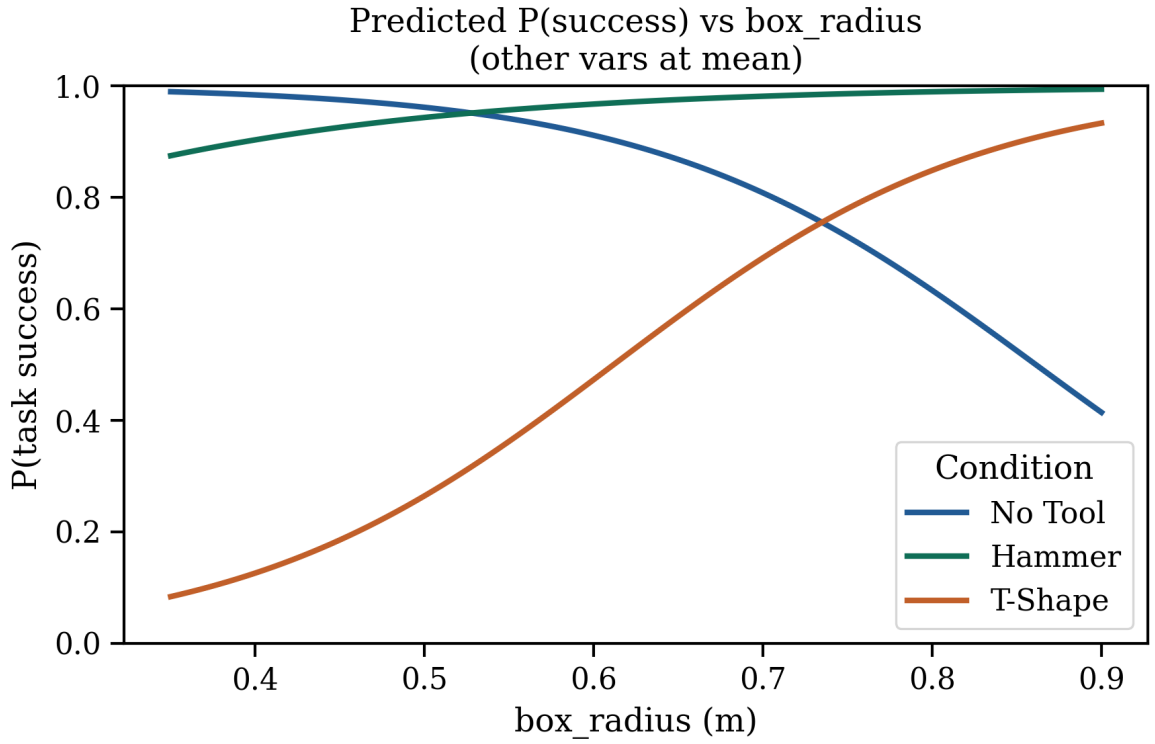


Figure 6.5: Logistic regression predicted probability of task success as a function of radius, with other covariates held at their means. Tool-condition curves have positive slopes. Hammer crosses No-Tool at $r* = 0.53\,\text{m}$ and T-Shape crosses No-Tool at $r* = 0.74\,\text{m}$ (Equation 6.8).

When it comes to effect of manipulability on the tools, the hammer and the T-shape differ. The $\bar{w}$ main effect is significant ($\beta = +38.72$, $p < 0.001$), indicating that higher mean manipulability increases task success across the board. The $\bar{w}\times$Hammer interaction is large and significant ($\beta = +247.08$, $p < 0.001$), meaning Hammer success is strongly amplified by manipulability (net $\bar{w}$ effect: $+285.8$). The $\bar{w}\times$T-Shape interaction on the other hand gives a negative coefficient of ($\beta = -48.62$, $p = 0.008$), which means that manipulability (net $\bar{w}$ effect: -9.9) has a negative effect on task success when the T-shape is used. The PDDL planner uses the same `drag_closer` action for both tools, however the regression reveals that tool shape significantly affects the probability of success. This is unexpected, since both tools use the same planning logic and should be similarly affected by arm manipulability. A likely explanation is geometric: the T-shape has a wider lateral profile than the hammer, giving it more surfaces that can contact the table or surrounding clutter during the drag. Simulated collisions are non-deformable, so a tool contacting multiple surfaces at once can become wedged, halting the drag regardless of the arm's posture. In a physical system, slight compliance in surfaces and contacts would let the tool slip free of such collisions. T-shape failures may therefore stem from this simulation-specific wedging rather than from arm kinematics, explaining why its success is largely insensitive to $\bar{w}$. This is a likely sim-to-real gap that warrants validation on physical hardware. Hammer success

is strongly dependent on arm manipulability during the drag phase ($\beta_{\bar{w}} = +285.8$), whereas T-Shape success is insensitive to robot configuration (net $\beta_{\bar{w}} : -9.9$). The per-phase analysis in Section 6.2.2 provides a more detailed view of this relationship.

6.2.2 Manipulability Analysis

Before running the Kruskal-Wallis and Mann-Whitney tests on $\bar{w}$, we check whether $\bar{w}$ is normally distributed within each group. For this the Shapiro-Wilk test is recommended [23]. Grouped by condition, all three groups reject the null-hypothesis of being normally distributed: No-Tool $W = 0.968$, $p = 3.3 \times 10^{-8}$; Hammer $W = 0.929$, $p = 1.2 \times 10^{-13}$; T-Shape $W = 0.946$, $p = 1.1 \times 10^{-11}$ (each $n = 442$). Grouped by outcome, both groups also reject null-hypothesis: Fail $W = 0.954$, $p = 3.0 \times 10^{-11}$ ($n = 493$); Success $W = 0.982$, $p = 1.3 \times 10^{-8}$ ($n = 833$).

It must be noted that at a sample size this large, the Shapiro-Wilk test is hypersensitive: even small deviations from normality become statistically significant [23], so a low p-value alone does not automatically mean the distribution is non-normal. Ghasemi and Zahediasl recommend to complement this with a visual check. Q-Q plots with 95% bootstrap confidence bands around the normal-fit line were produced for each group, shown in figs. 6.6a and 6.6b. In every group, the observed quantiles fall outside the band, particularly in the tails, indicating that the data is not normally distributed.

Interestingly, Rochon, Gondan, and Kieser [40] comment on the necessity of formality testing and conclude that preliminary testing for normality in incorrect an should be avoided. Establishing whether a the data is normally distributed should be supported by earlier trials or pilot studies. As for large samples the t-test has been shown robust. They mention that if the validity of a t-test is unsure, nonparametric tests are the best choice.

$\bar{w}$ is a meaningful positive predictor of task success. The Mann-Whitney test between failed and successful trials gives $U = 86930$, $p < 0.001$, with a clear difference in medians (fail: $\bar{w} = 0.041$; success: $\bar{w} = 0.057$), shown in fig. 6.9a.

Global mean manipulability $\bar{w}$ differs by condition (Kruskal-Wallis $H = 97.0$, $p < 0.001$). Hammer has the lowest median ($\bar{w} = 0.042$), significantly below both No-Tool ($\bar{w} = 0.054$; Mann-Whitney $p < 0.001$) and T-Shape ($\bar{w} = 0.052$; $p < 0.001$). No-Tool and T-Shape are statistically indistinguishable ($p = 0.656$), shown in fig. 6.9b. The Hammer condition therefore requires more extreme robot configurations than the other two conditions; T-Shape manipulability is comparable to direct reach.

If we evaluate the Success and Failure cases of each phase (Pick, Place, Drag) separately for manipulability we get fig. 6.8. This reveals that the Drag action becomes a bottleneck causing failures for the tool conditions. Manipulability during the Drag phase is a strong predictor of success. For the Pick-phase there is no large difference in manipulability between Failure and Success for the tool conditions. While the No Tool condition during Pick shows very low manipulability during Failure, suggesting that the robot configuration reaches singularity. The Place phase on the other hand

Table 6.3: Point-biserial correlation between $\bar{w}$ and task success rate, by phase and condition.

Phase	Condition	r_{pb}	p
Pick	No-Tool	0.472	< 0.001
Pick	Hammer	0.060	0.332
Pick	T-Shape	0.145	0.022
Drag	Hammer	0.727	< 0.001
Drag	T-Shape	0.514	< 0.001

contains no failure cases. This is because all failures during the trial occur *before* the Place action. Once the robot manages to succesfully Drag and Pick, the Place action always succeeds.

Additionally, to obtain the correlation between mean manipulability $\bar{w}$ and success rate we calculate the point-biserial correlation, used to measure the correlation between a continuous variable ($\bar{w}$) and a binary variable (success rate). Table 6.3 reports the coefficients per phase and condition, and fig. 6.7 shows the corresponding figure. The Place phase is not added in this analysis, as all failures occur *before* the Place phase. Meaning there are no failure events during placing.

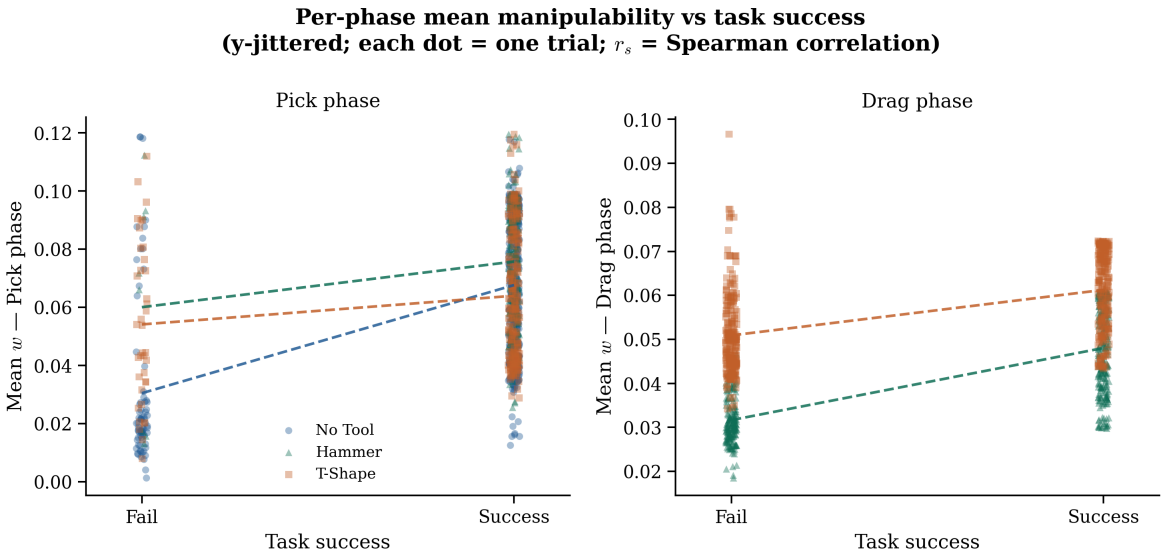


Figure 6.7: Point-biserial correlation between $\bar{w}$ and task success rate, by phase and condition (see table 6.3).

Table 6.3 and fig. 6.7 show that the correlation between $\bar{w}$ and success is phase-dependent rather than uniform across a trial. For No-Tool, the Pick phase already carries a strong correlation with success ($r_{pb} = 0.472$, $p < 0.001$): since Pick is the only phase in which the No-Tool arm moves toward the object, low manipulability there directly reflects the arm approaching its kinematic limit. For Hammer and T-Shape, the Pick phase correlation is weak and, for Hammer, not statistically significant ($r_{pb} = 0.060$, $p = 0.332$), since picking up the tool takes place close to the robot base

regardless of where the object is placed and is rarely the limiting step. The Drag phase instead carries the strongest correlations for both tool conditions (Hammer $r_{pb} = 0.727$; T-Shape $r_{pb} = 0.514$; both $p < 0.001$), showing that dragging the object once the tool is grasped is where manipulability determines success. Together these results localise the effect of manipulability to a different phase depending on the condition: Pick for No-Tool, Drag for the tool conditions.

6.3 Manipulability Across the Workspace

Figure 6.10 shows a heat map of mean manipulability $\bar{w}$ across the workspace, and fig. 6.11 shows a LOESS curve with bootstrap 95% confidence bands of $\bar{w}$ against object radius r for each condition. Both show a pattern that depends clearly on which condition is used.

For No-Tool, $\bar{w}$ rises through the near and middle radii, reaches a peak around $r \approx 0.65$–$0.70\,\text{m}$, and then drops sharply for $r > 0.8\,\text{m}$. This drop lines up with the arm getting close to singular configurations. For Hammer and T-Shape, $\bar{w}$ rises in a similar way at close radii, but then falls off more gently (Hammer) or stays roughly flat (T-Shape) at far radii. In both tool conditions, $\bar{w}$ stays noticeably higher than No-Tool for $r > 0.7\,\text{m}$, which is also the range where No-Tool can no longer complete the task.

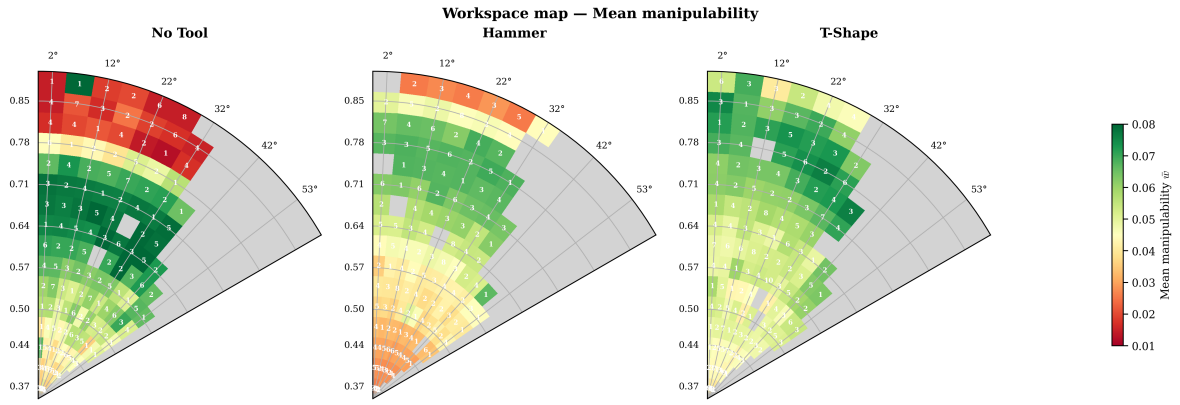
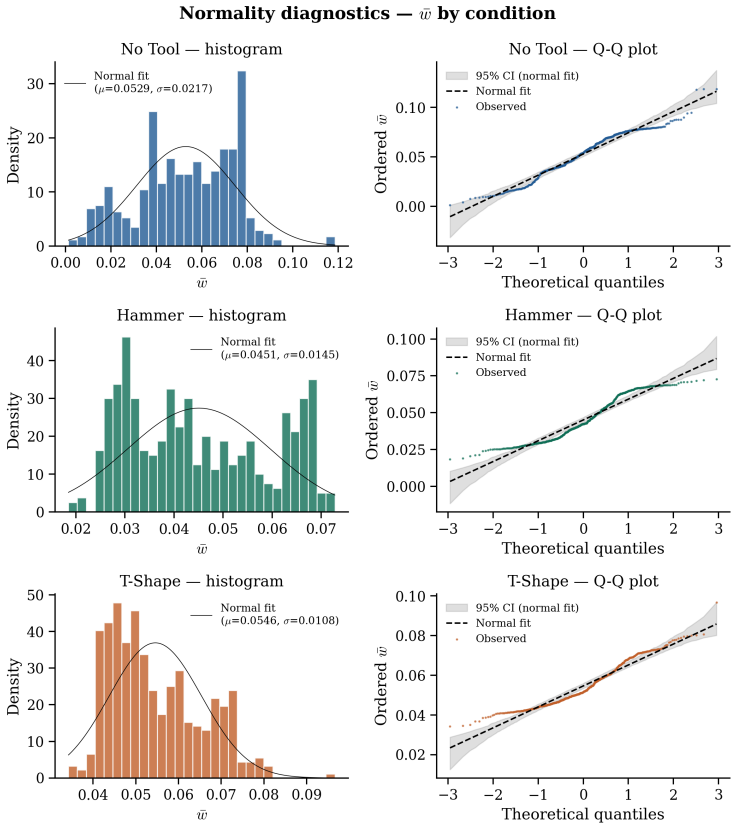


Figure 6.10: Heat map of mean manipulability $\bar{w}$ across the workspace (radius r and azimuth θ) for each condition.

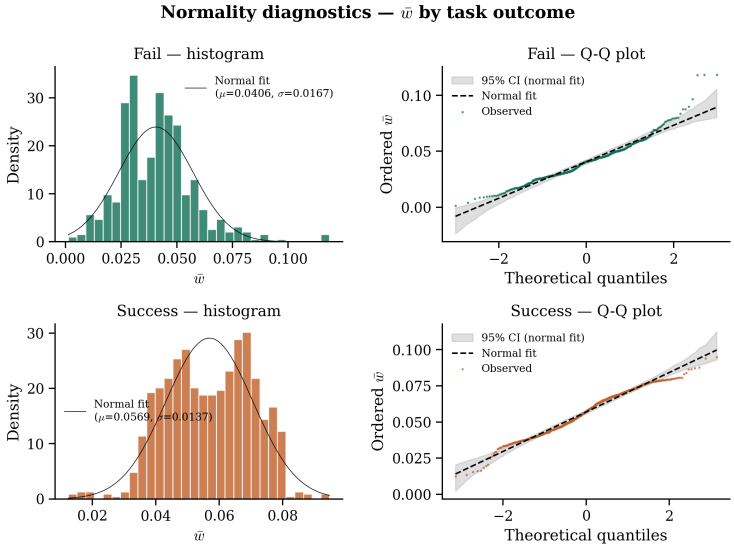
6.4 Correlation Matrix

Figure 6.12 shows the pairwise correlation between the main variables: object radius r, azimuth θ, orientation ψ, mean manipulability $\bar{w}$, and task success. Spearman's r_s is used instead of Pearson's r to calculate the correlation, since section 6.2.2 already shows that $\bar{w}$ is not normally distributed [2]. Spearman's r_s does not assume normality, as it is computed on the ranks of the data rather than on the raw values.

None of the predictors are strongly correlated with each other. The correlation between $\bar{w}$ and r is a moderate ($r_s = 0.405$), which makes sense, since reaching further requires more arm extension and therefore affects manipulability. Similarly, $\bar{w}$ is moderately correlated with task success ($r_s = 0.483$, interpretation of the correlation coefficients follow Dacey & Reidy as mentioned in [2]). All other predictor pairs have $|r_s| \leq 0.16$. This shows that these predictors are not correlated with eachother and multicollinearity is not a concern for the logistic regression in table 6.2.



(a) Q-Q plots of $\bar{w}$ for each condition, with 95% bootstrap confidence bands around the normal-fit line. Observed quantiles fall outside the band in the tails for all three conditions.



(b) Q-Q plots of $\bar{w}$ for failed and successful trials, with 95% bootstrap confidence bands. Both groups show deiations at the tail from the normal-fit.

Figure 6.6: Q-Q plots with 95% bootstrap confidence bands for $\bar{w}$, by condition and by task outcome. These support the Shapiro-Wilk results above and justify the use of non-parametric tests.

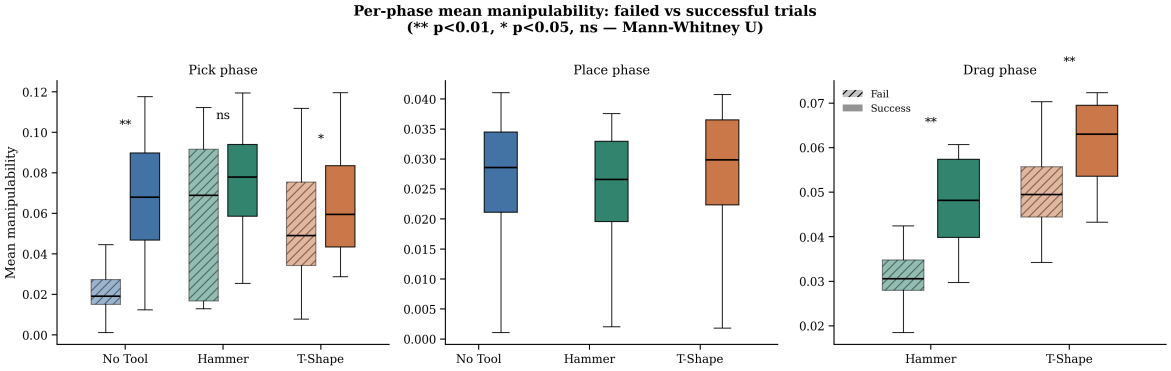
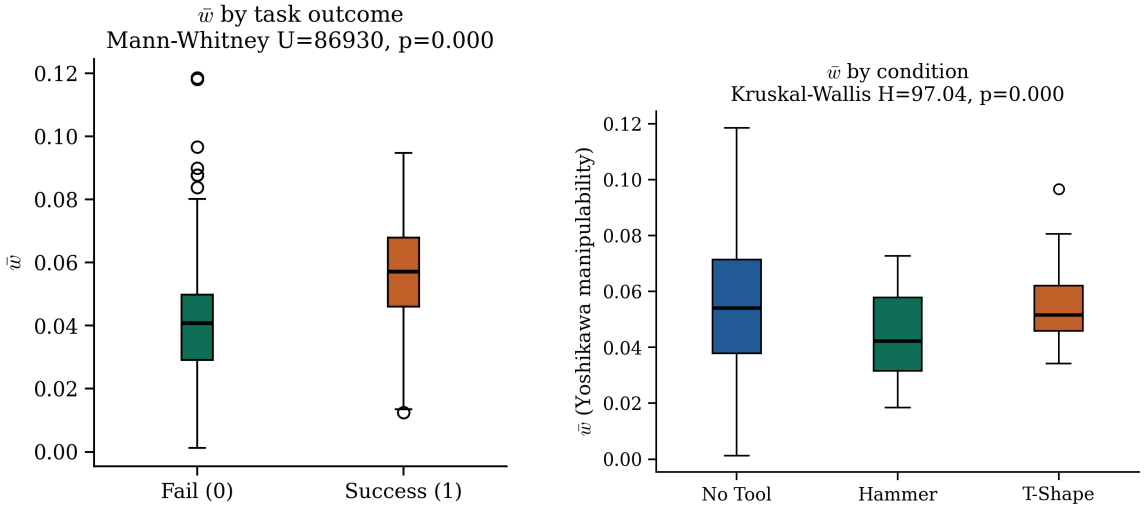


Figure 6.8: Mean manipulability $\bar{w}$ per action phase and condition with Success and Failure separate. Notable is that failure cases consistently have lower manipulability, and particularly for the Drag phase, there is a large difference in manipulability between Failures and Successes.



(a) Mean manipulability $\bar{w}$ by task outcome. Successful trials have higher $\bar{w}$ than failed trials (Mann-Whitney $U = 86930$, $p < 0.001$; median fail = 0.041, success = 0.057).

(b) Mean manipulability $\bar{w}$ by condition. Hammer is significantly lower than both No-Tool and T-Shape (Kruskal-Wallis $H = 97.0$, $p < 0.001$; Mann-Whitney $p < 0.001$ for both Hammer pairs). Mann-Whitney for No-Tool and T-Shape gives $p = 0.656$, meaning they are statistically indistinguishable.

Figure 6.9: Mean manipulability $\bar{w}$ by task outcome and condition

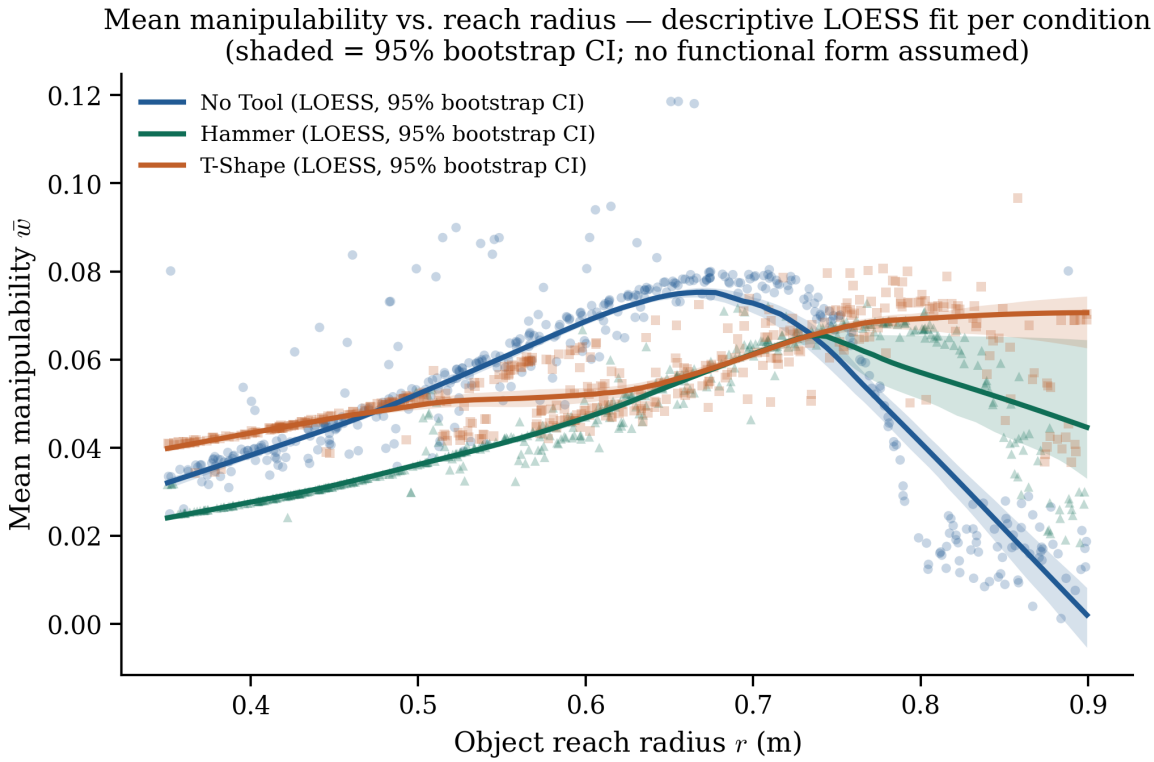


Figure 6.11: LOESS smooth of mean manipulability $\bar{w}$ against object radius r for each condition, with bootstrap 95% confidence bands. No-Tool peaks around $r \approx 0.65$–$0.70\,\mathrm{m}$ before dropping sharply, while Hammer and T-Shape decline more gently or stay flat at far radii.

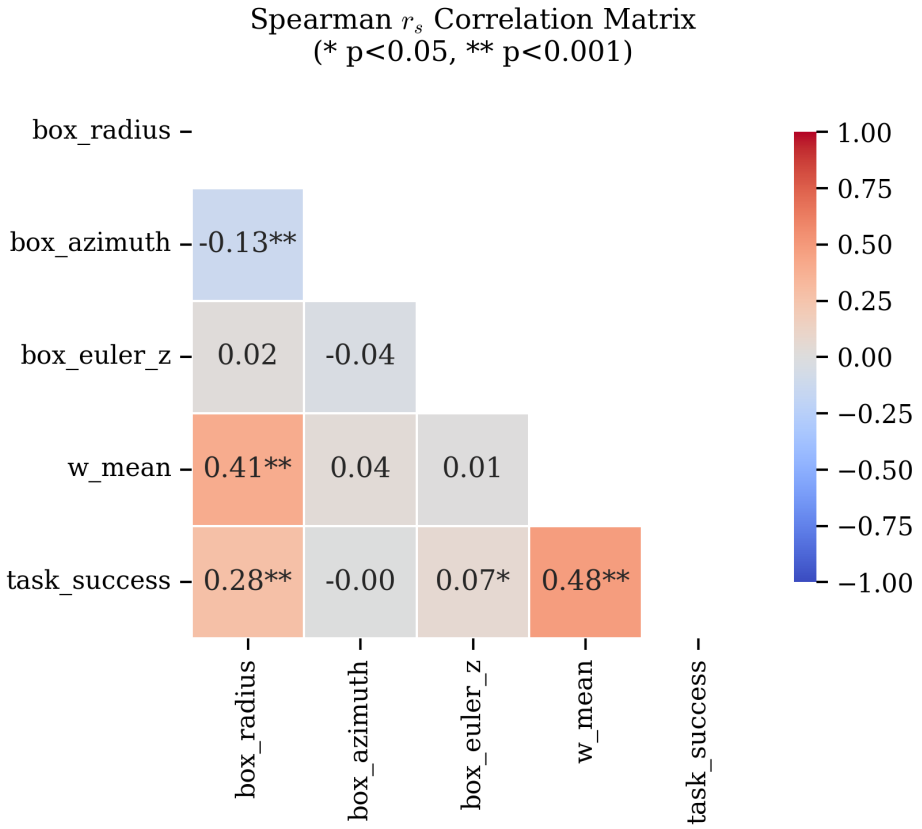


Figure 6.12: Spearman correlation matrix between object radius, azimuth, orientation, mean manipulability $\bar{w}$, and task success. All predictor variables are weakly to moderately correlated with each other ($|r_s| \leq 0.483$).

Discussion

This chapter interprets the results from chapter 6. We first discuss how much tool use extends the robot's effective workspace, and the trade-off this introduces at close range. We then compares the Hammer and T-Shape geometries, since both extend reach by a similar amount but differ in sensitivity to manipulability. Manipulability is then examined as a predictor of task success, overall and per action phase, followed by how manipulability varies across the workspace. The results are then compared with Nguyen and Choe [32], before closing with limitations and future work. Section 7.7 summarises the conclusions reached in this chapter.

7.1 Effective Workspace Extension

The results confirm the central hypothesis: tool use measurably extends the effective workspace of the robot beyond its direct kinematic reach. Under the No-Tool condition, task success collapsed to approximately 49% at $r > 0.72\,\text{m}$ and to 0% at $r = 0.87\,\text{m}$, consistent with the Franka Emika Panda operating near the boundary of its reachable workspace [15]. Hammer and T-Shape conditions maintained 72% and 82% success beyond $r = 0.72\,\text{m}$, demonstrating that the `drag_closer` action reliably retrieves objects that are unreachable with direct pick-and-place.

The logistic regression quantifies this trade-off. Tool conditions penalize task success at close ranges (Hammer $\beta = -20.33$, T-Shape $\beta = -10.81$, both $p < 0.001$), reflecting the cost of the additional Drag phase: at short distances the object is already within direct reach, so the tool's physical extension (11 cm for the Hammer, 12.7 cm for the T-Shape; section 5.3.1) provides no benefit while the extra drag step introduces additional failure modes. The Tool conditions however introduce a positive interactions interaction with further radii(Hammer +14.72, T-Shape +18.08, both $p < 0.001$), meaning that as radius increases, tool-use provides an increase in task success. Giving predicted crossovers at $r^* \approx 0.53\,\text{m}$ (Hammer) and $r^* \approx 0.74\,\text{m}$ (T-Shape) at the sample mean $\bar{w}$ (Equation 6.8) beyond which tool use becomes the dominant strategy. The empirical transition zone of approximately $r = 0.55$–$0.65\,\text{m}$, visible in Figure 6.3,

These findings support dynamic tool-use selection: rather than always using a tool, the system should invoke the `drag_closer` plan only when an IK feasibility check

indicates that direct pick-and-place is unlikely to succeed. Below the transition zone, the extra Drag phase introduces unnecessary complexity to a plan, above it, tool use is necessary.

7.2 Tool-Shape influence

Hammer and T-Shape conditions provide statistically similar reach extension. The radius-interaction coefficients (+14.72 vs. +18.08) differ by $\Delta\beta = 3.36$ with heavily overlapping confidence intervals (Hammer $[10.66, 18.78]$; T-Shape $[14.32, 21.83]$), confirming that both tools extend the workspace by a comparable amount. The TAMP planner encodes both geometries through the same `drag_closer` action, and the radius data confirm the reach extension.

Nonetheless, Hammer annd T-Shape differ in their sensitivity to manipulability. The $\bar{w}\times$Hammer interaction is large and significant ($\beta = +247.08$, $p < 0.001$), meaning Hammer success is strongly amplified by manipulability (net $\bar{w}$ effect: +285.8). The $\bar{w}\times$T-Shape interaction partially reverses this ($\beta = -48.62$, $p = 0.008$), leaving T-Shape success nearly insensitive to manipulability (net $\bar{w}$ effect: ≈ -10, near zero). Tool shape therefore affects the kinematic cost of execution even when it does not affect the reach benefit: the Hammer requires systematically higher arm dexterity to execute the drag phase than the T-Shape.

This contrasts with Nguyen and Choe [32], who find T > L > I in an end-to-end RL setting because the RL agent learns strategies suited to each tool's shape from scratch. The TAMP planner runs the same motion for every tool shape, so it stays predictable but cannot take advantage of shape-specific benefits.

The T-Shape's near-zero sensitivity to manipulability is unlikely to come from a real kinematic advantage, since both tools use the same drag motion. A simpler explanation is that this is due to a sim-to-real gap. the T-Shape is wider than the Hammer, so it has more surfaces that can be in collision with the table or nearby clutter during the drag. Since collisions in MuJoCo are non-deformable, a tool touching several surfaces at once (like the table and the robot gripper) can get stuck, stopping the drag no matter how good the arm's posture is. On real hardware, surfaces and contacts can have some deformation, so the tool could slip free of these collisions instead of getting stuck. This suggests that T-Shape failures may be mostly caused by these collision behaviour in simulation, rather than by the arm's posture. Notably, these collision failures do not happen in real life experiments.

7.3 Manipulability as a predictor of Task Success

Mean manipulability $\bar{w}$ is a moderately positive predictor of task success (following Dancey & Reidy's interpretation in [2]) across all conditions (Spearman's $r_s = +0.483$, $p < 0.001$ (fig. 6.12); Mann-Whitney $p < 0.001$, median $\bar{w}$: fail = 0.041, success = 0.057, fig. 6.9a). This is partly due to the tool conditions, they have lower $\bar{w}$ and

lower task success rate, both following from close object radii. Analyzing the different phases reveals the cause for the lower performance.

Per-phase analysis reveals the actual bottleneck, see fig. 6.8. Failures during the Drag-phase systematically have lower mean manipulability than successes. While for tool conditions during the Pick-phase this is much less the case. The failure mode for tool conditions is therefore concentrated in the drag trajectory, not in the tool-grasp step. This is due to the drag motion requiring the arm to sweep through a wider range of configurations, thus being more likely to reach singular configurations. The Pick action in contrast only needs to move towards the object.

These findings motivate incorporating $\bar{w}$ as a cost term in trajectory optimisation for the drag phase specifically, as explored by Ti et al. [42]. Penalising low-manipulability configurations during drag-path planning could reduce the proportion of trials that reach kinematic limits mid-trajectory. The scalar manipulability index does have known limitations. Yoshikawa's manipulability [47] does not include directional constraints or joint-limit proximity directly [26], and a directional index aligned with the drag axis may be more informative for this specific phase. A version of this directional manipulability is implemented by Ti et al. [42].

7.4 Manipulability across the Workspace

A LOESS regression method was chosen to smoothen the scatterplot data in fig. 6.11. Figure 6.11 shows a clear drop in manipulability for the No-Tool condition around $r \approx 0.7m$, while the Hammer and T-Shape manipulability remains high. This pattern most likely comes from *where* the arm spends its motion during a trial, rather than tools making far-away postures kinematically easier to reach. In the No-Tool condition, most of the trial is spent reaching toward the box at radius r, so $\bar{w}$ tracks closely how near the arm gets to full extension, which explains the sharp drop at large r. In the Hammer and T-Shape conditions, a large part of the trial, picking up the tool and lining it up, happens close to the robot regardless of where the box is placed. This ensures the robot stays further from singular configurations and keeps $\bar{w}$ higher and steadier across the sampled range. The claim should therefore be framed carefully: tool use changes *where* the arm spends its motion, and this, as a side effect, keeps manipulability higher and more stable at large radii.

7.5 Comparison with Related Work

At $r > 0.72\,\text{m}$, the Hammer and T-Shape conditions achieve 72% and 82% success respectively, compared to the best marginal rate of 62% reported by Nguyen and Choe [32] for their T-shape condition. When the task closely resembles the TAMP system's dragging task, symbolic planning with fixed motion primitives achieves higher success rate than the end-to-end RL baseline.

The most relevant contribution relative to that baseline is the characterisation of the kinematic transition zone. Nguyen and Choe place objects exclusively beyond direct reach, so the near-field regime where tool use is actively worse is never observed. The continuous polar sampling used here reveals a trade-off across a large range: from near-0% success with tools at $r = 0.35\,\text{m}$ to near-100% at $r = 0.80\,\text{m}$, with a crossover at $r^* \approx 0.74\,\text{m}$.

The key difference in methodology is that the RL agent can produce emergent strategies (wide sweeps, contact corrections, multi-step repositioning) that the TAMP system cannot. Therefore the interpretability of this TAMP system comes at the cost of flexibility.

7.6 Limitations and Future Work

Collision awareness. The current system does not take collisions into account. During dragging, the held tool can collide with the table in the planner's model even when the physical trajectory is valid, causing premature plan termination. Adding a point-cloud-derived collision scene would resolve this and can allow for collision avoidance. Which may become particularly useful at orientations where the tool must be held close to the table surface.

Grasp pose estimation. Grasp poses are currently set equal to the detected object pose, which is sufficient for the experimental evaluation but is not robust to variation in object geometry. A dedicated grasp pose estimation module (e.g. GraspNet [16]) would improve generality and reduce orientation-dependent grasp failures.

Dynamic tool selection. The current system selects the tool condition at the experiment level. Integrating an IK-feasibility pre-check into the PDDL problem generator would allow the planner to select between No-Tool and tool-use plans at runtime, avoiding the near-field penalty described in Section 6.

7.7 Summary

Tool use extends the robot's effective workspace beyond its direct kinematic reach, but only beyond a crossover radius ($r^* \approx 0.53\,\text{m}$ for Hammer, $r^* \approx 0.74\,\text{m}$ for T-Shape); below it, the added Drag phase makes tool use a liability rather than a benefit. Hammer and T-Shape extend reach by a statistically similar amount, but they differ in how sensitive their success is to arm manipulability: Hammer success is strongly amplified by manipulability, while T-Shape success is nearly insensitive to it, likely because of a sim-to-real collision-wedging artefact rather than a genuine kinematic difference. Manipulability is a moderate positive predictor of task success overall, and the per-phase analysis shows that the Drag phase, is the main bottleneck for both tool conditions. Relative to Nguyen and Choe [32], the TAMP system achieves higher success in the far-field regime but cannot produce the emergent, shape-specific strategies available to an RL agent. Taken together, these findings support dynamic,

IK-feasibility-based tool selection and motivate the collision-awareness and grasp-pose improvements outlined above as the most impactful next steps.

Conclusion

This work presented an affordance-aware TAMP pipeline that enables a Franka Emika Panda robot to use a tool to drag out-of-reach objects into its workspace, extending task execution beyond the robot's nominal kinematic reach limit. The system was evaluated across 1 326 simulation trials sampled from a continuous polar grid ($r \in [0.35, 0.90]$ m, $\theta \in [0, \pi/3]$ rad) under three conditions: No-Tool, Hammer, and T-Shape. Task success rate and Yoshikawa's mean manipulability index $\bar{w}$ were used as evaluation metrics.

The primary finding is that tool use enables reliable task completion beyond the robot's direct kinematic reach. The No-Tool condition achieved near-0% success beyond $r = 0.87$ m while Hammer and T-Shape stay around $50\% - 60\%$ success at $r = 0.87$ m. A logistic regression model (McFadden $R^2 = 0.483$, accuracy 88%) quantifies the trade-off: tool conditions penalizes nearby radii (Hammer $\beta = -20.33$, T-Shape $\beta = -10.81$, both $p < 0.001$) which is fully reversed by positive radius interactions (Hammer $+14.72$, T-Shape $+18.08$, both $p < 0.001$), yielding predicted crossovers at $r^* \approx 0.53$ m (Hammer) and $r^* \approx 0.74$ m (T-Shape) at the sample mean $\bar{w}$, beyond which tool use is the dominant strategy. This directly addresses the primary research question: for the evaluated task, affordance predicates in PDDL are sufficient to integrate `drag_closer` actions into a TAMP pipeline, and this integration measurably extends the robot's effective workspace.

Hammer and T-Shape conditions provide statistically similar reach extension (radius-interaction difference $\Delta\beta = 3.36$, confidence intervals overlapping: Hammer $[10.66, 18.78]$, T-Shape $[14.32, 21.83]$). The tools diverge in their sensitivity to arm posture quality: Hammer success is strongly amplified by mean manipulability ($\bar{w}\times$Hammer $\beta = +247.08$, $p < 0.001$), while T-Shape success is near-insensitive ($\bar{w}\times$T-Shape $\beta = -48.62$, $p = 0.008$, net effect ≈ 0). For both tool shapes, the TAMP planner uses a shared `drag_closer` action to reach out with a tool. However, the T-Shape's low sensitivity to manipulability is likely caused by a sim-to-real gap. Its wider profile increases the chance of becoming wedged against other collision objects, and get stuck due to non-deformable collisions. Importantly, this behavior does not happen in real-world experiments.

Failures in tool conditions arise during the drag trajectory, not during the tool-grasp step. The manipulability analysis shows that mean manipulability $\bar{w}$ is a moderately positive predictor of task success ($r_s = +0.483$, $p < 0.001$). When analyzing each

phase (Pick, Place, Drag) separately we find that drag-phase manipulability is a strong predictor of success for tool conditions ($r_{pb} = +0.727, p < 0.001$ for Hammer, $r = +0.514, p < 0.001$ for T-Shape). Meaning there is a strong positive correlation between $\bar{w}$ and succes rate for the Hammer condition and a moderate positive correlation for the T-Shape.

Future work should address these limitations. First, incorporating a point-cloud-derived collision scene into the planning model would prevent premature plan termination caused by tool-table collisions, and would enable collision avoidance for orientations where the tool must pass close to the table surface. Second, a dedicated grasp pose estimation module (e.g. GraspNet [16]) would reduce orientation-dependent grasp failures, since grasp poses are currently set equal to the detected object pose. Third, dynamic tool-use selection based on IK feasibility would avoid the near-field penalty by invoking the drag plan only when direct pick-and-place is not kinematically feasible. Finally, incorporating $\bar{w}$ as a directional cost term aligned with the drag axis, as explored by Ti et al. [42], in drag-trajectory optimisation would reduce the frequency of kinematic-limit failures during motion.

Real-World Experiments

Prior to the simulation study in chapter 6, a smaller set of experiments were conducted on the physical Franka Emika Panda. These real-world experiments also contained additional sensor noise and used a coarser, fixed grid of positions rather than the continuous sampling used in simulation, so the two datasets are compared qualitatively rather than through a shared statistical model.

A.1 Experimental Setup

Two conditions were tested on the real robot: No-Tool and Hammer. The T-Shape was not included in the real-world trials. Grasp poses were provided via ArUco markers, and the robot ran the same `drag_closer` plan as in simulation.

The main difference from the simulation design is how positions are sampled. In simulation, radius and azimuth are drawn continuously from a polar grid. In the real-world trials, three fixed radii are used: $r \in \{655, 755, 855\}$ mm, and two azimuths: $\phi \in \{0°, 30°\}$. The radii were chosen to cover the three relevant distance regimes: 655 mm is comfortably within reach, 755 mm is a mid-range distance, and 855 mm is the maximum reach of the Franka Emika Panda specified by the manufacturer [15]. Four object orientations (0°, 90°, 180°, 270°) were tested, with ten trials per configuration. The discrete grid was chosen to keep the number of manually supervised trials manageable.

The same metrics are reported as in the simulation: task success rate and Yoshikawa's manipulability index. In the real-world trials, the mean manipulability $\bar{w}$ observed during each trial was logged, consistent with chapter 6.

A.2 Task Success Rate

Task success rates are shown in table A.1. Averaged across azimuths and orientations, the No-Tool condition reached 79.7% at $r = 655$ mm and 81.5% at $r = 755$ mm, but 0% at $r = 855$ mm: no trial succeeded at the maximum reach without a tool. The hammer condition completed trials at all three radii, including 51.8% at $r = 855$ mm.

Table A.1: Real-world task success rate (successful trials / total trials) for the No-Tool and Hammer conditions, by object orientation, radius r, and azimuth ϕ. A trial counts as successful only when all plan steps complete without failure. Each cell aggregates ten trials unless annotated.

		$r = 655\,\text{mm}$		$r = 755\,\text{mm}$		$r = 855\,\text{mm}$	
Condition	**Orient.**	$\phi = 0°$	$\phi = 30°$	$\phi = 0°$	$\phi = 30°$	$\phi = 0°$	$\phi = 30°$
No-Tool	0°	10/10	10/10	10/10	11/11†	0/10	0/10
	90°	6/10	6/10	7/10	7/10	0/10	0/10
	180°	6/10	8/10	6/10	5/10	0/10	0/10
	270°	8/10	9/9§	10/10	10/10	0/10	0/10
Hammer	0°	9/10	8/9§	10/10	10/10	7/10	3/10
	90°	5/13*	8/10	4/10	7/10	4/9§	1/10
	180°	7/10	3/10	9/10	8/9§	10/10	2/10
	270°	8/10	5/11†	9/10	9/10	9/10	5/10

§ 9 trials † 11 trials * 13 trials

The pattern with radius matches simulation. Figure A.1 shows the success rates plotted at the two azimuths $\theta \in \{0°, 30°\}$ and radii $r \in \{655, 755, 855\}$ mm. At $r = 755\,\text{mm}$ both conditions performed similarly (83.5% hammer vs. 81.5% No-Tool). At $r = 655\,\text{mm}$ the hammer was lower (63.8% vs. 79.7%), because the extra drag step adds an additional point of failure not present in the No-Tool plan. This is the same near-field penalty captured by the regression in chapter 6. The crossover from No-Tool being better to the hammer being better therefore falls somewhere between $r = 655\,\text{mm}$ and $r = 855\,\text{mm}$, which is consistent with the transition zone found in simulation.

Object orientation had a noticeable effect under the hammer condition. The 90° orientation produced the worst results on average (46.8%), compared to 79.7% at 0°, 66.1% at 180°, and 73.8% at 270°. At $r = 855\,\text{mm}$ the gap was most pronounced: 26.3% at 90° versus 70.0% at 270°. The cause is discussed in appendix A.4 and shown in fig. A.2. This differs from simulation, where success was largely insensitive to orientation (fig. 6.4), pointing to a real-world failure mode the simulator does not reproduce.

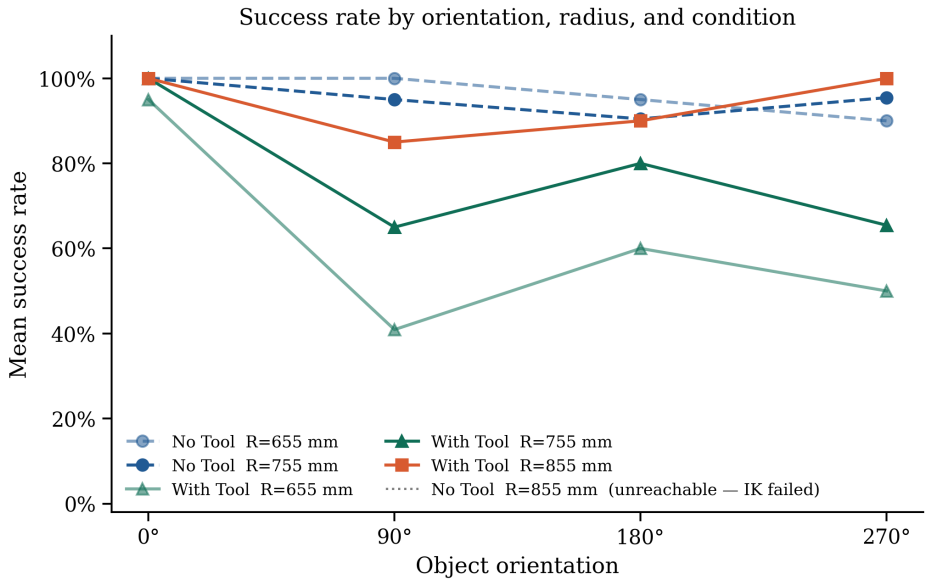


Figure A.2: Task success by orientation in real-world experiments. Different object orientations how significant differences in task success.

A.3 Manipulability Analysis

The real-world trials confirm that the drag phase is the hardest part for the arm kinematically. Visually this is shown in fig. A.4 Under the hammer condition, the drag phase consistently produced the lowest mean manipulability $\bar{w}$ across all radii (mean 0.023, compared to 0.051 for pick and 0.057 for place), meaning the arm passes through more awkward configurations during the drag than during any other part of the plan. At $r = 855\,\text{mm}$ the drag phase reached a mean $\bar{w}$ of 0.008, the lowest value recorded. The place phase stayed close to 0.057 across all radii, because the target bin is fixed and well within reach.

Under No-Tool, $\bar{w}$ during the pick phase dropped from 0.094 at $r = 655\,\text{mm}$ to 0.0 at $r = 855\,\text{mm}$. This near-zero value is consistent with the arm being at the edge of its reach and failing to complete any trial at that radius. Figure A.3a shows this at the different azimuths $\theta \in \{0°, 30°\}$.

A.4 Failure Modes

Two failure modes were observed in the real-world trials that do not appear in simulation. First, the collision model given to the planner did not include the table or the box, so during a drag the held tool could intersect the table in the planner's model without the planner detecting it. Second, lens distortion in the camera affected ArUco pose estimates, occasionally causing orientation-constraint failures during motion planning. At distances far away from the camera, this lens distortion in combination with

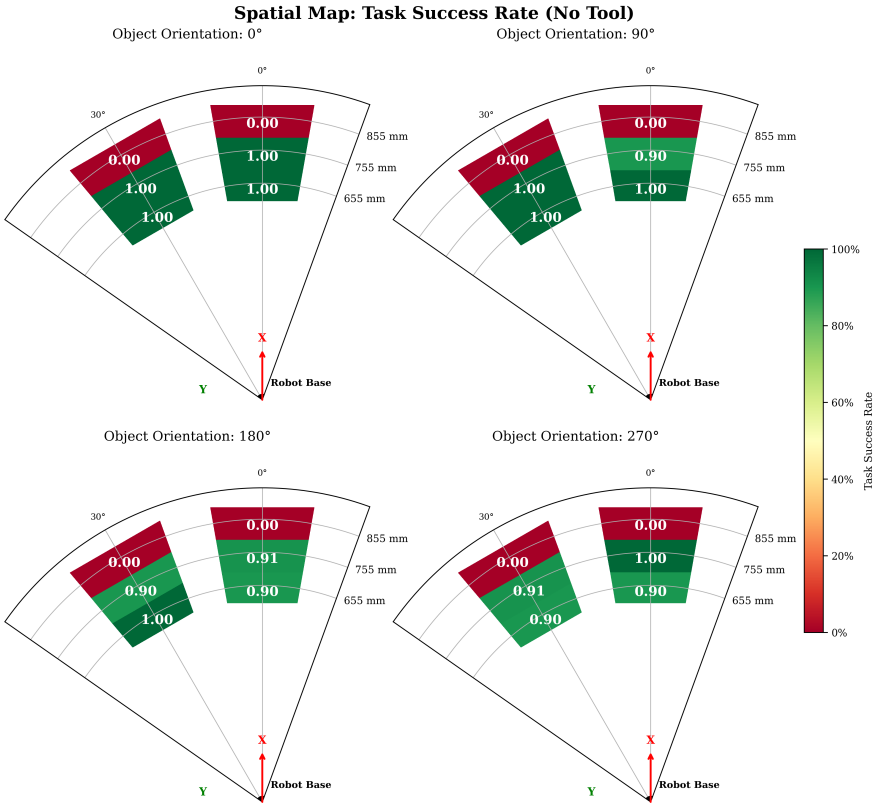
particular object orientations caused larger reprojection errors for the ArUco pose estimates, This is the main reason the 90° orientation performs poorly. These issues are addressed in the limitations discussed in chapter 7: adding a collision scene from point cloud data, and using a dedicated grasp-pose estimation module.

A.5 Comparison with Simulation

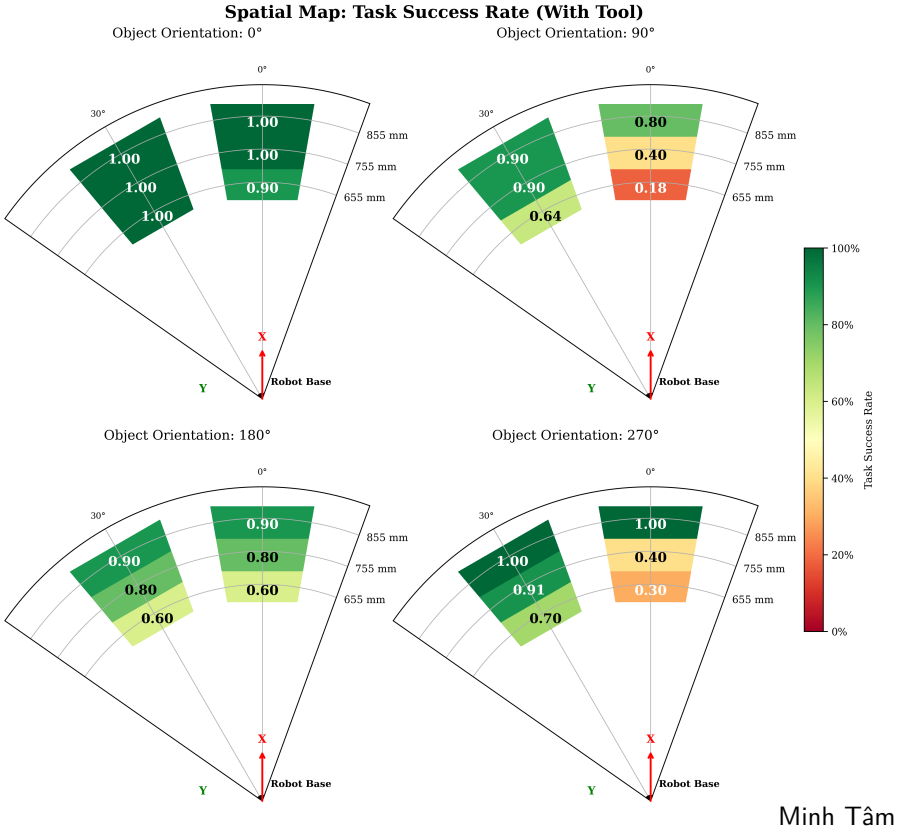
The real-world and simulation results agree on the main findings. In both studies, the No-Tool condition reaches 0% success at the maximum radius ($r = 855\,\text{mm}$ on hardware, $r > 0.87\,\text{m}$ in simulation), while the hammer condition still completes a substantial fraction of trials. In both, the hammer performs worse at close radii because of the extra drag step. In both real-world and simulation, the drag phase is identified as the kinematic bottleneck through low manipulability values, where the simulated results show a positive correlation with task success.

Two differences exist between the two studies. The real-world design uses a coarse discrete grid, so the exact location of the crossover radius cannot be calculated as precisely as in simulation. And the real-world trials show orientation- and sensing-related failures due to sensor noise not present in simulation.

An additional difference is worth noting in the other direction. In simulation, the T-Shape shows near-zero sensitivity to manipulability, which is attributed to a collision-wedging failure mode: MuJoCo uses non-deformable contacts, so when the T-Shape tool touches multiple surfaces at once it can become stuck, stopping the drag regardless of arm posture. The T-Shape was not included in these experiments on the real robot. However during some qualitative testing of the T-shape on the real robot, none of the trials showed any mechanical wedging. The contact-related failures that did occur were caused by the missing table in the collision model, not by the tool physically getting stuck. In the real world, contacts are compliant and a tool touching multiple surfaces will typically slip free rather than lock in place.

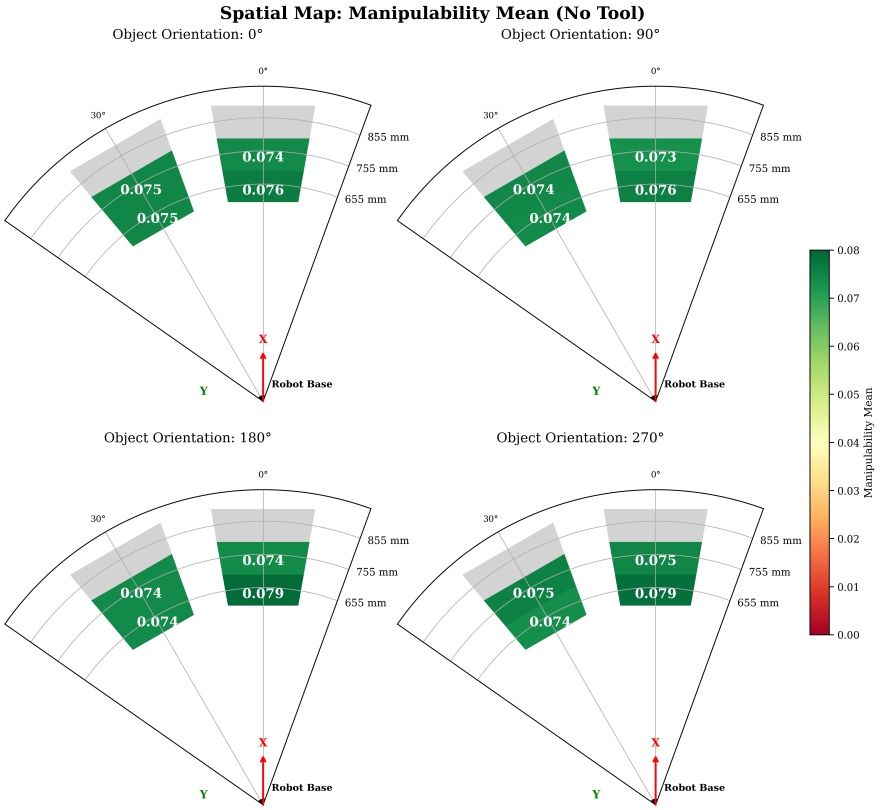


(a) Success rate plotted at azimuth $\theta \in \{0°, 30°\}$ for the No Tool condition. At 855mm, the robot never manages to complete the task and gets a success rate of 0.0.

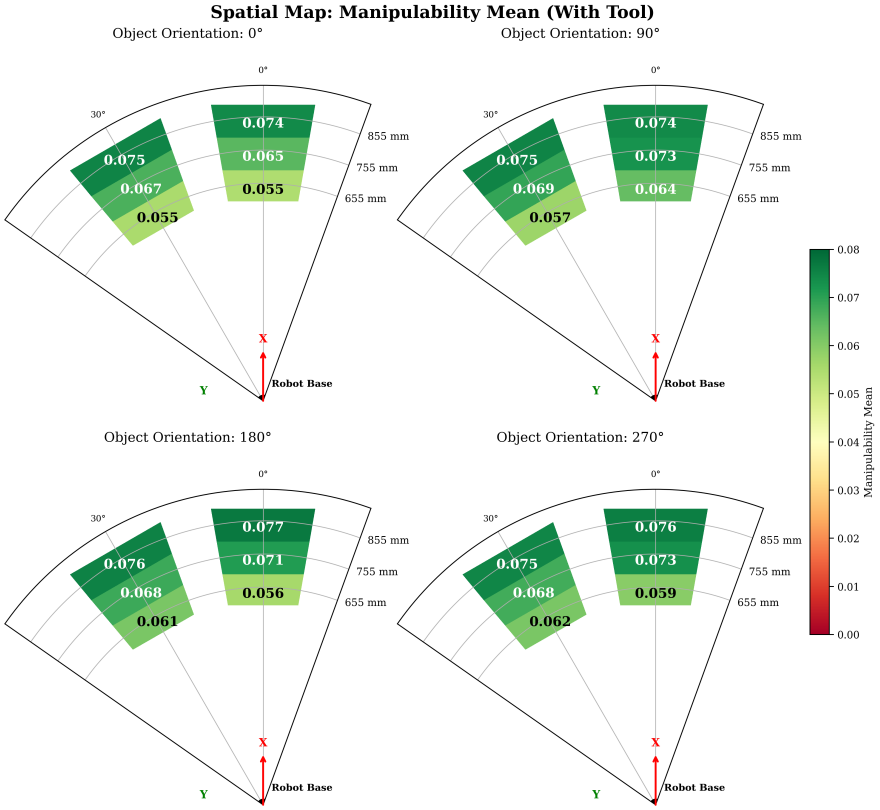


(b) Success rate plotted at azimuth $\theta \in \{0°, 30°\}$ for the Hammer condition. At 855mm, Success Rate is between 1.0 and 0.8.

Figure A.1: Polar heat maps of Success Rate for No Tool and Hammer conditions in real-world experiments.

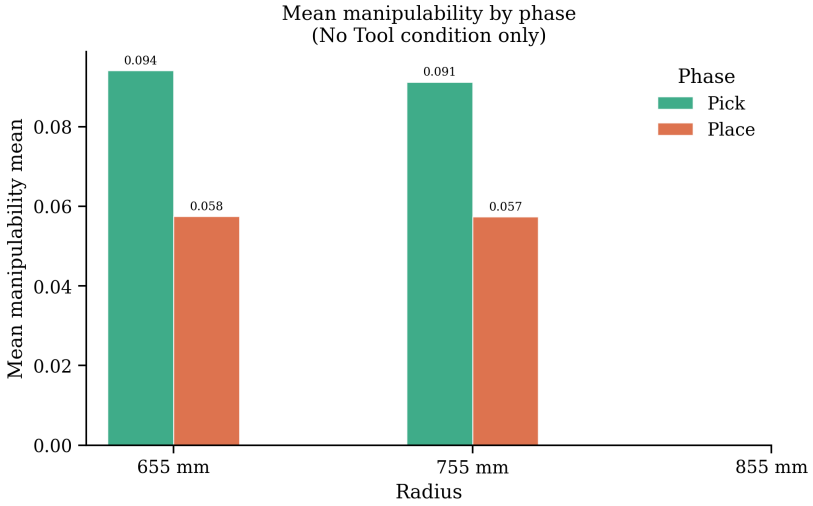


(a) Mean manipulability $\bar{w}$ plotted at azimuth $\theta \in \{0°, 30°\}$ for the No Tool condition.

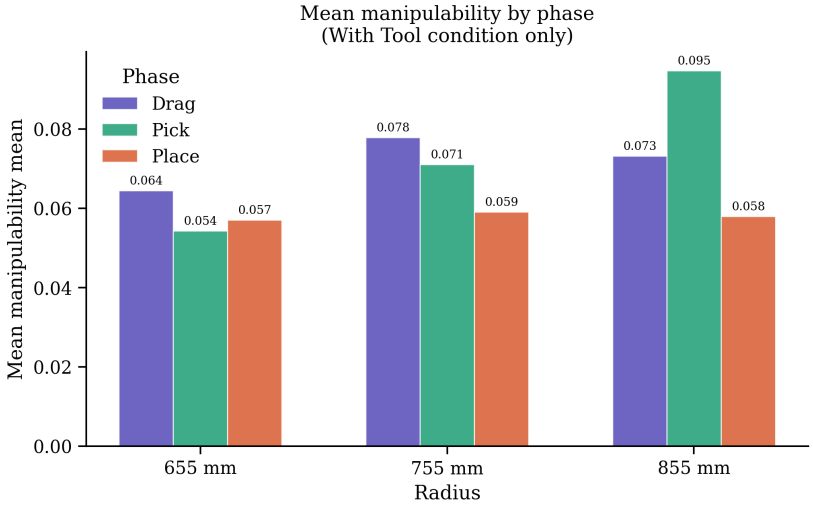


(b) Mean manipulability $\bar{w}$ plotted at azimuth $\theta \in \{0°, 30°\}$ for the Hammer condition.

Figure A.3: Polar heat maps of mean manipulability $\bar{w}$ for No Tool and Hammer conditions in real-world experiments.



(a) Mean manipulability by phases (Pick, Place) for the No Tool condition.



(b) Mean manipulability by phases (Pick, Place, Drag) for the Hammer condition

Figure A.4: Mean manipulability by phase for No Tool and Hammer condition

Bibliography

[1] Leona S. Aiken, Stephen G. West, and Raymond R. Reno. *Multiple Regression: Testing and Interpreting Interactions. Database*. Sage Publications, Inc., 1991.

[2] Haldun Akoglu. "User's Guide to Correlation Coefficients". In: *Turkish Journal of Emergency Medicine* 18.3 (Sept. 2018), pp. 91–93. ISSN: 24522473. DOI: 10.1016/j.tjem.2018.08.001.

[3] Iman Awaad, Gerhard Kraetzschmar, and Joachim Hertzberg. "Affordance-Based Reasoning in Robot Task Planning". In: June 2013.

[4] Gabriel Barth-Maron et al. "Affordances as Transferable Knowledge for Planning Agents". In: *AAI Fall Symposia*. 2014.

[5] BehaviorTree. *BehaviorTree.CPP Cppzmq*. BehaviorTree.CPP.

[6] J. Benton, Amanda Coles, and Andrew Coles. "Temporal Planning with Preferences and Time-Dependent Continuous Costs". In: *Proceedings of the International Conference on Automated Planning and Scheduling* 22 (May 2012), pp. 2–10. ISSN: 2334-0843, 2334-0835. DOI: 10.1609/icaps.v22i1.13509.

[7] Arvind Car et al. *PLATO: Planning with LLMs and Affordances for Tool Manipulation*. Sept. 2024. DOI: 10.48550/arXiv.2409.11580. arXiv: 2409.11580 [cs]. (Visited on 03/11/2025).

[8] Chang Wang, Koen V. Hindriks, and Robert Babuska. "Robot Learning and Use of Affordances in Goal-Directed Tasks". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo: IEEE, Nov. 2013, pp. 2288–2294. ISBN: 978-1-4673-6358-7 978-1-4673-6357-0. DOI: 10.1109/IROS.2013.6696676.

[9] Fu-Jen Chu et al. "Toward Affordance Detection and Ranking on Novel Objects for Real-World Robotic Manipulation". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019), pp. 4070–4077. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/lra.2019.2930364.

[10] Peter Corke. *Robotics, Vision and Control*. Vol. 118. Springer Tracts in Advanced Robotics. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-54412-0 978-3-319-54413-7. DOI: 10.1007/978-3-319-54413-7.

[11] Francisco Cruz et al. “Training Agents With Interactive Reinforcement Learning and Contextual Affordances”. In: *IEEE Transactions on Cognitive and Developmental Systems* 8.4 (Dec. 2016), pp. 271–284. ISSN: 2379-8920, 2379-8939. DOI: 10.1109/TCDS.2016.2543839.

[12] Neil T. Dantam, Swarat Chaudhuri, and Lydia E. Kavraki. “The Task-Motion Kit: An Open Source, General-Purpose Task and Motion-Planning Framework”. In: *IEEE Robotics & Automation Magazine* 25.3 (Sept. 2018), pp. 61–70. ISSN: 1070-9932, 1558-223X. DOI: 10.1109/MRA.2018.2815081.

[13] Yifei Dong et al. *Robustness-Aware Tool Selection and Manipulation Planning with Learned Energy-Informed Guidance*. Mar. 2026. DOI: 10.48550/arXiv.2506.03362. arXiv: 2506.03362 [cs.RO]. (Visited on 05/27/2026).

[14] Guoguang Du et al. “Vision-Based Robotic Grasping from Object Localization, Object Pose Estimation to Grasp Estimation for Parallel Grippers: A Review”. In: *Artificial Intelligence Review* 54.3 (Mar. 2021), pp. 1677–1734. ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-020-09888-5.

[15] Franka Emika. *Franka Emika Panda Technical Data*.

[16] Hao-Shu Fang et al. “GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[17] Richard E. Fikes and Nils J. Nilsson. “Strips: A New Approach to the Application of Theorem Proving to Problem Solving”. In: *Artificial Intelligence* 2.3-4 (Dec. 1971), pp. 189–208. ISSN: 00043702. DOI: 10.1016/0004-3702(71)90010-5.

[18] M. Fox and D. Long. “PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains”. In: *Journal of Artificial Intelligence Research* 20 (Dec. 2003), pp. 61–124. ISSN: 1076-9757. DOI: 10.1613/jair.1129. arXiv: 1106.4561 [cs].

[19] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. “PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning”. In: *Proceedings of the International Conference on Automated Planning and Scheduling* 30 (June 2020), pp. 440–448. ISSN: 2334-0843, 2334-0835. DOI: 10.1609/icaps.v30i1.6739.

[20] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. “Sampling Based Methods for Factored Task and Motion Planning”. In: *The International Journal of Robotics Research* 37.13-14 (Dec. 2018), pp. 1796–1825. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364918802962. arXiv: 1801.00680 [cs].

[21] Caelan Reed Garrett et al. “Integrated Task and Motion Planning”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (May 2021), pp. 265–293. ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-091420-084139.

[22] Ilche Georgievski and Marco Aiello. *An Overview of Hierarchical Task Network Planning*. Mar. 2014. DOI: 10.48550/arXiv.1403.7426. arXiv: 1403.7426 [cs]. (Visited on 05/18/2025).

[23] Asghar Ghasemi and Saleh Zahediasl. “Normality Tests for Statistical Analysis: A Guide for Non-Statisticians”. In: *International Journal of Endocrinology and Metabolism* 10.2 (Dec. 2012), pp. 486–489. ISSN: 1726-913X, 1726-9148. DOI: 10.5812/ijem.3505.

[24] James J. Gibson. *The Theory of Affordances. Perceiving, Acting, and Knowing: Toward an Ecological Psychology*. Boston, 1977.

[25] Jeongmin Jeon et al. "Primitive Action Based Combined Task and Motion Planning for the Service Robot". In: *Frontiers in Robotics and AI* 9 (Feb. 2022), p. 713470. ISSN: 2296-9144. DOI: 10.3389/frobt.2022.713470.

[26] Jihong Lee. "A Study on the Manipulability Measures for Robot Manipulators". In: *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97*. Vol. 3. Grenoble, France: IEEE, 1997, pp. 1458–1465. ISBN: 978-0-7803-4119-7. DOI: 10.1109/IROS.1997.656551.

[27] Alexander Kirillov et al. *Segment Anything*. 2023. DOI: 10.48550/ARXIV.2304.02643. (Visited on 08/05/2025).

[28] Hoi-Yin Lee et al. "Non-Prehensile Tool-Object Manipulation by Integrating LLM-based Planning and Manoeuvrability-Driven Controls". In: *Robotics and Computer-Integrated Manufacturing* 100 (Aug. 2026), p. 103231. ISSN: 07365845. DOI: 10.1016/j.rcim.2026.103231.

[29] Ian Lenz, Honglak Lee, and Ashutosh Saxena. *Deep Learning for Detecting Robotic Grasps*. 2013. DOI: 10.48550/ARXIV.1301.3592. (Visited on 08/05/2025).

[30] Vladimir Lifschitz. "What Is Answer Set Programming?" In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. Chicago, Illinois, 2009, pp. 1594–1597. ISBN: 978-1-57735-368-3.

[31] Christopher Lörken and Joachim Hertzberg. "Grounding Planning Operators by Affordances". In: *International Conference on Cognitive Systems (CogSys*. 2008, pp. 79–84.

[32] Khuong Nguyen and Yoonsuck Choe. "Emergence of Different Modes of Tool Use in a Reaching and Dragging Task". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. Shenzhen, China: IEEE, July 2021, pp. 1–8. ISBN: 978-1-6654-3900-8. DOI: 10.1109/IJCNN52387.2021.9533775.

[33] Nils J. Nilsson. *Shakey the Robot*. Technical Note No. 323. Menlo Park, CA: Artif. Intell. Cent SRI Int., 1984.

[34] Donald A. Norman. *The Design of Everyday Things*. First Basic paperback, [Nachdr.] New York: Basic Books, 2008. ISBN: 978-0-465-06710-7.

[35] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2023. DOI: 10.48550/ARXIV.2304.07193. (Visited on 08/05/2025).

[36] Peter Peduzzi et al. "A Simulation Study of the Number of Events per Variable in Logistic Regression Analysis". In: *Journal of Clinical Epidemiology* 49.12 (Dec. 1996), pp. 1373–1379. ISSN: 08954356. DOI: 10.1016/S0895-4356(96)00236-3.

[37] D. Pellier et al. *HDDL 2.1: Towards Defining an HTN Formalism with Time*. June 2022. DOI: 10.48550/arXiv.2206.01822. arXiv: 2206.01822 [cs]. (Visited on 05/18/2025).

[38] Christoph Pohl et al. "Affordance-Based Grasping and Manipulation in Real World Applications". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 9569–9576. ISBN: 978-1-7281-6212-6. DOI: 10.1109/IROS45743.2020.9341482.

[39] Nathan Ratliff et al. “CHOMP: Gradient Optimization Techniques for Efficient Motion Planning”. In: *2009 IEEE International Conference on Robotics and Automation*. Kobe: IEEE, May 2009, pp. 489–494. ISBN: 978-1-4244-2788-8. DOI: 10.1109/ROBOT.2009.5152817.

[40] Justine Rochon, Matthias Gondan, and Meinhard Kieser. “To Test or Not to Test: Preliminary Assessment of Normality When Comparing Two Independent Samples”. In: *BMC Medical Research Methodology* 12.1 (Dec. 2012), p. 81. ISSN: 1471-2288. DOI: 10.1186/1471-2288-12-81.

[41] Erol Şahin et al. “To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control”. In: *Adaptive Behavior* 15.4 (Dec. 2007), pp. 447–472. ISSN: 1059-7123, 1741-2633. DOI: 10.1177/1059712307084689.

[42] Boyang Ti et al. “An Optimal Control Formulation of Tool Affordance Applied to Impact Tasks”. In: *IEEE Transactions on Robotics* 40 (2024), pp. 1966–1982. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2024.3365993.

[43] Marc A Toussaint et al. “Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning”. In: *Robotics: Science and Systems Foundation* (June 2018).

[44] Jonathan Tremblay et al. *Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects*. Sept. 2018. DOI: 10.48550/arXiv.1809.10790. arXiv: 1809.10790 [cs]. (Visited on 01/26/2026).

[45] Jane Van Lawick-Goodall. “Tool-Using in Primates and Other Vertebrates”. In: *Advances in the Study of Behavior*. Vol. 3. Elsevier, 1971, pp. 195–249. ISBN: 978-0-12-004503-7. DOI: 10.1016/S0065-3454(08)60157-6.

[46] Mengdi Xu et al. *Creative Robot Tool Use with Large Language Models*. Oct. 2023. DOI: 10.48550/arXiv.2310.13065. arXiv: 2310.13065 [cs]. (Visited on 03/11/2025).

[47] Tsuneo Yoshikawa. “Manipulability of Robotic Mechanisms”. In: *The International Journal of Robotics Research* 4.2 (June 1985), pp. 3–9. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/027836498500400201.