Autonomous Object Handling in Collaborative Robotics

by

Philippe Nadeau

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Institute for Aerospace Studies
University of Toronto

Autonomous Object Handling in Collaborative Robotics

Philippe Nadeau
Doctor of Philosophy
Institute for Aerospace Studies
University of Toronto
2026

Abstract

There is a growing demand for robots that can perform efficient object handling operations, autonomously and safely, in unstructured human-shared spaces. *Collaborative robots* could take on various menial tasks as needed, allowing skilled workers to focus on more valuable activities. In this context, object handling directives may be abstract, requiring robots to independently determine suitable plans for transporting and placing objects. However, to truly alleviate the burden on human workers, it is essential that robots exhibit a high level of self-reliance and efficiency. To this end, this thesis presents key contributions to enhance the autonomy and efficiency of mobile manipulators in collaborative environments through novel algorithms addressing key aspects of the object handling problem. Specifically, we introduce two approaches to improve the speed and accuracy of inertial parameter identification with collaborative robots. Our first approach addresses the main sources of noise during the identification procedure by approximating rigid body dynamics for slower motions and transitioning to the complete dynamics model for motions with higher signal-to-noise ratios. Our second approach combines visual and force-torque measurements to enable inertial parameter identification using safer motions, making it suitable for use around humans. We also propose a method to assess the robustness of objects in frictional contact to force perturbations, which provides a foundation for autonomous robot decision-making when interacting with objects in frictional contact. Finally, we introduce an object placement planning algorithm making use of our robustness assessment to generate stable placements of rigid objects much more efficiently. Overall, the algorithms developed in this thesis improve mobile manipulators' autonomy by providing efficient inertial identification, object transportation, and placement planning capabilities, paving the way for self-reliant robots in human-shared spaces.

Epigraph

Thinking begins when you ask really difficult questions.

Slavoj Žižek.

Acknowledgements

Innovation can only appear in a society that fosters diversity of thought and provides the necessary conditions for individuals to explore new ideas. During my doctoral degree, I was working in a safe environment, encouraged to read widely and form my own opinions, and I was given the opportunity to travel the world to share my work. Recognizing that people struggled and fought to provide me with the freedom, safety, and resources to pursue my research, I feel indebted to stand up for those who are not provided with the same opportunities.

Within the vast network of people who supported me in my scientific endeavours, I would like to thank some of those that have been most personally involved in my academic journey, which started many years ago. I would like to sincerely thank my first supervisor, Rachid Aissaoui, for initiating me to robotics research and for showing me how robotics can truly impact people's well-being. In addition, I would like to extend my thanks to Vincent Duchaine, who took me under his wing and propelled me forward in my research career. I am grateful for his continuous support throughout my PhD. To Hannah Stuart, under whose guidance I wrote my first paper, and who mentored me with great care, I express my profound appreciation. I would also recognize the outstanding work of my committee members, Steven Waslander and Gabriele D'Eleuterio, for delivering me constructive feedback in the kindest manner. Of course, I am deeply thankful to my advisor, Jonathan Kelly, for his dedication to the research group. While there are undoubtedly many great scientists, few possess Jonathan's rare combination of scientific excellence, genuine care for his students, and unwavering commitment to fostering a collaborative and supportive research environment.

I was fortunate to be surrounded by kind and supportive people who made a huge difference in my success and well-being throughout my PhD. I'm especially grateful of the support provided by the residents of the *Three STARS Hotel* (Adam, Andrej, Olivier, and Emily), who always offered their help when I needed it. Likewise, I would like to thank the members of the laboratory for creating a welcoming and stimulating environment, and for the many insightful discussions that helped shape my research.

J'ai eu la chance de grandir dans une famille exceptionelle qui valorise la science, l'éducation, et la curiosité. C'est bien vrai: « Ce sont les encouragements et les marques d'affection de tous les membres de ma famille, tout au long de mes études et spécialement durant les moments les plus difficiles, qui ont, par-dessus tout, rendu possible la persévérence qui a conduit à mes succès » (Nadeau, 1981). Je suis particulièrement redevable à ceux qui m'ont aidé à m'installer à Toronto et à ceux qui m'ont accueilli à bras ouverts lors de mes passages fréquents à Montréal. Le support indéfectible d'une famille et d'un réseau d'amis tissés serrés a fait toute la différence, et j'espère avoir l'occasion de vous rendre la pareille. Je dois spécialement remercier Flavie d'avoir eu le courage d'embarquer dans cette aventure avec moi, malgré les sacrifices nécessaires à son accomplissement.

Contents

Notation

In this dissertation, lowercase bold letters denote column vectors, while uppercase bold letters represent matrices. Common symbols are summarized in the following table.

Expression	**Description** (w.r.t ≡ with respect to)
$\mathbf{1}_3$	Identity matrix of size 3×3
$\hat{\mathbf{i}}$	Unit-length column vector $\mathbf{i}$
$\mathbf{a}^\mathsf{T}$	Transpose of $\mathbf{a}$
$[\mathbf{a}]_\times$	Skew-symmetric operation on $\mathbf{a}$
$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^\mathsf{T}\mathbf{b}$	Dot product between vectors $\mathbf{a}$ and $\mathbf{b}$
$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$	Cross product of vector $\mathbf{a}$ with vector $\mathbf{b}$
$\mathbf{M}^{-1}$	Inverse of matrix $\mathbf{M}$
$\boldsymbol{\mathcal{F}}_o$	Cartesian reference frame named o
${}^c_b\mathbf{p}_a$	Position of a w.r.t. b, expressed in c
${}_b\mathbf{R}_a$	Rotation/Orientation of a w.r.t. b
${}^c_b\mathbf{T}_a$	Pose of a w.r.t. b, with position expressed in c
$\dot{\mathbf{a}}$	First derivative of $\mathbf{a}$ w.r.t. time
${}^c_b\mathbf{v}_a$	Linear velocity of a relative to b, expressed in c
$\boldsymbol{\omega}$	Angular velocity
$\mathbf{a}, \boldsymbol{\alpha}$	Respectively the linear and angular accelerations
$\rho(\mathbf{p})$	Mass density function defined over point position $\mathbf{p}$
m_b	Mass of b
$\mathbf{c}$	Centre of mass, also symbolized with ◒
${}^c_b\mathbf{I}_a$	Inertia tensor of a computed w.r.t. b, expressed in c
$\boldsymbol{\phi}$	A 10×1 vector of inertial parameters
${}^c_b\boldsymbol{\mathcal{I}}_a$	Spatial (6D) inertia matrix of body a computed w.r.t. b, expressed in c
${}^c_b\mathbf{f}_a$	Force exerted at a, experienced in b, expressed in c
${}^c_b\boldsymbol{\tau}_a$	Torque exerted at a, experienced in b, expressed in c
${}^c_b\mathbf{w}_a = \left[{}^c_b\mathbf{f}_a^\mathsf{T},\ {}^c_b\boldsymbol{\tau}_a^\mathsf{T}\right]^\mathsf{T}$	Wrench (6D)
${}^c_b\mathbf{E}_a = m_a\, {}^c_b\mathbf{v}_a$	Linear momentum of a relative to b, expressed in c
${}^c_b\boldsymbol{\varepsilon}_a$	Real angular momentum when velocity is ${}^c_b\boldsymbol{\nu}_a = \left[{}^c_b\mathbf{v}_a^\mathsf{T},\ {}^c_b\boldsymbol{\omega}_a^\mathsf{T}\right]^\mathsf{T}$
$\mathbf{g} \approx 9.81\hat{\mathbf{g}}$	Gravitational force at the surface of the Earth

Chapter 1

Introduction

> As development continues, and robots become more and more adept, a large segment of humanity will be released for endeavors more worthy of human beings.
>
> Joseph Engelberger, 1980.

The canonical task of a robot manipulator involves picking an object, moving it to a target location, and placing it amongst other objects. In fact, the first mass-produced industrial robot, the Unimate, was used to pick up hot pieces of metal from a die-casting machine and transport them to a cooling station, a dangerous operation. To this day, development in robotics has been driven by the desire to avoid relying on human workers for dangerous, repetitive, or tedious tasks. Human intelligence, agility, and dexterity are rightly deemed better utilized in tasks requiring creativity, decision-making, and social interaction.

The World Robotics Report 2024 highlights a growing demand for mobile manipulators in handling tasks, driven by a desire for local manufacturing but faced with labour scarcity (International Federation of Robotics, 2024). Small and medium-sized businesses (SMBs) are increasingly adopting robotics to remain afloat and are particularly affected by the challenge of small batch sizes, for which the cost of automation can be prohibitive. Particularly relevant for SMBs, Billard and Kragic (2019) highlight that greater collaboration between humans and robots would enable combining the unrivaled human intelligence with the strength and repeatability of robots. This desire for an improved human-robot synergy has spurred the advent of *collaborative robotics*, a branch of robotics focusing on the development of robots that can safely operate alongside humans. However, to avoid becoming a burden on human workers, collaborative robots (cobots) must be able to plan their actions independently and adapt in dynamic environments. Currently, manipulators lack of versatility and adaptability restricts their use to well-defined tasks in unchanging environments (Christensen, 2024). This thesis focuses on enhancing the autonomy of col-

laborative mobile robots in object handling tasks, a key challenge impeding the efficiency of robot manipulators (Alterovitz et al., 2016).

The operations that this thesis contributes to facilitate is illustrated in Figure 1.1, where an autonomous mobile robot (AMR) is responding to high-level object handling directives (e.g., "bring the red and green objects from the conveyor to workstation #2"). The first step involves picking up two objects from a conveyor and placing them on the robot's tray. Since humans are far from the robot at this point, the robot might use its sensors and actuators at a higher speed to quickly identify important object characteristics (e.g., mass and shape) during the loading operation. After navigating to the second stop, the robot must pick up a blue object from a human worker's station while operating at a slower speed to ensure the safety of the nearby human worker. In Chapter 3, we present algorithms to quickly identify the inertial parameters of manipulated objects while respecting the safety constraints of collaborative operations. The AMR must then move to the second stop as quickly as possible while ensuring that the objects remain stable on the tray. In Chapter 4, we present an algorithm to determine the maximal acceleration that the AMR can generate without causing instabilities in the object stack it carries. The third step involves placing the objects on a shelf cluttered with other objects whose characteristics are presumably known to the robot. In Chapter 5, we present a placement planning algorithm that can find stable placements in arbitrary assemblies of rigid objects under frictional contacts. In the final step, the AMR docks to a charging station, where it can recharge its batteries and wait for new directives.

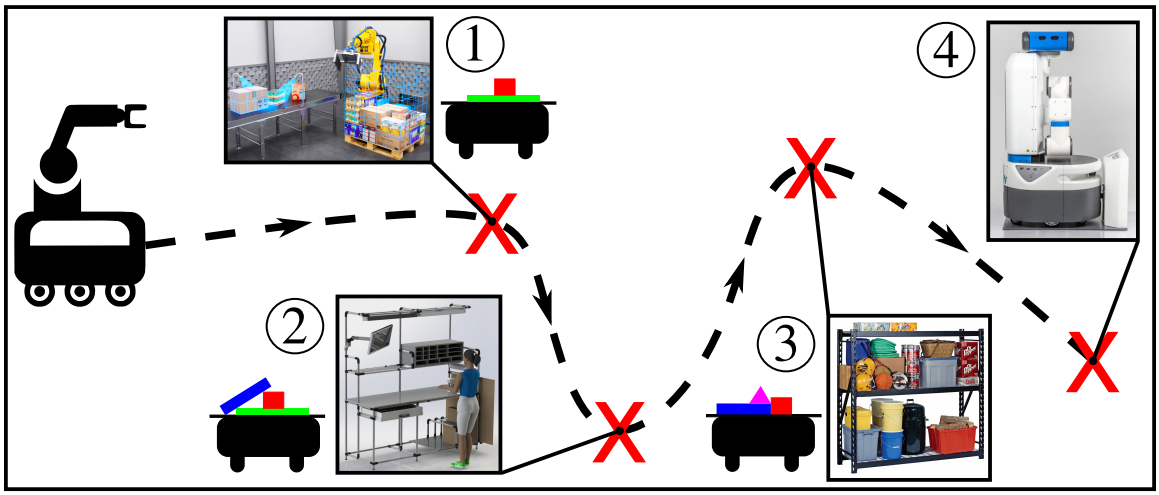


Figure 1.1: Illustration of the autonomous object handling problem considered in this thesis. An AMR is handling objects in an industrial environment as efficiently as possible. The first task consists of loading two objects from a conveyor. The second stop involves safely picking up a blue object from a person's workspace. The third step consist of placing objects on a shelf, before finally docking to a charging station. Image credits are provided in Section 6.2.

In essence, the task of autonomous object handling involves a sequence of picking, plac-

ing, and transporting operations. Each of these operations requires careful planning to ensure the stability of the objects and the efficiency of the execution. Picking an object typically involves generating a set of grasp candidates, evaluating the probability of each grasp's success, and executing the grasp that is the most likely to succeed. Object placement entails finding a placement pose that meets a set of directives or constraints, followed by positioning the object to that pose. Finally, object transportation involves planning a motion trajectory to carry a set of objects to a target location, ensuring the stability of the payload along the way. While grasp planning has received significant research attention, placement planning remains comparatively underexplored with about 22 times fewer publications on the topic according to Elsevier's Scopus database.[1] We posit that this gap in research is due to the complexity of the placement planning problem, which requires reasoning about the stability of contacting objects, a problem whose computational complexity has been proved to be very high (i.e., NP-hard) (Baraff, 1993). In turn, the stability of contacting objects depends on their inertial parameters (i.e., mass, centre of mass, moments of inertia), which cannot be directly measured and must be identified from sensor data. As depicted in Figure 1.2, this thesis addresses key elements of the autonomous object handling problem in a vertically-integrated manner, from inertial parameter identification to placement planning, to improve the autonomy of mobile manipulators in object handling tasks.

1.1 Original Contributions

This thesis presents three key contributions to enhance the autonomy and efficiency of mobile manipulators in collaborative environments. The first contribution consists of algorithms for rapid and accurate inertial parameter identification of manipulated objects, leveraging visual and force-torque data to enable safe operations around humans. The second contribution proposes a physically-grounded method to assess the robustness of assemblies made from rigid objects of any shape and mass distribution. The third contribution introduces a stability-aware placement planning method that enables efficient placements in general rigid-object environments. Overall, the algorithms developed in this thesis enhance the autonomy of mobile manipulators by equipping them with efficient inertial identification and placement planning capabilities, paving the way for more capable robots in human-shared spaces.

Associated Publications

- **P. Nadeau**, M. Giamou and J. Kelly, "Fast Object Inertial Parameter Identification for Collaborative Robots," 2022 International Conference on Robotics

[1] Scopus catalogs 543 publications with "grasp planning" and 25 publications with "placement planning" in paper titles or abstracts amongst publications containing "robotics" or "robot manipulator" in the title or the abstract. Consulted on March 21 2025.

and Automation (ICRA), Philadelphia, PA, USA, 2022, pp. 3560-3566, doi: 10.1109/ICRA46639.2022.9916213.

- **P. Nadeau**, M. Giamou and J. Kelly, "The Sum of Its Parts: Visual Part Segmentation for Inertial Parameter Identification of Manipulated Objects," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023, pp. 3779-3785, doi: 10.1109/ICRA48891.2023.10160394.
- **P. Nadeau** and J. Kelly, "Stable Object Placement Planning From Contact Point Robustness," IEEE Transactions on Robotics, 2025, doi: 10.1109/TRO.2025.3577049.
- **P. Nadeau** and J. Kelly, "Robustness Assessment of Assemblies in Frictional Contact," IEEE Transactions on Automation Science and Engineering (Under review), doi: 10.48550/arXiv.2411.09810.

To support the development of the algorithms presented in this thesis, and the experiments conducted to validate them, we have developed a set of custom hardware tools. This includes the design and construction of two height-adjustable robot experiment tables, enabling safe and accurate positioning of the uFactory xArm 7 (xArm 7) and Franka Emika Research 3 (FR3) manipulators on their respective tables, as well as the design of mechanical adapters to enable the attachment of the gripper and camera to the xArm 7. Robots and their sensors have been calibrated to ensure accurate positioning and measurements. Software implementations have also been developed along the way, including a realistic simulation of the Robotiq 2F-85 gripper in the PyBullet engine[2], a Python driver for the Robotiq 2F-85 gripper[3], a Python driver for the ATI Axia 80 force-torque sensor (FT sensor)[4], and a kinematic calibration library for any serial manipulator[5]. All of these software implementations are made available as open-source software.

[2] Available at https://github.com/utiasSTARS/pyb-sim-models
[3] Available at https://github.com/PhilNad/2f85-python-driver
[4] Available at https://github.com/PhilNad/axia-python-driver
[5] Available at https://github.com/PhilNad/robot-arm-kinematic-calibration

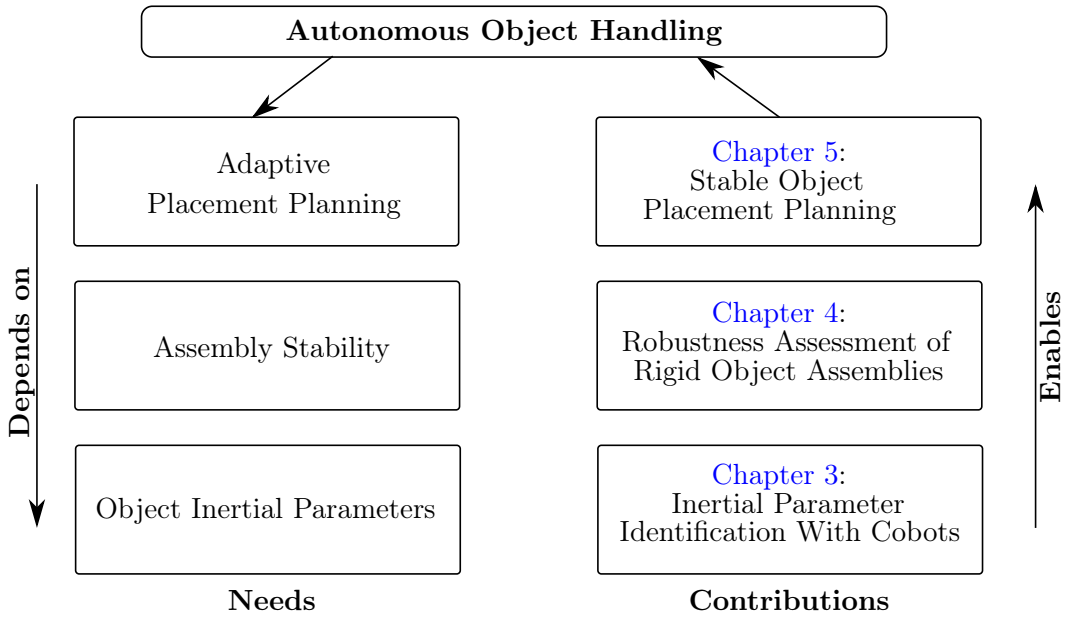


Figure 1.2: This thesis considers the problem of autonomous object handling with cobots, which involves grasping, transporting, and placing objects in contact with other objects. Autonomy, in this context, hinges on the capacity to model interactions in object assemblies. This thesis contributes vertically-integrated algorithms to improve the autonomy of cobots in object handling tasks.

Chapter 2

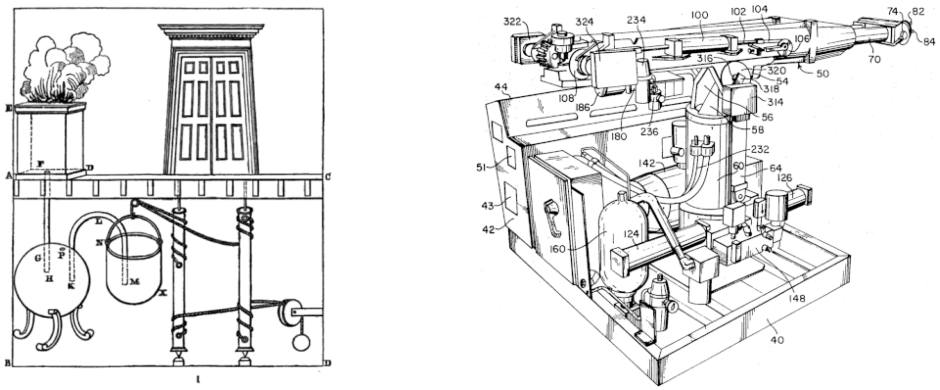
Preliminaries

> But since the manual arts are chiefly conversant in the moving of bodies, it comes to pass that geometry is commonly referred to their magnitudes, and mechanics to their motion.
>
> Isaac Newton, 1686.

This chapter provides an overview of the fundamental concepts and mathematical tools that are used throughout this thesis. We review elements of robot systems, define fundamental concepts related to the relative positioning of rigid bodies (i.e., their *geometry*), and introduce models relating robot motions to forces (i.e., *dynamics*).

2.1 Robot Systems

Today, a great variety of robot systems are used in industry, while some even permeate our daily lives. Although the *concept* of the robot dates back to Hellenistic Greece, notably with Heron of Alexandria's devices (one of which is shown in Figure 2.1), the term *robot* has only been introduced in the popular culture in the twentieth century. The term *robot* was coined by Czech writer Karel Čapek in his play *R.U.R.* (Rossum's Universal Robots) in 1920, and is derived from the word *robota*, which means *hard work*. Two decades later, departing from the cliché of rebellious mechanical beings, Isaac Asimov's *Robbie* and subsequent short stories of the *I, Robot* series greatly increased people's familiarity with robots. While science-fiction has often portrayed robots as sentient being with human-like form, today's robots are typically electro-mechanical systems designed to perform specific tasks in structured environments. We have to admit that robots are still closer to the early automata than to Čapek's and Asimov's creatures. Nonetheless, preconceptions instilled by popular culture are certainly influencing the development of robots, which people expect to be highly autonomous human-like machines.



(a) Heron's device (b) Unimate

Figure 2.1: (a) Heron of Alexandria built a mechanical device that perceived when fire was lit and automatically opened the temple doors. (b) Artwork from Engelberger's patent of the Unimate, the first industrial robot. Image credits are provided in Section 6.2.

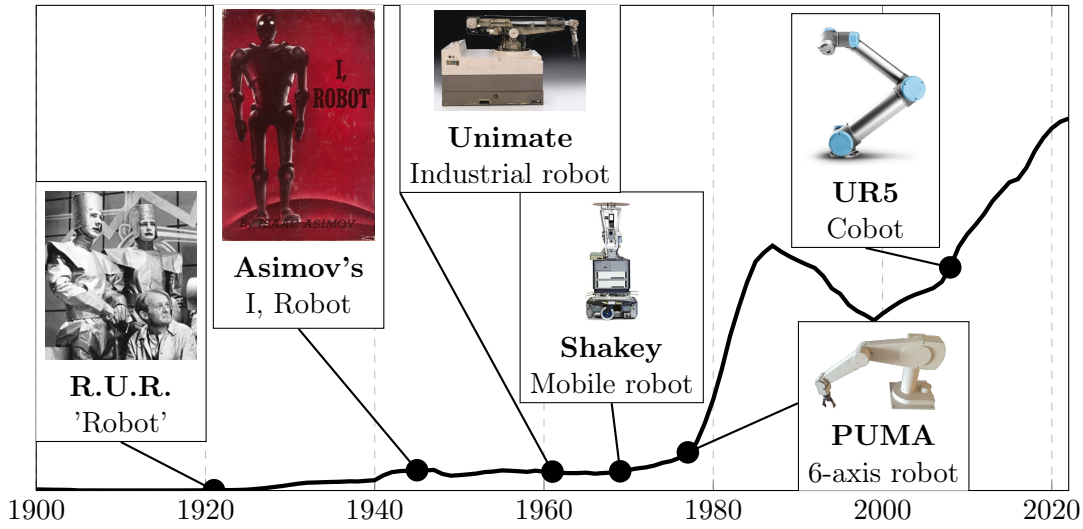


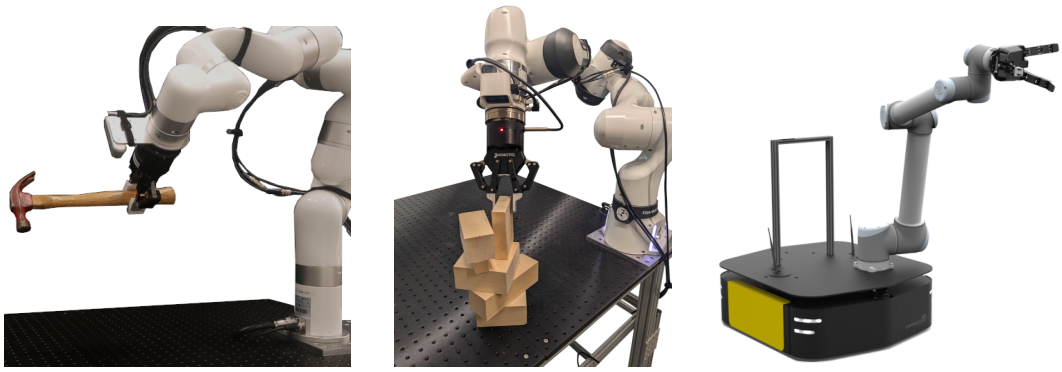
Figure 2.2: Important events in the history of robotics in relation with the popularity of the term "robot" in English literature. Image credits are provided in Section 6.2.

(a) 6-axis robot (b) SCARA robot (c) Delta robot

Figure 2.3: Examples of different robot structures. Image credits are provided in Section 6.2.

The Unimate was arguably the first mass-produced industrial robot, invented in 1959 by George Devol and introduced to the worldwide market by Joseph Engelberger. With three rotary and one linear (or *prismatic*) actuators, the Unimate was used with great success in die-casting operations, in which toxic fumes and high temperatures endangered human workers. The Unimate was a *serial* robot, meaning that each *link* is connected to the previous one through a *joint*, thereby forming a chain. The joints of a robot are often referred to as *axes*, and the most common type of robot is undoubtedly the *6-axis* serial manipulator. Robots with different structures have been cleverly devised to improve performance in specific tasks, at the expense of reduced versatility, two of which are illustrated in Figure 2.3. For instance, *SCARA* robots feature a serial chain of axes that are all parallel to gravity, enabling the weight of the payload to be supported by the robot's structure rather than by its actuators. In comparison, *Delta* robots possess a *parallel* structure, in which the end-effector (i.e., the link distal to the base) is connected to the base through multiple serial chains. Such a parallel structure enables Delta robots to perform high-speed pick-and-place operations in small workspaces, but with a limited range of motion.

The desire to increase the workspace in which a robot is constrained to operate has led to the advent of *mobile manipulators* that combine a mobile platform with a serial manipulator. In turn, mobile manipulation has motivated the development of lightweight robots that can be more easily transported by the mobile base. Additionally, enabling manipulators to move beyond the constraints of a fixed base has spurred the development of *collaborative robots* that implement safety constraints to reduce the risk of injury for nearby workers. Specifically, standards #10218 and #15066 from the International Organization for Standardization (ISO) define requirements under which a robot can operate collaboratively with a human worker, including a maximal end-effector velocity of 0.25 m/s and the capability to detect impacts. Force sensing can be implemented through force gauges positioned



(a) Gladys (b) Ada (c) Victor

Figure 2.4: The robots used in the experiments presented in this thesis.

in the joints of the manipulators enabling the detection of external forces exerted on the links of the robot. These *torque sensors* can also be used for torque control, where the desired end-effector position is reached by having the joint torques follow a desired trajectory. When greater accuracy is required, a dedicated force-torque sensor (FT sensor) can be used to directly measure the forces and torques exerted on the end-effector. Apart from force sensing, RGB-D cameras that concurrently capture colour and depth images have become ubiquitous in robotics, enabling mobile manipulators to adapt to changing environments. For instance, cameras can be used to detect obstacles in the workspace, to identify objects to be manipulated, or to track the position of the robot relative to the environment.

Work done as part of this thesis has relied on three robotic systems shown in Figure 2.4:

- A uFactory xArm 7 (xArm 7) manipulator equipped with a Robotiq FT-300s FT sensor and RealSense D435 RGB-D camera (Gladys[1]).
- A Franka Emika Research 3 (FR3) manipulator equipped with a ATI Axia 80 FT sensor, a Robotiq 2F-85 gripper, and a RealSense D415 RGB-D camera (Ada[2]).
- A Universal Robots UR10 (UR10) manipulator fixed atop a Clearpath Ridgeback mobile base and equipped with a Robotiq FT-300 FT sensor and Three-Finger gripper (Victor[3]).

The xArm 7 manipulator was used for the experiments performed as part of the work presented in Chapter 3, while the FR3 manipulator was used for the experiments presented in Chapter 5, and a simulation of the UR10 mobile manipulator was used for some of the experiments presented in Chapter 4.

[1] In honor of Dr. Gladys West.
[2] As a tribute to Ada Lovelace.
[3] In recognition of Dr. Victor Scheinman.

2.2 Geometry for Robotics

In "Leçons sur les applications du calcul infinitésimal à la géometrie" (1826), Cauchy defines the *radius vector* as the line going from a point assumed to be fixed to another point assumed to be mobile. While the prefix *radius* has been dropped from the term, the concept of a vector as the relative position of one point to another has remained. Vectors were eventually formalized and their use has been generalized to other physical quantities (e.g., velocity) and dimensionalities (e.g., Hamilton's quaternions). Crucially, vectors are not merely *tuples*—ordered lists of numbers—but mathematical objects respecting a set of axioms. Vectors can however be *represented* as tuples through the use of a *reference frame* map.

A *Cartesian reference frame* consists in a set of three unit-length basis vector $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ respecting

$$\hat{\mathbf{x}} \times \hat{\mathbf{y}} = \hat{\mathbf{z}}, \tag{2.1}$$

where $\times$ denotes Hamilton's *cross-product* operation.[4] A *skew-symmetric operator* $[\mathbf{u}]_\times$ can equivalently be used to perform the cross-product operation as a matrix multiplication,

$$\mathbf{u} \times \mathbf{v} = [\mathbf{u}]_\times \mathbf{v} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_z \end{bmatrix}, \tag{2.2}$$

for any vectors $\mathbf{u}$ and $\mathbf{v}$ in R^3. A vector $\mathbf{v}$ can be *expressed* in the Cartesian reference frame $\boldsymbol{\mathcal{F}}$ (henceforth simply referred to as *reference frame* or *frame*) by projecting $\mathbf{v}$ onto each basis vectors with

$$\begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_z \end{bmatrix} = \begin{bmatrix} \mathbf{v} \cdot \hat{\mathbf{x}} \\ \mathbf{v} \cdot \hat{\mathbf{y}} \\ \mathbf{v} \cdot \hat{\mathbf{z}} \end{bmatrix} = \underbrace{\begin{bmatrix} \hat{\mathbf{x}}^\mathsf{T} \\ \hat{\mathbf{y}}^\mathsf{T} \\ \hat{\mathbf{z}}^\mathsf{T} \end{bmatrix}}_{\boldsymbol{\mathcal{F}}} \mathbf{v} = \boldsymbol{\mathcal{F}}\mathbf{v}, \tag{2.3}$$

where $\cdot$ denotes the dot-product operation. In robotics, vectors usually represent physical quantities (e.g., distances, forces) associated with well-defined units. A reference frame for which the *scale* (e.g., millimetres, metres, kilometres) of the physical unit is defined is called a *coordinate system*. Hence, the length of a vector expressed in a coordinate system is well-defined as a physical quantity.

In this thesis, the RIGID notation convention (Nadeau, 2024) is used to denote vectors as ${}^{c}_{b}\mathbf{v}_{s}$ where $\mathbf{v}$ is the physical quantity of interest, b is the *basis*, s is the *subject*, and c is the *coordinate system* used to express $\mathbf{v}$. A vector starts from its basis and reaches its subject, representing a directed line segment in the vector space. For instance, ${}^{w}_{w}\mathbf{p}_{e}$ denotes the position of the robot end-effector e with respect to the world frame w and expressed in the

[4] An alternate convention, called *left-handed*, is to define $\hat{\mathbf{x}} \times \hat{\mathbf{y}} = -\hat{\mathbf{z}}$, which is non-standard and can lead to confusion.

world frame. The RIGID notation aims at being unambiguous, well-defined, and compliant with the ISO 80000-2:2019 *Standard on Quantities and Units (Part 2: Mathematics)*. The notation convention defines conditions under which a concise version of the notation can be used. Accordingly, in this thesis, a vector that is expressed in the same frame that it is defined with respect to is denoted concisely as ${}_b\mathbf{v}_s$ instead of ${}^b_b\mathbf{v}_s$.

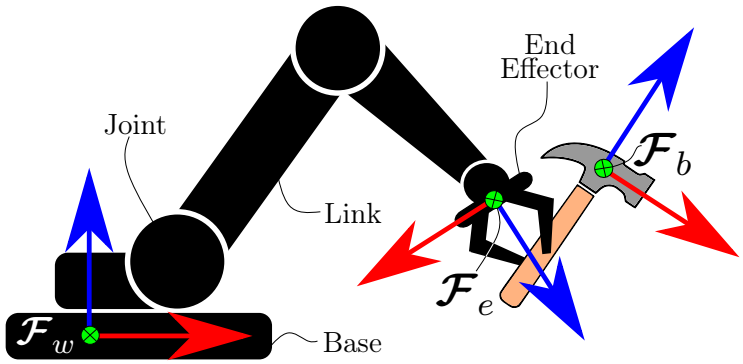


Figure 2.5: A serial manipulator holding a hammer in its end-effector. The world frame $\mathcal{F}_w$ is fixed to the robot base, the end-effector frame $\mathcal{F}_e$ is attached to the end-effector, and the body frame $\mathcal{F}_b$ is attached to the hammer.

Assuming that a frame $\mathcal{F}_e$ is attached to the end-effector of a robot, that a frame $\mathcal{F}_b$ is attached to an object held by the robot, and that a frame $\mathcal{F}_w$ is fixed in the workspace of the robot, the origin of $\mathcal{F}_b$ relative to $\mathcal{F}_e$ and expressed in $\mathcal{F}_w$ is given by

$$ {}^w_e\mathbf{p}_b = \mathcal{F}_w \, {}_e\mathbf{p}_b, \tag{2.4} $$

where ${}_e\mathbf{p}_b$ is exceptionally used to denote the position vector not expressed in any frame. This situation is illustrated in Figure 2.5. Similarly, the same position vector can be expressed in $\mathcal{F}_e$ via

$$ {}^e_e\mathbf{p}_b = \mathcal{F}_e \, {}_e\mathbf{p}_b. \tag{2.5} $$

Making use of the fact that the basis vectors in (2.3) are orthogonal and unit-length, we can state

$$ {}_e\mathbf{p}_b = \mathcal{F}_w^{\mathsf{T}} \, {}^w_e\mathbf{p}_b \tag{2.6} $$

$$ {}_e\mathbf{p}_b = \mathcal{F}_e^{\mathsf{T}} \, {}^e_e\mathbf{p}_b, \tag{2.7} $$

and therefore

$$\boldsymbol{\mathcal{F}}_w^{\mathsf{T}} \, {}^{w}_{e}\mathbf{p}_b = \boldsymbol{\mathcal{F}}_e^{\mathsf{T}} \, {}^{e}_{e}\mathbf{p}_b \tag{2.8}$$

$${}^{w}_{e}\mathbf{p}_b = \boldsymbol{\mathcal{F}}_w \boldsymbol{\mathcal{F}}_e^{\mathsf{T}} \, {}^{e}_{e}\mathbf{p}_b \tag{2.9}$$

$$= \begin{bmatrix} \hat{\mathbf{w}}_x \\ \hat{\mathbf{w}}_y \\ \hat{\mathbf{w}}_z \end{bmatrix} \begin{bmatrix} \hat{\mathbf{e}}_x & \hat{\mathbf{e}}_y & \hat{\mathbf{e}}_z \end{bmatrix} {}^{e}_{e}\mathbf{p}_b \tag{2.10}$$

$$= \underbrace{\begin{bmatrix} \hat{\mathbf{w}}_x \cdot \hat{\mathbf{e}}_x & \hat{\mathbf{w}}_x \cdot \hat{\mathbf{e}}_y & \hat{\mathbf{w}}_x \cdot \hat{\mathbf{e}}_z \\ \hat{\mathbf{w}}_y \cdot \hat{\mathbf{e}}_x & \hat{\mathbf{w}}_y \cdot \hat{\mathbf{e}}_y & \hat{\mathbf{w}}_y \cdot \hat{\mathbf{e}}_z \\ \hat{\mathbf{w}}_z \cdot \hat{\mathbf{e}}_x & \hat{\mathbf{w}}_z \cdot \hat{\mathbf{e}}_y & \hat{\mathbf{w}}_z \cdot \hat{\mathbf{e}}_z \end{bmatrix}}_{{}_{w}\mathbf{R}_e} {}^{e}_{e}\mathbf{p}_b, \tag{2.11}$$

where ${}_{w}\mathbf{R}_e$ is the *direction cosine matrix* that expresses the orientation of $\boldsymbol{\mathcal{F}}_e$ with respect to $\boldsymbol{\mathcal{F}}_w$. The leftmost column in ${}_{w}\mathbf{R}_e$ describes the direction of the x-axis of $\boldsymbol{\mathcal{F}}_e$ in $\boldsymbol{\mathcal{F}}_w$, the middle column describes the direction of the y-axis, and the rightmost column describes the direction of the z-axis.

Remark 1: Direction cosine matrix
The name of the matrix comes from the fact that each element in ${}_{w}\mathbf{R}_e$ is equal to the cosine of the angle between the corresponding basis vectors of $\boldsymbol{\mathcal{F}}_e$ and $\boldsymbol{\mathcal{F}}_w$ (since $\hat{\mathbf{u}} \cdot \hat{\mathbf{v}} = \|\hat{\mathbf{u}}\| \, \|\hat{\mathbf{v}}\| \cos(\theta) = \cos(\theta)$). In this thesis, we will use the term *rotation matrix* or *orientation matrix* to refer to the direction cosine matrix.

Due to the special structure of ${}_{w}\mathbf{R}_e$, in which each element is the projection of a frame's unit-length basis vector onto another frame's unit-length basis vector, ${}_{w}\mathbf{R}_e$ is orthonormal, implying that

$${}_{w}\mathbf{R}_e^{-1} = {}_{w}\mathbf{R}_e^{\mathsf{T}}, \tag{2.12}$$

$${}_{w}\mathbf{R}_e \, {}_{e}\mathbf{R}_w = \mathbf{1}_3, \tag{2.13}$$

$${}_{w}\mathbf{R}_e^{\mathsf{T}} = {}_{e}\mathbf{R}_w. \tag{2.14}$$

The rotation matrix can be used to map vector coordinates from one frame to another with

$${}^{w}_{e}\mathbf{p}_b = {}_{w}\mathbf{R}_e \, {}^{e}_{e}\mathbf{p}_b \tag{2.15}$$

for any $\boldsymbol{\mathcal{F}}_e$, $\boldsymbol{\mathcal{F}}_w$, and $\boldsymbol{\mathcal{F}}_b$. Note that the vector is unaffected by the rotation matrix, only its representation is changed. As for any vector, position vectors can be added or subtracted as long as they are expressed in the same frame.

A reference frame is defined by the position of its origin and the orientation of its axes, and this information is commonly bundled into a single object called a *pose*. The pose can

be represented by a homogeneous transformation matrix

$$ {}_e\mathbf{T}_w = \begin{bmatrix} {}_e\mathbf{R}_w & {}_e\mathbf{p}_w \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.16} $$

that can be used to map vectors from one frame to another with

$$ {}_e\mathbf{p}_b = {}_e\mathbf{T}_w \begin{bmatrix} {}_w\mathbf{p}_b \\ 1 \end{bmatrix} = {}_e\mathbf{R}_w \, {}_w\mathbf{p}_b + {}_e\mathbf{p}_w. \tag{2.17} $$

The transformation performing the reversed operation to ${}_e\mathbf{T}_w$ is given by

$$ {}_e\mathbf{T}_w^{-1} = \begin{bmatrix} {}_e\mathbf{R}_w^\mathsf{T} & -{}_e\mathbf{R}_w^\mathsf{T} \, {}_e\mathbf{p}_w \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}_w\mathbf{R}_e & {}_w\mathbf{p}_e \\ 0 & 1 \end{bmatrix} = {}_w\mathbf{T}_e, \tag{2.18} $$

where $\mathbf{R}^{-1} = \mathbf{R}^\mathsf{T}$ has been used. Rotation and rigid transformation matrices can be composed through matrix pre-multiplication with[5]

$$ {}_w\mathbf{R}_b = {}_w\mathbf{R}_e \, {}_e\mathbf{R}_b \tag{2.19} $$

$$ {}_w\mathbf{T}_b = {}_w\mathbf{T}_e \, {}_e\mathbf{T}_b, \tag{2.20} $$

where the subject of the transformation that is pre-multiplied must match the basis of the transformation that is post-multiplied.

2.3 Dynamics for Robotics

2.3.1 Bodies

In general, a *body* can be considered to be a collection of closely related elements put together (by nature or otherwise) to form a contiguous entity. For instance, a "body of water" refers to a contiguous collection of water molecules (and possibly impurities) while a "body of knowledge" refers to a set of information pieces that are related to a field of study. In the context of robotics, bodies usually refer to physical objects, like the links of a robot or items that are meant to be interacted with. An expanding field of robotics considers bodies that are mechanically compliant, or *soft*. Robots able to work with soft bodies are poised to improve the safety of robot systems and improve grasping capabilities. In this thesis, however, we focus on bodies that are presumed to be *rigid*. Such bodies have the particularity that the relative position between any two points in the body remains constant over time.

[5] Here, the composition is *active* — each transformation is defined relative to a moving frame. Alternatively, *passive* transformations that are defined relative to a fixed frame could be composed through matrix post-multiplication.

Assuming that bodies are rigid greatly simplifies many aspects of robot modeling, planning, and control, and is therefore an assumption commonly made in the field. For instance, for a rigid body with known shape, the position of all points in the body can be concisely described by the pose of a frame $\mathcal{F}_b$ attached to the body. Solving the motion planning problem, in which contact with obstacles must be avoided when moving, is considerably easier with rigid robots for which the position of all points can be described by a single pose per link.

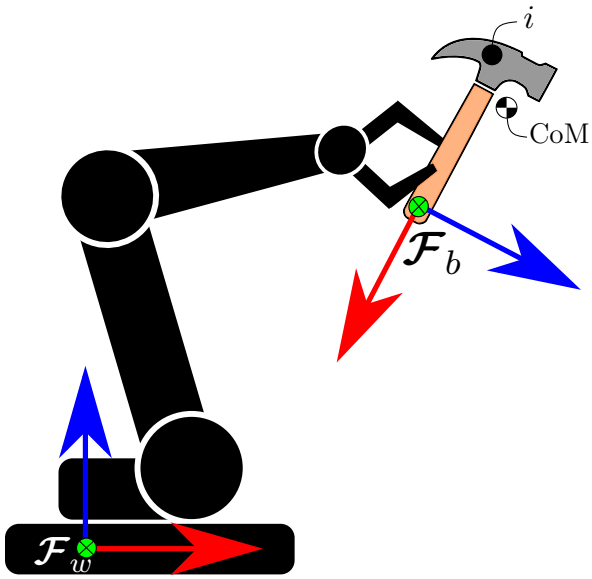


Figure 2.6: A hammer is being manipulated by a robot arm whose base link is attached to a fixed frame $\mathcal{F}_w$. All points on the hammer (like the point labeled i) are rigidly attached to its body frame $\mathcal{F}_b$ with the centre of mass labeled CoM.

2.3.2 Mass Distributions

The fact that the relative position between points in rigid bodies is fixed over time, combined with the assumption that the mass of each element in the body is also unchanging (i.e., assuming that the body is chemically inert), makes it possible to simplify the description of a body's response to external forces. Specifically, the zeroth, first, and second moments of a body's mass distribution are sufficient to describe the body's *inertia*.

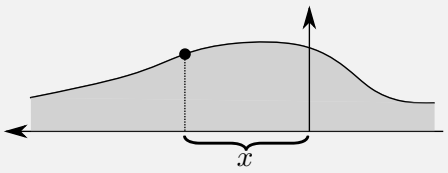
In mechanics, the *moment* of a vector refers to the product $\mathbf{p} \times \mathbf{q}$ where $\mathbf{q}$ is a physical quantity and $\mathbf{p}$ is a position vector. For instance, the moment of force (also called *torque*) is $\mathbf{p} \times \mathbf{f}$ where $\mathbf{f}$ is a force vector, and the moment of momentum (also called *angular momentum*) is $\mathbf{p} \times \mathbf{E}$ where $\mathbf{E}$ is a linear momentum vector.

Remark 2: Moments in mechanics and statistics
In statistics and probability theory, moments are used to characterize probability dis-

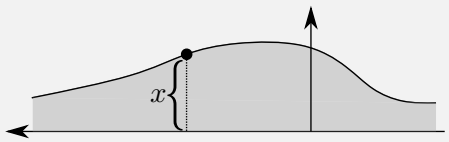
tributions. The first moment of a probability distribution is the mean, and the second moment is the variance. The terminology is no coincidence and is likely due to Chebyshev, a mathematician who was trained in mechanics before making several contributions to probability theory. In 1873, Chebyshev read a letter to the Congress of the French Association for the Advancement of the Sciences, in which he relates the integral $\int_A^B x^m f(x)dx$ to a problem in mechanics. The analogy between probability distributions and mass distributions became ingrained in the field, and the integral

$$\int_{-\infty}^{\infty} x^N f(x)dx \tag{2.21}$$

is now known as the Nth moment of the probability density function $f(x)$. It must be noted, however, that the definition of the moment in mechanics differs from the one used in statistics. Indeed, while x in (2.21) is the distance from the origin *parallel* to an axis, the moment of a physical vector is a function of the *perpendicular* distance between a point and the vector.



The distance x in statistics



The distance x in mechanics

The Nth moment of a body's mass density $\rho({}_b\mathbf{p}_i)$ is defined as

$$\int_V [{}_b\mathbf{p}_i]_\times^N \; \rho({}_b\mathbf{p}_i)\, dV, \tag{2.22}$$

where ${}_b\mathbf{p}_i$ is the position of the i-th point in the volume V of the body relative to a frame $\boldsymbol{\mathcal{F}}_b$ fixed on the body. The mass density function $\rho : \mathbb{R}^3 \to \mathbb{R}_+$ maps a point in V to a nonnegative mass density. The zeroth moment is therefore given by

$$\int_V [{}_b\mathbf{p}_i]_\times^0 \; \rho({}_b\mathbf{p}_i)dV = \mathbf{1}_3 \underbrace{\int_V \rho({}_b\mathbf{p}_i)dV}_{m} = m\mathbf{1}_3, \tag{2.23}$$

where m is the total mass of the body. The first moment is given by

$$\int_V [{}_b\mathbf{p}_i]_\times^1 \; \rho({}_b\mathbf{p}_i)\, dV = \underbrace{\left[\int_V {}_b\mathbf{p}_i \; \rho({}_b\mathbf{p}_i)\, dV\right]}_{m\,{}_b\mathbf{p}_c}{}_\times = m\,[{}_b\mathbf{p}_c]_\times\,, \tag{2.24}$$

where the identity $[\mathbf{u}]_\times + [\mathbf{v}]_\times = [\mathbf{u}+\mathbf{v}]_\times$ is used, and where ${}_b\mathbf{p}_c = \mathbf{c}$ is the centre of mass (CoM) of the body. The CoM is the point about which zero torque is exerted when the mass

of the body is uniformly accelerated. Mathematically, with the origin of $\boldsymbol{\mathcal{F}}_b$ fixed in space,

$$\int_V {}_c\mathbf{p}_i \; \rho({}_c\mathbf{p}_i)\, dV \times \mathbf{a} = \mathbf{0}, \tag{2.25}$$

which implies that

$${}_b\mathbf{p}_c \times m\mathbf{a} = \mathbf{0} \tag{2.26}$$

as the mass cannot be zero. In other words, the motion of the body under uniform acceleration can be equivalently described by a single point positioned at ${}_b\mathbf{p}_c$ whose mass is m. For bodies of homogeneous mass density, the location of the CoM is given by the geometrical centre, or *centroid*, of the body. The second moment of the body's mass distribution is given by

$$\int_V [{}_b\mathbf{p}_i]_\times^2 \;\; \rho({}_b\mathbf{p}_i)\, dV = \int_V [{}_b\mathbf{p}_i]_\times^\mathsf{T} [{}_b\mathbf{p}_i]_\times \;\; \rho({}_b\mathbf{p}_i)\, dV = {}_b\mathbf{I}_b, \tag{2.27}$$

which is the *inertia matrix* of the body computed relative to $\boldsymbol{\mathcal{F}}_b$. For a point-mass with position $\mathbf{p} = \begin{bmatrix}\mathbf{p}_x & \mathbf{p}_y & \mathbf{p}_z\end{bmatrix}^\mathsf{T}$ and mass m, the inertia matrix is given by

$$\mathbf{I} = m\,[\mathbf{p}]_\times^2 = m \begin{bmatrix} \mathbf{p}_y^2 + \mathbf{p}_z^2 & -\mathbf{p}_x\mathbf{p}_y & -\mathbf{p}_x\mathbf{p}_z \\ -\mathbf{p}_x\mathbf{p}_y & \mathbf{p}_x^2 + \mathbf{p}_z^2 & -\mathbf{p}_y\mathbf{p}_z \\ -\mathbf{p}_x\mathbf{p}_z & -\mathbf{p}_y\mathbf{p}_z & \mathbf{p}_x^2 + \mathbf{p}_y^2 \end{bmatrix}, \tag{2.28}$$

which clearly expose the symmetry of the inertia matrix. We can also notice that for any vector $\mathbf{p} \neq \mathbf{0}$, the diagonal elements of the inertia matrix are always positive. More subtle is the fact that the diagonal elements enforce a triangle inequality with

$$\mathbf{I}_{xx} \leq \mathbf{I}_{yy} + \mathbf{I}_{zz} \quad \text{or} \quad \mathbf{p}_y^2 + \mathbf{p}_z^2 \leq (\mathbf{p}_x^2 + \mathbf{p}_z^2) + (\mathbf{p}_x^2 + \mathbf{p}_y^2), \tag{2.29}$$

$$\mathbf{I}_{yy} \leq \mathbf{I}_{xx} + \mathbf{I}_{zz} \quad \text{or} \quad \mathbf{p}_x^2 + \mathbf{p}_z^2 \leq (\mathbf{p}_y^2 + \mathbf{p}_z^2) + (\mathbf{p}_x^2 + \mathbf{p}_y^2), \tag{2.30}$$

$$\mathbf{I}_{zz} \leq \mathbf{I}_{xx} + \mathbf{I}_{yy} \quad \text{or} \quad \mathbf{p}_x^2 + \mathbf{p}_y^2 \leq (\mathbf{p}_y^2 + \mathbf{p}_z^2) + (\mathbf{p}_x^2 + \mathbf{p}_z^2), \tag{2.31}$$

which, surprisingly, was only recently highlighted in robotics literature by Traversaro et al. (2016). The ij-th element of the inertia matrix, which can be extracted with

$$\mathbf{I}_{ij} = \hat{\mathbf{e}}_i^\mathsf{T} \mathbf{I} \hat{\mathbf{e}}_j, \tag{2.32}$$

expresses how torque applied about axis $\hat{\mathbf{e}}_i$ produces angular acceleration about axis $\hat{\mathbf{e}}_j$. The elements of $\mathbf{I}$ that are on the diagonal of the matrix are called the *moments of inertia* while the off-diagonal terms are commonly referred to as the *products of inertia*. While symmetries in the mass distribution will make some of the products of inertia be zero, the moments of inertia are always positive. Indeed, if moments of inertia were allowed to be zero, it would imply that acceleration about some axis required no torque, which would violate the law of energy conservation.

For a frame $\mathcal{F}_a$ whose origin is the same as $\mathcal{F}_b$ but whose axes are oriented with ${}_b\mathbf{R}_a$ relative to $\mathcal{F}_b$, we have that ${}^a_b\mathbf{p}_i = {}_a\mathbf{R}_b\, {}_b\mathbf{p}_i$ from (2.15). The inertia matrix computed relative to $\mathcal{F}_a$ is given by

$$
\begin{aligned}
{}^a_b\mathbf{I}_b = \int_V [{}^a_b\mathbf{p}_i]_\times^2 \ \rho({}_b\mathbf{p}_i) dV &= \int_V [{}_a\mathbf{R}_b\, {}_b\mathbf{p}_i]_\times^2 \ \rho({}_b\mathbf{p}_i) dV && (2.33)\\
&= \int_V [{}_a\mathbf{R}_b\, {}_b\mathbf{p}_i]_\times^\mathsf{T} [{}_a\mathbf{R}_b\, {}_b\mathbf{p}_i]_\times \ \rho({}_b\mathbf{p}_i) dV && (2.34)\\
&= \int_V {}_a\mathbf{R}_b\, [{}_b\mathbf{p}_i]_\times^\mathsf{T} \underbrace{{}_a\mathbf{R}_b^\mathsf{T}\, {}_a\mathbf{R}_b}_{=\mathbf{1}} [{}_b\mathbf{p}_i]_\times\, {}_a\mathbf{R}_b^\mathsf{T} \ \rho({}_b\mathbf{p}_i) dV && (2.35)\\
&= \int_V {}_a\mathbf{R}_b\, [{}_b\mathbf{p}_i]_\times^2\, {}_a\mathbf{R}_b^\mathsf{T} \ \rho({}_b\mathbf{p}_i) dV && (2.36)\\
&= {}_a\mathbf{R}_b \underbrace{\int_V [{}_b\mathbf{p}_i]_\times^2 \ \rho({}_b\mathbf{p}_i) dV}_{{}_b\mathbf{I}_b}\, {}_a\mathbf{R}_b^\mathsf{T} && (2.37)\\
&= {}_a\mathbf{R}_b\, {}_b\mathbf{I}_b\, {}_a\mathbf{R}_b^\mathsf{T}, && (2.38)
\end{aligned}
$$

where (2.35) makes use of the identity $[\mathbf{R}\mathbf{u}]_\times = \mathbf{R}\,[\mathbf{u}]_\times\,\mathbf{R}^\mathsf{T}$. The relation in (2.38) defines how the inertia matrix changes when the coordinate system used to express point positions is changed.

Given a frame $\mathcal{F}_c$ whose origin is located at the CoM and whose axes are aligned with the body frame $\mathcal{F}_b$, the inertia matrix computed relative to $\mathcal{F}_b$ is expressed as

$$
\begin{aligned}
{}_b\mathbf{I}_b &= \int_V [{}_b\mathbf{p}_i]_\times^2 \ \rho({}_b\mathbf{p}_i) dV && (2.39)\\
&= \int_V [{}_b\mathbf{p}_c + {}_c\mathbf{p}_i]_\times^2 \ \rho({}_c\mathbf{p}_i) dV && (2.40)\\
&= \int_V [{}_b\mathbf{p}_c]_\times^2 \ \rho({}_c\mathbf{p}_i) dV + \int_V [{}_c\mathbf{p}_i]_\times^2 \ \rho({}_c\mathbf{p}_i) dV && (2.41)\\
&\quad - [{}_b\mathbf{p}_c]_\times \underbrace{\int_V [{}_c\mathbf{p}_i]_\times \ \rho({}_c\mathbf{p}_i) dV}_{=\mathbf{0}} - \underbrace{\int_V [{}_c\mathbf{p}_i]_\times \ \rho({}_c\mathbf{p}_i) dV}_{=\mathbf{0}} [{}_b\mathbf{p}_c]_\times && (2.42)\\
&= [{}_b\mathbf{p}_c]_\times^2 \underbrace{\int_V \rho({}_c\mathbf{p}_i) dV}_{m} + \underbrace{\int_V [{}_c\mathbf{p}_i]_\times^2 \ \rho({}_c\mathbf{p}_i) dV}_{{}_c\mathbf{I}_b} && (2.43)\\
&= m\,[{}_b\mathbf{p}_c]_\times^2 + {}_c\mathbf{I}_b, && (2.44)
\end{aligned}
$$

where ${}_c\mathbf{I}_b$ is the inertia matrix of the body computed relative to the CoM. The result from (2.25) was used in (2.42) and the relation between ${}_b\mathbf{I}_b$ and ${}_c\mathbf{I}_b$ in (2.44) is known as *Steiner's theorem* or as the *parallel axis theorem*. The result from (2.44) essentially states that the moments of inertia are smallest when they are computed relative to the CoM. The same conclusion could have been reached by noticing that the sum of squared distances in (2.27)

is minimized when distances are measured relative to the mean position, which is the CoM. Consequently, a body is easier to accelerate about its CoM than about any other point.

Since the inertia matrix is symmetric (as shown in (2.28)), there exists a rotation matrix ${}_p\mathbf{R}_c$ that diagonalizes ${}_c\mathbf{I}_b$ using (2.38) with

$$ {}^p_c\mathbf{I}_b = {}_p\mathbf{R}_c \, {}_c\mathbf{I}_b \, {}_p\mathbf{R}_c^\mathsf{T} = \begin{bmatrix} \mathbf{I}_{px} & 0 & 0 \\ 0 & \mathbf{I}_{py} & 0 \\ 0 & 0 & \mathbf{I}_{pz} \end{bmatrix}, \tag{2.45} $$

where $\mathbf{I}_{px}$, $\mathbf{I}_{py}$, and $\mathbf{I}_{pz}$ are the *principal moments of inertia*. The rotation ${}_p\mathbf{R}_c$ can be obtained through an eigendecomposition of ${}_c\mathbf{I}_b$, with the columns of ${}_p\mathbf{R}_c$ being given by the eigenvectors of ${}_c\mathbf{I}_b$. The fact that, with the right choice of reference frame, the inertia matrix of any body can collapse to a set of three real numbers is quite remarkable and has important ramifications in dynamics. Perhaps most importantly, the free motion of any rigid body about its CoM is strictly determined by the principal moments of inertia.

The mass, centre of mass, and moments of inertia represent the *inertial parameters* of a body that can be regrouped into the 10×1 vector

$$ \boldsymbol{\phi} = \begin{bmatrix} m & \mathbf{c}_x & \mathbf{c}_y & \mathbf{c}_z & \mathbf{I}_{xx} & \mathbf{I}_{xy} & \mathbf{I}_{xz} & \mathbf{I}_{yy} & \mathbf{I}_{yz} & \mathbf{I}_{zz} \end{bmatrix}^\mathsf{T}, \tag{2.46} $$

or into the 6×6 *spatial inertia matrix*

$$ \boldsymbol{\mathcal{I}} = \begin{bmatrix} m\mathbf{1}_3 & -m\,[\mathbf{c}]_\times \\ m\,[\mathbf{c}]_\times & \mathbf{I} \end{bmatrix}. \tag{2.47} $$

However, not all 6×6 matrices are valid spatial inertia matrices and not all 10×1 vectors are valid inertial parameter vectors. Indeed, due to the mass density function $\rho(\mathbf{p})$ being strictly positive and defined only over the volume of the body, only a subset of all possible sets of inertial parameters are *physically realizable*. Inertial parameters that are physically realizable for a given volume are said to be *physically consistent* (Traversaro et al., 2016).

Definition 1 (Physical consistency)**.** *A set of inertial parameters $\boldsymbol{\phi}$ is physically consistent for a volume $V \subset \mathbb{R}^3$ if there exists a mass density function*

$$ \rho(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}_+ \quad \forall \quad \mathbf{p} \in V \tag{2.48} $$

that produces elements of $\boldsymbol{\phi}$ when Eqs. (2.23), (2.24) and (2.27) are evaluated.

It follows from Definition 1 and from (2.23) that a physically consistent mass is strictly positive. Furthermore, since (2.24) is a density-weighted integral of positions, the CoM must be a point in the convex hull of the body's volume.

Definition 2 (Convex hull)**.** *The convex hull of a volume V is the smallest convex volume that contains all points in V.*

The convex hull computation over a volume discretised into N points can be performed with the QuickHull algorithm (Barber et al., 1996) whose worst-case complexity is $O(N^2)$.

Definition 3 (Convex volume)**.** *A volume V is convex if for any two points $\mathbf{p}_1$ and $\mathbf{p}_2$ in V, the line segment between $\mathbf{p}_1$ and $\mathbf{p}_2$ is entirely contained in V.*

Additionally, as highlighted in (2.28), the inertia matrix must be symmetric, positive definite, and with diagonal elements satisfying the triangle inequalities in Eqs. (2.29) and (2.31). In summary, for a set of inertial parameters to be physically consistent for a given body, the following conditions must be satisfied:

- The mass must be strictly positive.
- The CoM must be in the convex hull of the body's volume.
- The inertia matrix must be symmetric, positive definite, and moments of inertia must satisfy the triangle inequality.

2.3.3 Motions

Velocities

In general, all points in the volume of a body moving in space (like the hammer pictured in Figure 2.6) will have different velocities relative to a fixed frame $\boldsymbol{\mathcal{F}}_w$ due to the orientation of the body changing over time. For a body whose volume is $V \subset \mathbb{R}^3$, the position of any point in the body relative to $\boldsymbol{\mathcal{F}}_w$ is given by

$$ {}_w\mathbf{p}_i = {}_w\mathbf{p}_b + {}_w\mathbf{R}_b \, {}_b\mathbf{p}_i \qquad \forall i \in V, \tag{2.49} $$

where ${}_w\mathbf{p}_b$ is the position of the body's origin relative to $\boldsymbol{\mathcal{F}}_w$ and ${}_w\mathbf{R}_b$ is the rotation matrix describing the orientation of the body relative to $\boldsymbol{\mathcal{F}}_w$. The velocity of the i-th point relative to $\boldsymbol{\mathcal{F}}_w$ is given by the time derivative of ${}_w\mathbf{p}_i$ with

$$ {}_w\dot{\mathbf{p}}_i = \frac{d}{dt}\left({}_w\mathbf{p}_b\right) + \frac{d}{dt}\left({}_w\mathbf{R}_b \, {}_b\mathbf{p}_i\right), \tag{2.50} $$

in which the rightmost term evaluates differently for all points in the body. Fortunately, with rigid objects, the velocity of any point in the body can be evaluated from a concise description of the body's velocity. This concise description takes the form of a set of two vectors: the linear velocity vector ${}_w\mathbf{v}_b$ and the angular velocity vector ${}_w\boldsymbol{\omega}_b$. The linear velocity describes the velocity of $\boldsymbol{\mathcal{F}}_b$'s origin relative to $\boldsymbol{\mathcal{F}}_w$ and is given by the time derivative of ${}_w\mathbf{p}_b$. The

angular velocity describes the axis about which $\boldsymbol{\mathcal{F}}_b$ is rotating relative to $\boldsymbol{\mathcal{F}}_w$ and the rate of rotation. From the linear and angular velocities, the velocity of any point in the body is given by

$$ {}_w\dot{\mathbf{p}}_i = {}_w\mathbf{v}_b + {}_w\mathbf{R}_b \, {}_b\mathbf{v}_i + {}_w\boldsymbol{\omega}_b \times {}_w\mathbf{R}_b \, {}_b\mathbf{p}_i \qquad \forall i \in V, \tag{2.51} $$

where $V \subset \mathbb{R}^3$ is the volume of the body and all vectors are expressed in $\boldsymbol{\mathcal{F}}_w$. Since (2.50) and (2.51) must be equal, we have that

$$ \underbrace{\frac{d}{dt}\left({}_w\mathbf{p}_b\right)}_{={}_w\mathbf{v}_b} + \frac{d}{dt}\left({}_w\mathbf{R}_b \, {}_b\mathbf{p}_i\right) = {}_w\mathbf{v}_b + {}_w\mathbf{R}_b \, {}_b\mathbf{v}_i + {}_w\boldsymbol{\omega}_b \times {}_w\mathbf{R}_b \, {}_b\mathbf{p}_i \tag{2.52} $$

$$ \frac{d}{dt}\left({}_w\mathbf{R}_b \, {}_b\mathbf{p}_i\right) = {}_w\mathbf{R}_b \frac{d}{dt}\left({}_b\mathbf{p}_i\right) + \frac{d}{dt}\left({}_w\mathbf{R}_b\right) {}_b\mathbf{p}_i \tag{2.53} $$

$$ = {}_w\mathbf{R}_b \, {}_b\mathbf{v}_i + {}_w\boldsymbol{\omega}_b \times {}_w\mathbf{R}_b \, {}_b\mathbf{p}_i, \tag{2.54} $$

which provides an expression for the time derivative of a rotated vector, an important result. By identification from the rightmost terms in (2.53) and (2.54), it follows that

$$ \frac{d}{dt}\left({}_w\mathbf{R}_b\right) = \left[{}_w\boldsymbol{\omega}_b\right]_\times {}_w\mathbf{R}_b, \tag{2.55} $$

which provides an expression for the time derivative of a rotation matrix, another important result. When we can assume that ${}_b\dot{\mathbf{p}}_i = \mathbf{0}$ due to rigidity, (2.54) simplifies to

$$ \frac{d}{dt}\left({}_w\mathbf{R}_b \, {}_b\mathbf{p}_i\right) = {}_w\boldsymbol{\omega}_b \times {}_w\mathbf{R}_b \, {}_b\mathbf{p}_i. \tag{2.56} $$

Accelerations

The rate of change of a point's velocity is given by the acceleration of the point. As we will see later, changing the velocity of a point-mass increases its kinetic energy. However, doing so requires the application of a force on the point-mass. Hence, the acceleration relates the force applied on a point-mass to its mass, making it a fundamental quantity in mechanics. Differentiating (2.51) with respect to time, we obtain

$$ \frac{d}{dt}\left({}_w\dot{\mathbf{p}}_i\right) = \frac{d}{dt}\left({}_w\mathbf{v}_b\right) + \frac{d}{dt}\left({}_w\mathbf{R}_b \, {}_b\dot{\mathbf{p}}_i\right) + \frac{d}{dt}\left({}_w\boldsymbol{\omega}_b \times {}_w\mathbf{R}_b \, {}_b\mathbf{p}_i\right) \tag{2.57} $$

$$ = \underbrace{\frac{d}{dt}\left({}_w\mathbf{v}_b\right)}_{={}_w\mathbf{a}_b} + \frac{d}{dt}\left({}_w\mathbf{R}_b \, {}_b\dot{\mathbf{p}}_i\right) + \underbrace{\frac{d}{dt}\left({}_w\boldsymbol{\omega}_b\right)}_{={}_w\boldsymbol{\alpha}_b} \times {}_w\mathbf{R}_b \, {}_b\mathbf{p}_i + {}_w\boldsymbol{\omega}_b \times \frac{d}{dt}\left({}_w\mathbf{R}_b \, {}_b\mathbf{p}_i\right), \tag{2.58} $$

where ${}_w\mathbf{a}_b$ and ${}_w\boldsymbol{\alpha}_b$ are respectively the linear and angular accelerations of the body relative to $\boldsymbol{\mathcal{F}}_w$. Using the result from (2.54) in (2.58), we get

$$\begin{aligned} {}_w\ddot{\mathbf{p}}_i &= {}_w\mathbf{a}_b + {}_w\mathbf{R}_b\, {}_b\mathbf{a}_i + {}_w\boldsymbol{\omega}_b \times {}_w\mathbf{R}_b\, {}_b\dot{\mathbf{p}}_i + {}_w\boldsymbol{\alpha}_b \times {}_w\mathbf{R}_b\, {}_b\mathbf{p}_i \\ &\quad + {}_w\boldsymbol{\omega}_b \times ({}_w\mathbf{R}_b\, {}_b\dot{\mathbf{p}}_i + {}_w\boldsymbol{\omega}_b \times {}_w\mathbf{R}_b\, {}_b\mathbf{p}_i) \qquad (2.59) \\ &= {}_w\mathbf{a}_b + {}^w_b\mathbf{a}_i + 2\,{}_w\boldsymbol{\omega}_b \times {}^w_b\dot{\mathbf{p}}_i + {}_w\boldsymbol{\omega}_b \times ({}_w\boldsymbol{\omega}_b \times {}^w_b\mathbf{p}_i) + {}_w\boldsymbol{\alpha}_b \times {}^w_b\mathbf{p}_i, \qquad (2.60) \end{aligned}$$

in which all vectors are expressed in $\boldsymbol{\mathcal{F}}_w$. While ${}_b\mathbf{a}_i$ is the acceleration of the i-th point as observed from the moving frame $\boldsymbol{\mathcal{F}}_b$, the left-hand side in (2.60) is the "true" acceleration of the point relative to the fixed frame $\boldsymbol{\mathcal{F}}_w$. The equation in (2.60) is so important that some terms have been given names:

$${}_w\ddot{\mathbf{p}}_i = {}^w_b\mathbf{a}_i + {}_w\mathbf{a}_b + \underbrace{2\,{}_w\boldsymbol{\omega}_b \times {}^w_b\dot{\mathbf{p}}_i}_{\text{Coriolis term}} + \underbrace{{}_w\boldsymbol{\omega}_b \times ({}_w\boldsymbol{\omega}_b \times {}^w_b\mathbf{p}_i)}_{\text{Centrifugal term}} + {}_w\boldsymbol{\alpha}_b \times {}^w_b\mathbf{p}_i. \qquad (2.61)$$

The Coriolis and centrifugal terms compensate for the discrepancy between the acceleration, as observed from the moving frame $\boldsymbol{\mathcal{F}}_b$, and the acceleration, as observed from the fixed frame $\boldsymbol{\mathcal{F}}_w$.

2.3.4 Inertial Frames

From (2.61), we concluded that the acceleration of a point as it is observed from $\boldsymbol{\mathcal{F}}_b$, a frame whose velocity is changing, is not the same as the acceleration of the point as observed from $\boldsymbol{\mathcal{F}}_w$. To determine the true acceleration of a point from the point of view of the moving frame $\boldsymbol{\mathcal{F}}_b$, additional terms had to be considered. A reference frame that can be considered to be fixed in space, like $\boldsymbol{\mathcal{F}}_w$, is an instance of an *inertial frame*. More generally, an inertial frame is considered to be a frame that does not accelerate.[6]

Definition 4 (Inertial frame)**.** *An inertial frame is a reference frame that does not accelerate.*

In an inertial frame, Newton's equations of motion directly apply without needing any additional terms. For most robotics applications on Earth, a frame fixed in the robot's workspace can be considered as an inertial frame for all practical purposes as the relative acceleration between two such frames is almost always negligible. In this thesis, a frame fixed in the robot workspace will be denoted $\boldsymbol{\mathcal{F}}_w$, called *world frame*, and will be assumed to be an inertial frame.

[6] Note that our definition of an inertial frame implies that any frame moving at a constant velocity relative to an inertial frame is also an inertial frame. The velocity of any one of these frames can be considered to be zero.

2.3.5 Forces and Torques

In Section 2.3.2, we defined the inertial parameters of a body as the mass, centre of mass, and moments of inertia. In Section 2.3.3, we derived expressions for the velocity and acceleration of points in a moving body relative to a fixed frame. The product of velocity and inertia yields two fundamental physical quantities: *momentum* and *kinetic energy*. The importance of these quantities comes from the fact that they are *conserved* in a closed system. The momentum of a point mass, originally termed "quantity of motion" by Isaac Newton, is given by

$$ {}_w\mathbf{E}_i = m_i \, {}_w\dot{\mathbf{p}}_i, \tag{2.62}$$

where ${}_w\mathbf{E}_i$ is the momentum of point-mass i when its velocity ${}_w\dot{\mathbf{p}}_i$ is observed in $\boldsymbol{\mathcal{F}}_w$. The kinetic energy of a point mass is given by integrating its momentum over the (relative) velocity with

$$ {}_wK_i = \int_0^{\dot{\mathbf{p}}} {}_w\mathbf{E}_i d\dot{\mathbf{p}} = \frac{1}{2} m_i \, {}_w\dot{\mathbf{p}}_i^{\mathsf{T}} \, {}_w\dot{\mathbf{p}}_i, \tag{2.63}$$

which is always positive. As the momentum and kinetic energy are proportional to the mass, the momentum and kinetic energy of multiple point masses is given by the sum of the individual contributions. Hence, the momentum of a rigid body $\{b\}$ with volume $V \subset \mathbb{R}^3$, mass density $\rho(\mathbf{p}_i)$, and centre of mass ${}_b\mathbf{p}_c$ is given by

$$\begin{aligned}
{}_w\mathbf{E}_b &= \int_V {}_w\dot{\mathbf{p}}_i \; \rho(\mathbf{p}_i) dV && (2.64)\\
&= \int_V ({}_w\dot{\mathbf{p}}_c + {}_c\dot{\mathbf{p}}_i) \; \rho(\mathbf{p}_i) dV && (2.65)\\
&= \int_V {}_w\dot{\mathbf{p}}_c \; \rho(\mathbf{p}_i) dV + \int_V {}_c\dot{\mathbf{p}}_i \; \rho(\mathbf{p}_i) dV && (2.66)\\
&= {}_w\dot{\mathbf{p}}_c \underbrace{\int_V \rho(\mathbf{p}_i) dV}_{=m} + \int_V {}_w\boldsymbol{\omega}_b \times {}_c\mathbf{p}_i \; \rho(\mathbf{p}_i) dV && (2.67)\\
&= m \, {}_w\dot{\mathbf{p}}_c + [{}_w\boldsymbol{\omega}_b]_\times \underbrace{\int_V {}_c\mathbf{p}_i \; \rho(\mathbf{p}_i) dV}_{=\mathbf{0}} && (2.68)\\
&= m \, {}_w\dot{\mathbf{p}}_c = m \left({}_w\mathbf{v}_b + {}_w\boldsymbol{\omega}_b \times {}_b\mathbf{p}_c + {}^w_b\dot{\mathbf{p}}_c \right), && (2.69)
\end{aligned}$$

where the integral in (2.68) equals zero by the definition of the CoM in (2.25). The expression in (2.69) represents the momentum of the body as if the total mass was concentrated at the CoM. By (2.69), the momentum of a spinning top would be zero if $\boldsymbol{\mathcal{F}}_w$ (the inertial frame) was fixed at the CoM. Hence, the expression in (2.69) is often referred to as the *linear momentum* of the body. Extending the idea of momentum to rotating motion, the moment

(computed relative to $\boldsymbol{\mathcal{F}}_b$) of the momentum (observed from $\boldsymbol{\mathcal{F}}_w$) is defined as

$$
\begin{aligned}
{}_b\boldsymbol{\varepsilon}_b &= \int_V {}_b\mathbf{p}_i \times {}_w\mathbf{E}_i \; \rho(\mathbf{p}_i) dV && (2.70)\\
&= \int_V {}_b\mathbf{p}_i \times {}^b_w\dot{\mathbf{p}}_i \; \rho(\mathbf{p}_i) dV && (2.71)\\
&= \int_V {}_b\mathbf{p}_i \times \left({}^b_w\mathbf{v}_b + {}^b_w\boldsymbol{\omega}_b \times {}_b\mathbf{p}_i \right) \; \rho(\mathbf{p}_i) dV && (2.72)\\
&= \int_V {}_b\mathbf{p}_i \times {}^b_w\mathbf{v}_b \; \rho(\mathbf{p}_i) dV + \int_V {}_b\mathbf{p}_i \times \left({}^b_w\boldsymbol{\omega}_b \times {}_b\mathbf{p}_i \right) \; \rho(\mathbf{p}_i) dV && (2.73)\\
&= \int_V {}_b\mathbf{p}_i \; \rho(\mathbf{p}_i) dV \times {}^b_w\mathbf{v}_b + \int_V -{}_b\mathbf{p}_i \times {}_b\mathbf{p}_i \times {}^b_w\boldsymbol{\omega}_b \; \rho(\mathbf{p}_i) dV && (2.74)\\
&= \underbrace{\int_V {}_b\mathbf{p}_i \; \rho(\mathbf{p}_i) dV}_{=m\,{}_b\mathbf{p}_c} \times {}^b_w\mathbf{v}_b + \underbrace{\int_V [{}_b\mathbf{p}_i]^{\mathsf{T}}_{\times} [{}_b\mathbf{p}_i]_{\times} \; \rho(\mathbf{p}_i) dV}_{={}_b\mathbf{I}_b} {}^b_w\boldsymbol{\omega}_b && (2.75)\\
&= m\,{}_b\mathbf{p}_c \times {}^b_w\mathbf{v}_b + {}_b\mathbf{I}_b \, {}^b_w\boldsymbol{\omega}_b, && (2.76)
\end{aligned}
$$

where the definition of the first and second mass moments in Eqs. (2.24) and (2.27) are used in (2.75). The expression on the left-hand side of (2.76) represents the moment of momentum, or *real* angular momentum of the body. The rightmost term in (2.76) is called the *intrinsic* angular momentum as it represents the angular momentum of the body relative to its CoM. Since the term "angular momentum" is sometimes used to refer to the real angular momentum and sometimes used to refer to the intrinsic angular momentum, we will refrain ourselves from using this ambiguous term. Instead, we will use the more specific terms "moment of momentum" and "intrinsic angular momentum".

Newton defined a *force* as "an action exerted upon a body in order to change its state, either of rest or of uniform motion in a right line" (Newton, 1962). Newton expanded by defining three axioms, or laws of motion, that were based on empirical observations. The first axiom introduces the concept of momentum conservation by stating that a body will maintain its linear velocity unless acted upon by a force. The second axiom establishes a proportionality relationship between the amount of force applied to a body and the change in momentum it experiences. The third axiom introduced the action-reaction principle between bodies, laying the foundation for the law of energy conservation in a closed system.

Together, the first two laws of motion can be summarized by the equation

$$
{}_b\mathbf{f}_b = \frac{d}{dt}\left(m\,{}_w\mathbf{E}_b\right) = m\,{}_w\ddot{\mathbf{p}}_c, \tag{2.77}
$$

where ${}_b\mathbf{f}_b$ is the force acting on the body as observed from $\boldsymbol{\mathcal{F}}_b$, the frame attached to the body, and ${}_w\ddot{\mathbf{p}}_c$ is the acceleration of the body's CoM as observed from $\boldsymbol{\mathcal{F}}_w$. From (2.61), the acceleration of the CoM relative to the inertial frame is given by

$$
{}_w\ddot{\mathbf{p}}_c = {}^w_b\mathbf{a}_c + {}_w\mathbf{a}_b + 2\,{}_w\boldsymbol{\omega}_b \times {}^w_b\mathbf{v}_c + {}_w\boldsymbol{\omega}_b \times ({}_w\boldsymbol{\omega}_b \times {}^w_b\mathbf{p}_c) + {}_w\boldsymbol{\alpha}_b \times {}^w_b\mathbf{p}_c. \tag{2.78}
$$

Substituting (2.78) into (2.77) yields

$$ {}^{w}_{b}\mathbf{f}_b = m\left({}^{w}_{b}\mathbf{a}_c + {}_{w}\mathbf{a}_b + 2\,{}_{w}\boldsymbol{\omega}_b \times {}^{w}_{b}\mathbf{v}_c + {}_{w}\boldsymbol{\omega}_b \times ({}_{w}\boldsymbol{\omega}_b \times {}^{w}_{b}\mathbf{p}_c) + {}_{w}\boldsymbol{\alpha}_b \times {}^{w}_{b}\mathbf{p}_c\right), \tag{2.79} $$

which relates the force acting on a rotating body as observed from $\boldsymbol{\mathcal{F}}_b$ to its kinematics measured relative to an inertial frame.

Considering the action causing a change in linear momentum led to the definition of force. It is natural to consider the action causing a change in real angular momentum as well. By definition, a change in a body's real angular momentum is given by

$$ \frac{d}{dt}\left({}^{w}_{b}\boldsymbol{\varepsilon}_b\right) = \frac{d}{dt}\left(\int_V {}^{w}_{b}\mathbf{p}_i \times {}_{w}\dot{\mathbf{p}}_i \, \rho(\mathbf{p}_i) dV\right), \tag{2.80} $$

where the integration is done over the body's volume, and vectors are expressed in $\boldsymbol{\mathcal{F}}_w$. Expanding the right-hand side, we get

$$ \begin{aligned} \frac{d}{dt}\left({}^{w}_{b}\boldsymbol{\varepsilon}_b\right) &= \frac{d}{dt}\left(\int_V {}^{w}_{b}\mathbf{p}_i \times {}_{w}\dot{\mathbf{p}}_i \, \rho(\mathbf{p}_i) dV\right) && (2.81)\\ &= \int_V {}^{w}_{b}\dot{\mathbf{p}}_i \times {}_{w}\dot{\mathbf{p}}_i \, \rho(\mathbf{p}_i) dV + \int_V {}^{w}_{b}\mathbf{p}_i \times {}_{w}\ddot{\mathbf{p}}_i \, \rho(\mathbf{p}_i) dV && (2.82)\\ &= \int_V \underbrace{{}^{w}_{b}\dot{\mathbf{p}}_i \times ({}_{w}\dot{\mathbf{p}}_b + {}^{w}_{b}\dot{\mathbf{p}}_i)}_{={}^{w}_{b}\dot{\mathbf{p}}_i \times {}_{w}\dot{\mathbf{p}}_b} \rho(\mathbf{p}_i) dV + \int_V {}^{w}_{b}\mathbf{p}_i \times {}_{w}\ddot{\mathbf{p}}_i \, \rho(\mathbf{p}_i) dV && (2.83)\\ &= \underbrace{\int_V {}^{w}_{b}\dot{\mathbf{p}}_i \, \rho(\mathbf{p}_i) dV}_{=m\,{}^{w}_{b}\dot{\mathbf{p}}_c} \times {}_{w}\dot{\mathbf{p}}_b + \underbrace{\int_V {}^{w}_{b}\mathbf{p}_i \times {}_{w}\ddot{\mathbf{p}}_i \, \rho(\mathbf{p}_i) dV}_{={}^{w}_{b}\boldsymbol{\tau}_b} && (2.84)\\ &= m\,{}^{w}_{b}\dot{\mathbf{p}}_c \times {}_{w}\dot{\mathbf{p}}_b + {}^{w}_{b}\boldsymbol{\tau}_b, && (2.85) \end{aligned} $$

where the term ${}^{w}_{b}\boldsymbol{\tau}_b$ is identified as the total *moment of force* acting on the body as observed from $\boldsymbol{\mathcal{F}}_b$. A moment of force, or torque, is a physical quantity representing the action of a force exerted at some distance from an axis. Similar to how all moments are computed in mechanics, it is given by

$$ {}_{b}\boldsymbol{\tau}_i = {}_{b}\mathbf{p}_i \times {}_{w}\mathbf{f}_i, \tag{2.86} $$

where the position vector is relative to $\boldsymbol{\mathcal{F}}_b$ while the physical quantity is observed in the inertial frame. Starting from (2.85) and using our result for the real angular momentum in

(2.76), we get

$$m\,{}^{w}_{b}\dot{\mathbf{p}}_c \times {}_{w}\dot{\mathbf{p}}_b + {}^{w}_{b}\boldsymbol{\tau}_b = \frac{d}{dt}\left({}^{w}_{b}\boldsymbol{\varepsilon}_b\right) \tag{2.87}$$

$$= \frac{d}{dt}\left(m\,{}_{w}\mathbf{R}_b\,{}_{b}\mathbf{p}_c \times {}_{w}\mathbf{v}_b + {}_{w}\mathbf{R}_b\,{}_{b}\mathbf{I}_b\,{}_{w}\boldsymbol{\omega}_b\right) \tag{2.88}$$

$$= m\,{}_{w}\mathbf{R}_b\,{}_{b}\dot{\mathbf{p}}_c \times {}_{w}\dot{\mathbf{p}}_b + m\,{}_{w}\mathbf{R}_b\,{}_{b}\mathbf{p}_c \times {}_{w}\ddot{\mathbf{p}}_b + {}_{w}\mathbf{R}_b\,{}_{b}\mathbf{I}_b\,{}_{w}\boldsymbol{\alpha}_b$$
$$+ {}_{w}\boldsymbol{\omega}_b \times {}_{w}\mathbf{R}_b\,{}_{b}\mathbf{I}_b\,{}_{w}\boldsymbol{\omega}_b, \tag{2.89}$$

and by canceling the leftmost terms on each side, we obtain

$${}_{b}\boldsymbol{\tau}_b = m\,{}_{b}\mathbf{p}_c \times {}^{b}_{w}\mathbf{a}_b + {}_{b}\mathbf{I}_b\,{}^{b}_{w}\boldsymbol{\alpha}_b + {}^{b}_{w}\boldsymbol{\omega}_b \times {}_{b}\mathbf{I}_b\,{}^{b}_{w}\boldsymbol{\omega}_b. \tag{2.90}$$

The equation in (2.90) defines the effect of moments of force measured relative to $\boldsymbol{\mathcal{F}}_b$ on the body's kinematics as observed from $\boldsymbol{\mathcal{F}}_w$.

In general, forces acting on a body will cumulate to produce a net force as well as a net moment of force about the body's origin. The effect of the net force and torque on the body's kinematics is given by the *Newton-Euler equations*, a set of equations resulting from the pairing of (2.79) and (2.90):

$${}_{b}\mathbf{f}_b = m\left({}_{b}\mathbf{a}_c + {}^{b}_{w}\mathbf{a}_b + 2\,{}^{b}_{w}\boldsymbol{\omega}_b \times {}_{b}\mathbf{v}_c + {}^{b}_{w}\boldsymbol{\omega}_b \times \left({}^{b}_{w}\boldsymbol{\omega}_b \times {}_{b}\mathbf{p}_c\right) + {}^{b}_{w}\boldsymbol{\alpha}_b \times {}_{b}\mathbf{p}_c\right), \tag{2.91}$$

$${}_{b}\boldsymbol{\tau}_b = m\,{}_{b}\mathbf{p}_c \times {}^{b}_{w}\mathbf{a}_b + {}_{b}\mathbf{I}_b\,{}^{b}_{w}\boldsymbol{\alpha}_b + {}^{b}_{w}\boldsymbol{\omega}_b \times {}_{b}\mathbf{I}_b\,{}^{b}_{w}\boldsymbol{\omega}_b. \tag{2.92}$$

Expressing all quantities in the frame in which the inertial parameters are defined relative to is convenient to keep the inertial parameter values constant over time. The Newton-Euler equations can be formulated as a matrix equation with

$$\underbrace{\begin{bmatrix} {}_{b}\mathbf{f}_b \\ {}_{b}\boldsymbol{\tau}_b \end{bmatrix}}_{{}_{b}\mathbf{w}_b} = \underbrace{\begin{bmatrix} m\mathbf{1}_3 & -m\left[{}_{b}\mathbf{p}_c\right]_\times \\ m\left[{}_{b}\mathbf{p}_c\right]_\times & {}_{b}\mathbf{I}_b \end{bmatrix}}_{{}_{b}\boldsymbol{\mathcal{I}}_b} \underbrace{\begin{bmatrix} {}^{b}_{w}\mathbf{a}_b \\ {}^{b}_{w}\boldsymbol{\alpha}_b \end{bmatrix}}_{{}^{b}_{w}\boldsymbol{\lambda}_b} + \underbrace{\begin{bmatrix} m\left({}_{b}\mathbf{a}_c + 2\,{}^{b}_{w}\boldsymbol{\omega}_b \times {}_{b}\mathbf{v}_c - \left[{}^{b}_{w}\boldsymbol{\omega}_b\right]^2_\times {}_{b}\mathbf{p}_c\right) \\ \left[{}^{b}_{w}\boldsymbol{\omega}_b\right]_\times {}_{b}\mathbf{I}_b\,{}^{b}_{w}\boldsymbol{\omega}_b \end{bmatrix}}_{\text{fictitious forces}}, \tag{2.93}$$

where the three leftmost matrices are the wrench (or spatial force), the spatial inertia, and the spatial acceleration, respectively. The rightmost term in (2.93) contains the so-called *fictitious forces*, which are terms added to account for the skewed perception of kinematics in $\boldsymbol{\mathcal{F}}_b$. In many robotics applications, it is possible to rigidly attach $\boldsymbol{\mathcal{F}}_b$ to the CoM of the body such that ${}_{b}\mathbf{v}_c = \mathbf{0}$ and ${}_{b}\mathbf{a}_c = \mathbf{0}$. Doing so simplifies (2.93) to

$${}_{b}\mathbf{w}_b = {}_{b}\boldsymbol{\mathcal{I}}_b\,{}^{b}_{w}\boldsymbol{\lambda}_b + \begin{bmatrix} -\left[{}^{b}_{w}\boldsymbol{\omega}_b\right]^2_\times {}_{b}\mathbf{p}_c \\ \left[{}^{b}_{w}\boldsymbol{\omega}_b\right]_\times {}_{b}\mathbf{I}_b\,{}^{b}_{w}\boldsymbol{\omega}_b \end{bmatrix}, \tag{2.94}$$

which is further simplified into ${}_b\mathbf{w}_b = {}_b\boldsymbol{\mathcal{I}}_b \, {}^b_w\boldsymbol{\lambda}_b$ when $\boldsymbol{\mathcal{F}}_b = \boldsymbol{\mathcal{F}}_w$, a case rarely encountered in practical robotics applications.

A key property of (2.93) is that it is linear in the inertial parameters of the body. From (2.46), the inertial parameters of a body can be regrouped into

$$\boldsymbol{\phi} = \begin{bmatrix} m & \mathbf{c}_x & \mathbf{c}_y & \mathbf{c}_z & \mathbf{I}_{xx} & \mathbf{I}_{xy} & \mathbf{I}_{xz} & \mathbf{I}_{yy} & \mathbf{I}_{yz} & \mathbf{I}_{zz} \end{bmatrix}^\mathsf{T}, \tag{2.95}$$

a 10×1 vector containing the zeroth, first, and second moments of body mass. The equation in (2.93) can be rewritten as

$${}_b\mathbf{w}_b = \mathbf{A}\boldsymbol{\phi}, \tag{2.96}$$

with

$$\begin{aligned} \mathbf{A} = &\begin{bmatrix} {}_w\mathbf{a}_b & \mathbf{0}_{3\times 9} \\ \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 9} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times 1} & [{}_w\boldsymbol{\omega}_b]_\times & [{}_w\boldsymbol{\omega}_b]_\times + [{}_w\boldsymbol{\alpha}_b]_\times & \mathbf{0}_{3\times 6} \\ \mathbf{0}_{3\times 1} & & -[{}_w\mathbf{a}_b]_\times & \mathbf{0}_{3\times 6} \end{bmatrix} \\ &+ \begin{bmatrix} & & & \mathbf{0}_{3\times 10} & & \\ \mathbf{0}_{1\times 4} & \alpha_x & \alpha_y - \omega_x\omega_z & \alpha_z + \omega_x\omega_y & -\omega_y\omega_z & \omega_y^2 - \omega_z^2 & \omega_y\omega_z \\ \mathbf{0}_{1\times 4} & \omega_x\omega_z & \alpha_x + \omega_y\omega_z & \omega_z^2 - \omega_x^2 & \alpha_y & \alpha_z - \omega_x\omega_y & -\omega_x\omega_z \\ \mathbf{0}_{1\times 4} & -\omega_x\omega_y & \omega_x^2 - \omega_y^2 & \alpha_x - \omega_y\omega_z & \omega_x\omega_y & \alpha_y + \omega_x\omega_z & \alpha_z \end{bmatrix}, \end{aligned} \tag{2.97}$$

where ${}^b_w\boldsymbol{\alpha}_b = \begin{bmatrix} \alpha_x & \alpha_y & \alpha_z \end{bmatrix}^\mathsf{T}$ and ${}^b_w\boldsymbol{\omega}_b = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^\mathsf{T}$. The matrices in (2.97), from left to right, are linear in the zeroth, first, and second moments of mass, respectively. The matrix $\mathbf{A}$ is called the "data matrix" or "regressor" in the context of inertial parameter identification.

Chapter 3

Inertial Parameter Identification For Collaborative Robots

> Inferring models from observations and studying their properties is really what science is about.
>
> Lennart Ljung, 1999.

As autonomous mobile robot (AMR) strive to perform object handling tasks with greater autonomy, predicting the dynamics of their interactions with the environment is crucial for effective and safe operation. In particular, identifying the inertial parameters of manipulated objects (i.e., their mass, centre of mass (CoM), and moments of inertia) is essential for reliable autonomous handling operations (Mason and Lynch, 1993). For instance, this information has been used to plan efficient motion trajectory (Mavrakis et al., 2016), for planning grasps in cluttered environments (Dogar et al., 2012), and has enabled impressive performance in pick-and-throw tasks (Zeng et al., 2019). Humans instinctively infer these parameters through haptic perception, allowing them to predict an object's behaviour and plan grasps accordingly (Lukos et al., 2007; Hamrick et al., 2016; Pagano and Turvey, 1992). We posit that, in order for robots to be effective at manipulation tasks, they should be given similar perceptual capacities.

Force-torque sensors (FT sensors) located in the joints or at the wrist of a robot arm can be used to determine the inertial parameters of a manipulated object by relating the motion of the object to the measured forces and torques. However, sensor noise can greatly limit parameter identification accuracy, especially for collaborative robots (cobots) that must move within safe velocity limits. Specifically, cobots' motion constraints lead to lower signal-to-noise ratios (SNRs) in velocity, acceleration, and force-torque data. In contrast, traditional methods for estimating the full set of inertial parameters rely on motions that are fast (to achieve a sufficient SNR), and consequently unsafe to perform around human

workers.

In this chapter, we introduce two approaches to improve the speed and accuracy of inertial parameter identification for cobots. The first approach, discussed in Section 3.1, addresses the main sources of noise during the identification procedure by approximating rigid body dynamics for slower motions and transitioning to the complete dynamics model for motions with higher SNR. The second approach, presented in Section 3.2, combines visual and force-torque measurements to enable inertial parameter identification using slow or "stop-and-go" motions, making it suitable for use around humans. Both approaches ensure physical consistency of the inertial parameters by utilizing knowledge about the object shape and discretizing the mass distribution. We demonstrate the effectiveness of our methods through extensive simulation studies and demonstrate their real-world performance on a low-cost cobot arm.

3.1 A Method Based on Mixed Dynamics Models

In this section, we analyse the dynamics of safe cobot operation, which typically hampers identification algorithms due to the low SNRs of the velocity, acceleration, and force-torque data. Our analysis motivates the use of an approximation of rigid body dynamics aimed at increasing the SNR and improving the accuracy and convergence rate of parameter identification. The proposed approximation can easily be integrated into a variety of existing

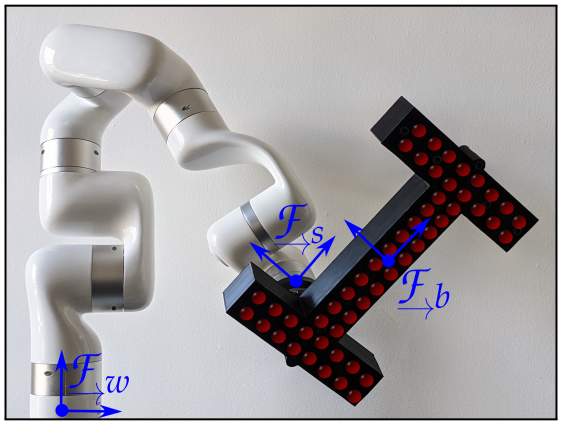


Figure 3.1: In our experiments, a test object (black) is attached to a robot arm and the inertial parameters of the test object are identified. The shape of the object is discretized and point masses (red spheres) are assigned at positions that are fixed relative to the body frame $\boldsymbol{\mathcal{F}}_b$. The object moves in the inertial frame $\boldsymbol{\mathcal{F}}_w$, while a sensor in frame $\boldsymbol{\mathcal{F}}_s$ measures forces.

identification algorithms. However, by using this approximation alone, we show that the identifiability of the full inertia tensor is lost.

Based on our analysis, we define a method to quickly identify all inertial parameters by combining the approximate dynamics model with the full model through a weighting

scheme.[1] We base our weighting scheme on a heuristic measure of the 'magnitude' of the cobot's motion. The proposed approach is based on a point-mass discretization that leverages knowledge of the shape of the manipulated object; shape information could be captured through the cobot's vision system or from another source. For instance, in Section 3.2, we make use of robot vision prior to the identification procedure to capture shape information and feed it to a part segmentation algorithm. Our method identifies physically consistent values of the inertial parameters and has only a single scalar hyperparameter that can be tuned on a per-robot basis.

3.1.1 Prior Work in Load Identification

Unlike quasi-static or kinematic manipulation tasks (Mason, 2018), dynamic manipulation requires explicit consideration of the forces applied to the manipulated object (Mason and Lynch, 1993). In this section, we briefly survey inertial parameter identification methods that are most relevant to collaborative robotic manipulation. For a recent review of inertial parameter identification for robotic systems and for manipulated objects, we direct our readers to (Golluccio et al., 2020) and (Mavrakis and Stolkin, 2020), respectively.

In (Atkeson et al., 1986), a least squares method is applied to determine the inertial parameters of a load manipulated by a robot arm. Atkeson et al. (1986) underline that poor performance can be caused by low SNRs in the force-torque measurements and acceleration signals, but that least squares methods cannot account for errors in the regressor matrix (defined in (2.97)). To address this issue, Kubus et al. (2008) propose to mitigate the effects of noisy or inaccurate kinematic data by using a recursive total least squares (RTLS) formulation, which can account for errors in the regressor matrix as well. Farsoni et al. (2018) benchmark the RTLS method on real robotic manipulation tasks, comparing its performance on acceleration data from multiple different sources. Farsoni et al. (2018) report large estimation errors and a slow convergence time, suggesting that the performance of RTLS is limited in noisy, real-world contexts. While our proposed method also minimizes a least squares cost, we employ a unique formulation that is well-suited to data gathered by a slower-moving cobot.

Sousa and Cortesao (2014) make the critical observation that inertial parameter identification has traditionally been ill-posed, and that many algorithms converge toward inertial parameters that are unphysical. Traversaro et al. (2016) state *sufficient* conditions for physical consistency and formulates a manifold optimization problem constraining the feasible set to physically-consistent solutions. In (Wensing et al., 2017), a set of linear matrix inequalities (LMIs) are used to enforce physical consistency without relying on the evaluation of computationally expensive nonlinear functions. Geodesic distance approximations from a prior solution are used in (Lee et al., 2019) to regularize the optimization problem with-

[1] We provide code, models of our test objects, a video showcasing our algorithm, and supplementary material at https://papers.starslab.ca/fast-inertial-identification/.

out introducing very long runtimes. In contrast, Ayusawa and Nakamura (2010) *implicitly* enforce physical consistency by segmenting the load with points and estimating the mass of each point instead of directly identifying the inertial parameters. Song and Boularias (2020) similarly discretize an object using fixed-size voxels and perform quasi-static exploratory planar manipulation. Ayusawa and Nakamura (2010) demonstrate that full physical consistency is guaranteed if the point masses are all located in the convex hull of the body, providing a strong theoretical foundation for the approach. While we also apply point mass discretization, our work does not assume prior knowledge of the inertial parameters or accurate knowledge of the dynamics, and instead uses information from visual or other perception sources.

3.1.2 Background

As mentioned in Section 2.3.2, the inertial parameters of a rigid body can be grouped as

$$\boldsymbol{\phi} = \begin{bmatrix} m & \mathbf{c}_x & \mathbf{c}_y & \mathbf{c}_z & \mathbf{I}_{xx} & \mathbf{I}_{xy} & \mathbf{I}_{xz} & \mathbf{I}_{yy} & \mathbf{I}_{yz} & \mathbf{I}_{zz} \end{bmatrix}^\mathsf{T}, \tag{3.1}$$

where $m \in \mathbb{R}_+$ is the mass, $\mathbf{c} = [c_x, c_y, c_z]^\mathsf{T} \in \mathbb{R}^3$ is the position of the CoM, and $\mathbf{I} \in \mathcal{S}^3_{++}$ (i.e., the set of 3×3 symmetric positive definite matrices) is the inertia tensor. Inertial parameters are related to the motion of a rigid body through the Newton-Euler equations that we derived in Section 2.3.5. Since the forces $\mathbf{f}$ and torques $\boldsymbol{\tau}$ measured by the robot are linear in the inertial parameters, they can be related with

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = \mathbf{A}(\mathbf{a}, \boldsymbol{\omega}, \boldsymbol{\alpha})\boldsymbol{\phi} \tag{3.2}$$

where $\mathbf{A}(\mathbf{a}, \boldsymbol{\omega}, \boldsymbol{\alpha}) \in \mathbb{R}^{6\times 10}$ is the "regressor" matrix defined in (2.97) that depends on the linear acceleration $\mathbf{a}$, angular acceleration $\boldsymbol{\alpha}$ and angular velocity $\boldsymbol{\omega}$. By observing $\mathbf{f}$, $\boldsymbol{\tau}$, $\mathbf{a}$, $\boldsymbol{\omega}$, and $\boldsymbol{\alpha}$, a least squares algorithm can be used to infer values for the vector of inertial parameters $\boldsymbol{\phi}$, even when noise is present in the measured forces and torques (Atkeson et al., 1986). However, according to Kroger et al. (2008), the following problems arise in practice:

- the signals from end-effector force-torque sensors are very noisy, with an unknown bias that drifts over time; and
- the amplitudes of the acceleration and velocity signals are small, resulting in low SNRs.

Additionally, velocity and acceleration signals can be very noisy if they are obtained through time-differentiation of robot joint positions. As shown in (Farsoni et al., 2018), equipping the robot end-effector with accelerometers does not necessarily yield better measurements, since these sensors can also be inaccurate, especially if they are not properly calibrated.

The SNR of a signal is defined as the ratio of signal power to noise power — it is unitless. The SNR can be computed as

$$\mathrm{SNR} = \frac{S^2}{N^2}, \tag{3.3}$$

where S and N are the root-mean-squared amplitudes of the signal and noise, respectively. In the context of load identification, we define the cumulative sum of SNRs as the *dynamism* ν of the motion with

$$\begin{aligned} \nu &= \mathrm{SNR}_{\mathbf{a}} + \mathrm{SNR}_{\boldsymbol{\alpha}} + \mathrm{SNR}_{\boldsymbol{\omega}} && (3.4)\\ &= \left(\frac{\|\mathbf{a}\|}{n_1}\right)^2 + \left(\frac{\|\boldsymbol{\alpha}\|}{n_2}\right)^2 + \left(\frac{\|\boldsymbol{\omega}\|}{n_3}\right)^2 && (3.5)\\ &= \left|\left|\left[\frac{\|\mathbf{a}\|}{n_1}, \frac{\|\boldsymbol{\alpha}\|}{n_2}, \frac{\|\boldsymbol{\omega}\|}{n_3}\right]\right|\right|_2^2 && (3.6) \end{aligned}$$

where n_1, n_2, and n_3 are the noise amplitudes of the respective signals. In this work, we do not assume access to prior knowledge about noise characteristics and set $n_1 = 1$, $n_2 = 1$, and $n_3 = 0.5$.

In the context of collaborative robotics, it is important to consider the typical velocities and accelerations of human manipulation tasks. We can assume that human workers would be comfortable working with a robot that moves at speeds similar to those they are accustomed to. To this end, we can leverage the extensive dataset of human manipulation during activities of daily living (ADL) provided by Saudabayev et al. (2018) to quantify the average velocities and accelerations at which objects are usually wielded. We present the resulting statistics in Table 3.1. The Daily Interactive Manipulation Dataset (Huang and Sun, 2019), provides similar average velocities, confirming the information in Saudabayev et al. (2018). The International Organization for Standardization (ISO) standards 10218 (ISO, 2011) and 15066 (ISO, 2016) define safety bounds for the operation of collaborative robots, including a maximum linear tool centre-point (TCP) speed of 0.25 m/s, which is slightly above the average linear velocity of 0.206 m/s observed in the ADL dataset. A noteworthy statistic extracted from the ADL dataset is the median duration of a manipulation task (2.3 seconds), which motivates the requirement for fast inertial parameter identification algorithms.

Gravity as a Dominating Force

Performing an analysis of objects from the Yale-CMU-Berkeley dataset (Calli et al., 2015), which categorizes 77 items typically used for ADL, allows us to estimate the average mass $\overline{m}$ of the manipulated objects and the average distance $\overline{r}$ from the CoM to the edge of the object. Excluding from the analysis any object that is flexible, has a mass less than 10 grams, or is clearly intended to be used with both hands, the remaining 63 objects have an average mass of $\overline{m} = 0.257$ kilograms and an $\overline{r} = 0.081$ metres.

Table 3.1: Statistics related to human manipulation during activities of daily living (ADL).

	Symbol	Mean	Median	Unit
Linear velocity	$\mathbf{v}$	0.206	0.128	m/s
Angular velocity	$\boldsymbol{\omega}$	1.08	0.703	rad/s
Linear acceleration	$\mathbf{a}$	1.45	1.20	$\mathrm{m/s^2}$
Angular acceleration	$\boldsymbol{\alpha}$	11.34	8.38	$\mathrm{rad/s^2}$
Object mass	m	0.257	0.110	kg
Object radius	r	0.081	0.068	m
Duration	d	6	2.3	s

Given the values in Table 3.1, we can compare the effect of gravity to the effect of other forces by computing the ratios of the magnitudes of non-gravitational forces and torques to the magnitudes of the total forces and torques applied during manipulation. These ratios indicate that gravity dominates over other forces by a factor of about five during ADL performed by human subjects. In turn, we can approximate the mass of the object with

$$m \approx \frac{||\mathbf{f}||_2}{||\mathbf{g}||_2} \tag{3.7}$$

where $\mathbf{g}$ is the gravitational acceleration. Similarly, we can approximate the measured torques with

$$\boldsymbol{\tau} \approx -m \left[\mathbf{c}\right]_\times \mathbf{g} \tag{3.8}$$

as if it was fully determined by the effect of gravity acting on the mass distribution of the object. In Section 3.1.3, we leverage these approximations to attenuate the impact of the noise associated with the manipulator velocity and acceleration signals.

Discretization of the Object Shape and Mass

The mass distribution of a rigid body can be approximated through the use of a large number of point masses, each with position ${}_b\mathbf{p}_i$ and mass m_i. The kth moment of the mass distribution, which we defined in (2.22), is thereby approximated as

$$\int_V \left[{}_b\mathbf{p}_i\right]_\times^k \rho({}_b\mathbf{p}_i) dV \approx \sum_i^n \left[{}_b\mathbf{p}_i\right]_\times^k m_i \tag{3.9}$$

where $\rho({}_b\mathbf{p}_i)$ is the density of the object at that point. Like force and torque, inertia is additive, such that the inertia of a body is the sum of the inertia of all its constituent point masses. By combining (3.8) with (3.9), the approximate measured torque becomes

$$\boldsymbol{\tau} \approx -\sum_i^n m_i \left[{}_b\mathbf{p}_i\right]_\times \mathbf{g} \ , \tag{3.10}$$

which we expect to be accurate at slow speeds that are representative of manipulation tasks in ADL.

3.1.3 The Point Mass Discretization Problem

In this section, we introduce our point mass discretization (PMD) formulation of inertial parameter estimation. Consider, as in Figure 3.1, a rigid body attached to reference frame $\boldsymbol{\mathcal{F}}_b$ and manipulated by a robot whose end-effector pose is described in the inertial reference frame $\boldsymbol{\mathcal{F}}_w$. This body is discretized into n point masses, each with constant position ${}_b\mathbf{p}_i \in \mathbb{R}^3$ relative to $\boldsymbol{\mathcal{F}}_b$, and constant mass $m_i > 0$. The position ${}_b\mathbf{p}_i$ is determined using a sampling algorithm that ensures every point mass is contained within the shape of the object. While manipulating the object, the robot uses a force-torque sensor attached to reference frame $\boldsymbol{\mathcal{F}}_s$ to take K measurements. According to our approximation in (3.10), at a particular time step j, the sum of the effects of gravity on each point mass should be equal to the jth torque measurement. Hence, the reduced model $\widetilde{\mathbf{A}}$ at timestep j is written in $\boldsymbol{\mathcal{F}}_w$ as:

$$ {}_w\boldsymbol{\tau}_{s_j} = -\begin{bmatrix} [{}_w\mathbf{p}_1]_\times \mathbf{g} & [{}_w\mathbf{p}_i]_\times \mathbf{g} & \cdots & [{}_w\mathbf{p}_n]_\times \mathbf{g} \end{bmatrix} \mathbf{m} \triangleq \widetilde{\mathbf{A}}_j \mathbf{m} \tag{3.11} $$

where

$$ {}_w\mathbf{p}_i = {}_w\mathbf{R}_{b_j}\, {}_b\mathbf{p}_i + {}_w\mathbf{p}_{b_j} \ , \tag{3.12} $$

and where $\mathbf{m} \in \mathbb{R}^n_+$ is a vector containing all masses. The orientation ${}_w\mathbf{R}_{b_j}$ and position ${}_w\mathbf{p}_{b_j}$ of $\boldsymbol{\mathcal{F}}_b$ for the j-th measurement are assumed to be known from the robot's forward kinematics. The optimization problem underlying PMD is defined as follows:

Problem 1 (Point Mass Discretization)**.**

$$ \begin{aligned} \min_{\mathbf{m}\in\mathbb{R}^n} \quad & ||(\mathbf{1}-\mathbf{w})^\mathsf{T}(\widetilde{\mathbf{A}}\mathbf{m}-\mathbf{b})||_2 + ||\mathbf{w}^\mathsf{T}(\mathbf{A}\mathbf{m}-\mathbf{b})||_2 + \lambda||\mathbf{m}||_2 \\ \text{s.t.} \quad & m_i \geq 0 \ \ \forall i \in \{1,\ldots,n\}. \end{aligned} $$

where $\widetilde{\mathbf{A}}$ and $\mathbf{A}$ are in $\mathbb{R}^{6K\times n}$ and are formed by stacking K measured reduced and full model submatrices respectively, which are in $\mathbb{R}^{6\times n}$. Wrench vector $\mathbf{b} \in \mathbb{R}^{6K}$ is formed by stacking K wrench measurements $[{}_w\boldsymbol{\tau}_{s_j}^\mathsf{T}, {}_w\mathbf{f}_{s_j}^\mathsf{T}]^\mathsf{T}$, $\mathbf{w} \in \mathbb{R}^{6K}$ is a weight vector formed by stacking K vectors $\mathbf{w}_j$ defined below, and $\lambda \in \mathbb{R}_+$ is a regularization factor encouraging homogeneity and ensuring that a unique solution exists.

At timestep k, the weight vector $\mathbf{w}_k$ is formed by stacking six identical weight values w that are computed according to

$$ w = \tanh\left(\frac{3\nu}{c_1}\right) \ , \tag{3.13} $$

where $\nu \in \mathbb{R}_+$ is the dynamism defined in (3.6). The tunable hyperparameter $c_1 \in \mathbb{R}_+$ in (3.13) defines the level of dynamism required for the full model to account for $\tanh(3) \approx$

Table 3.2: Standard deviations of zero-mean Gaussian distributions used to generate noise for simulation experiments. The specifications of the Robotiq FT-300 sensor were used to select these values.

	Ang. Accel.	Lin. Accel.	Force	Torque
Low Noise	0.25	0.025	0.05	0.0025
Moderate Noise	0.5	0.05	0.1	0.005
High Noise	1	0.1	0.33	0.0067

99.5% of the objective function. We designate c_1 as a *hyperparameter* to differentiate it from the parameters used by the solver. In this work, the regularization factor λ was set to 0.1 for all experiments.

Since the objective function of Problem 1 is the norm of an affine expression and the constraints are all linear, Problem 1 is a convex optimization problem and can therefore be efficiently solved with general-purpose methods (Boyd and Vandenberghe, 2004).

3.1.4 Experiments

In this section, we test PMD in a variety of experiments to highlight the impact of the c_1 hyperparameter and point mass density and on identification performance. We also compare the PMD algorithm to three benchmarks: the ordinary least squares (OLS) algorithm from (Atkeson et al., 1986), the RTLS algorithm from (Kubus et al., 2008), and the algorithm using the entropic distance metric defined in (Lee et al., 2019) (GEO). Data from simulations are leveraged to compare the performance of identification algorithms under varying levels of synthetic noise and at multiple TCP velocities. The specifications of the Robotiq FT-300 sensor, a common FT sensor which we used in our robot experiments, were used to select the standard deviations of the zero-mean Gaussian noise that was added to the signals as described in Table 3.2. In cases where an identification algorithm requires an initial guess, a homogeneous distribution is assumed. Finally, these algorithms are compared on data obtained through manipulation trajectories performed by a real robot arm on a variety of test objects. Our findings demonstrate that PMD outperforms benchmark algorithms in scenarios commonly encountered by collaborative robots.

Experimental Setup

We built a modular test object (shown in Figure 3.2) consisting of a 3D-printed poly lactic acid plastic (PLA) structure with holes allowing the placement of weights or bolts. Various configurations (shown in Figure 3.3) can be obtained by choosing the locations of the weights, and precise values of the inertial parameters can easily be computed for any configuration.

To ensure a fair comparison, algorithms are compared on data obtained with the same motion trajectory. Following (Kubus et al., 2008), the motion of the end-effector is deter-

mined by commanding the three robot joints proximal to the end-effector follow sinusoidal trajectories. Each trial used the same 35-second motion trajectory obtained by joints following

$$\alpha_n(t) = \alpha_n(0) + 45^\circ \sin\left(\frac{2\pi f_n t}{240}\right) \tag{3.14}$$

where $\alpha_n(0)$ is the n-th joint's initial position, f_n is $\begin{bmatrix}0 & 0 & 0 & 0.1 & 0.13 & 0.16\end{bmatrix}$ for joints 1 to 6 respectively, and t is the time elapsed in seconds.

Error Metrics

To compare the performance of our algorithm to the benchmarks, we propose error metrics that are scale-invariant with respect to the mass and size of the manipulated object and work for all physically plausible scenarios. While (Kubus et al., 2008; Farsoni et al., 2018) use the standard relative error to put the absolute estimation error in perspective, this metric is not meaningful when the reference value approaches zero, as is often the case with inertial parameters. The identification error can also be expressed as a distance on the manifold of inertial parameters (Traversaro et al., 2016; Lee et al., 2019), which is useful for comparing methods or training learning algorithms, but can be difficult to interpret in practice.

In this work, we propose error metrics (Eqs. (3.15) to (3.17)) that are: (i) scale-invariant with respect to the mass and size of the manipulated object, and (ii) work for all physically plausible scenarios. To fulfill these requirements while allowing easy interpretation of the errors, we propose to relate identification errors to the inertial parameters and size of the object's bounding box. The error metric for the mass is simply defined as the relative error

$$e_m = \left|\frac{\hat{x} - x}{m}\right| \cdot 100\% \ , \tag{3.15}$$

where $\hat{x}$ and x are the estimated and true mass values, respectively. The error metric for

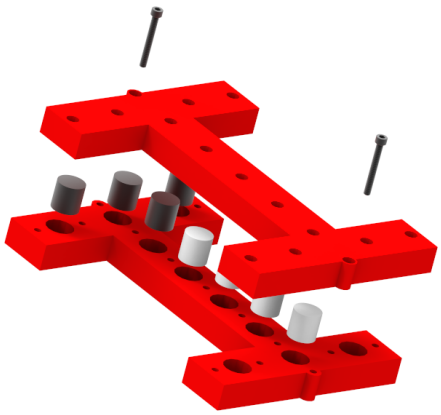
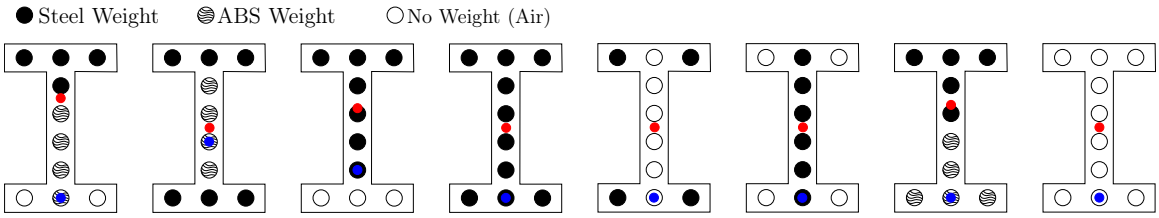


Figure 3.2: Rendering of the modular test object in the *Hammer* configuration with black steel and white ABS weights.



(a) Hammer (b) Barbell (c) Tee (d) Uniform (e) Corners (f) Rod (g) Half-N-Half (h) Empty

Figure 3.3: Eight weight configurations used to benchmark the identification algorithms. The CoM (red point) and TCP location (blue point) are shown. The steel and acrylonitrile butadiene styrene plastic (ABS) weights are 101 and 17 grams each, respectively.

the CoM is defined as

$$e_{c_i} = \left| \frac{\hat{x}^{(i)} - x^{(i)}}{a^{(i)}} \right| \cdot 100\% \ , \tag{3.16}$$

where $a^{(i)}$ is the length of the bounding box of the object along axis i. Finally, the error metric for the inertia tensor is defined as

$$e_{I_{ij}} = \left| \frac{(\hat{x}^{(i,j)} - x^{(i,j)})}{\frac{m}{12} \cdot \left(\delta_{i,j} \left({a^{(i)}}^2 + {a^{(j)}}^2 + {a^{(k)}}^2 \right) - a^{(i)} a^{(j)} \right)} \right| \cdot 100\% \ , \tag{3.17}$$

where $\delta_{i,j}$ is the Kronecker delta, and $\hat{x}^{(i,j)}$ is the estimation of the (i,j) element of the inertia tensor. For moments of inertia (i.e., diagonal elements of the inertia matrix), the error metric in (3.17) relate the identification error to the moments of inertia of a solid cuboid encompassing the object. For products of inertia (i.e., off-diagonal elements of the inertia matrix), the error metric in (3.17) relates the identification error to a term proportional to the mass and volume of the object. The average identification error for the CoM is computed by averaging the errors along the three axes. Since $\mathbf{I}$ is symmetric, the inertia matrix average error (henceforth referred to as the *inertia error*) is obtained by averaging the six errors on and above the main diagonal. In the following section, simulation experiments confirm that our error metric is interpretable, scale-invariant and does not break for all physically plausible scenarios

Simulation Experiments

Point mass density. The number of point masses used to discretize the object impacts the accuracy of the identification algorithms, as well as the compute time. While a greater point mass density can better approximate the object's mass distribution, it also increases the number of variables in Problem 1. To investigate the impact of point mass density on the accuracy and compute time of PMD, we performed simulations with a varying number of point masses under noisy conditions. The results in Figure 3.4 show that the accuracy of PMD improves with a greater point mass density, up to a limit imposed by the noise level.

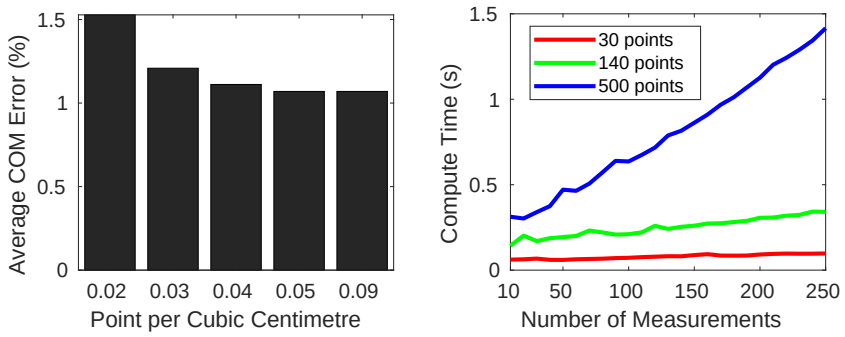


Figure 3.4: *Left*: Accuracy improves, to some extent, with a greater point mass density. *Right*: Impact of the number of point masses on the compute time as more data is gathered.

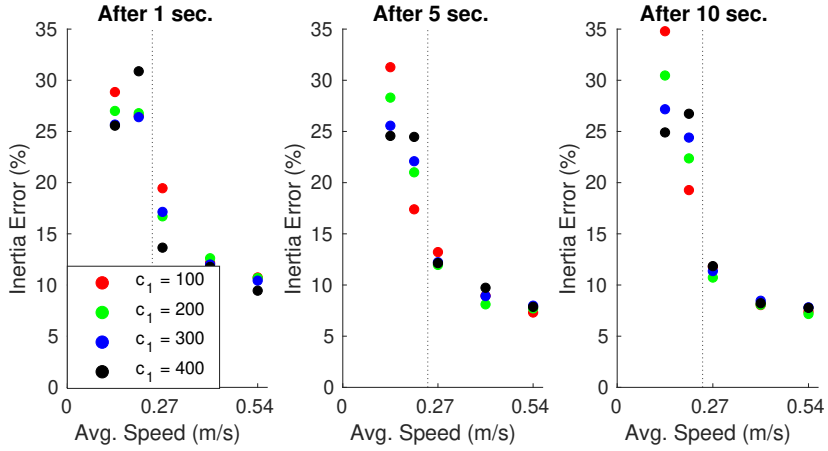


Figure 3.5: PMD accuracy over time as a function of average TCP speed for four different values of the c_1 parameter and five different velocities. The dotted line denotes the maximal TCP velocity of collaborative robots as per ISO 10218-1.

The compute time increases linearly with the number of measurements, as expected, but the rate at which it increases depends on the number of point masses. In all subsequent experiments, a point mass density of 0.04 points/cm^3 was used, corresponding to 56 point masses for our object.

Impact of the c_1 hyperparameter. The c_1 hyperparameter in (3.13) controls the level of dynamism required for the full model to weight 99.5% in Problem 1's objective function. For a given noise level, a c_1 value that is too low will result in decreased accuracy due to the full model being used while the reduced model would yield better results (e.g., due to a low SNR). Conversely, a c_1 value that is too high will result in decreased accuracy due to the reduced model being used when the SNR is large enough for the full model to provide more accurate results. A series of simulation experiments were performed at five velocities and for four c_1 values with the moderate noise level in Table 3.2. The c_1 value yielding the best performance at 0.2 m/s (close to the maximal allowed TCP velocity) with one second of data is $c_1 = 300$ (blues dots) according to Figure 3.5. This value was used in all subsequent

simulation experiments.

Identification accuracy benchmark. Our proposed algorithm was compared to OLS, RTLS, and GEO on a series of experiments where a simulated robot arm manipulated test objects for which ground truth inertial parameters were known. To ensure a fair comparison of the identification algorithms, experiments are conducted with the wielded object attaining average angular velocities of 1, 1.5, 2, 3, and 4 rad/s, and average linear velocities of 0.14, 0.2, 0.27, 0.41, and 0.54 m/s. For each object in Figure 3.3 and for each velocity, simulated experiments were performed under four different noise levels for a total of 640 experiments. All algorithms based on optimization problems (i.e., PMD, OLS, GEO) used the MOSEK (Andersen and Andersen, 2000) solver on the same machine for a fair comparison. For each experiment, the average error of the inertia tensor as computed with (3.17) was used to quantify the identification performance. Figure 3.6 and Figure 3.7 summarize results from the experiments where noise was added to the signals.

Fast identification benchmark. In a realistic manipulation scenario, only a very short amount of time can be devoted to inertial parameter identification. Indeed, the median duration of manipulation tasks during ADL is 2.3 seconds according to Table 3.1. Assuming a standard force-torque sensing frequency of 100 Hz and taking into account the duration of the optimization procedure, approximately 150 observations can be gathered before the algorithm needs to generate its solution. To test the performance of the algorithms in this setting, we performed experiments with short (i.e., $\sim$1.5 seconds) motion trajectories performed at three velocities (i.e., 1.0, 1.5, 2.0 rad/s), with moderate noise added to the forces and torques, and with velocity and acceleration signals estimated through Kalman filtering as described in (Farsoni et al., 2017). In this setting, the average condition number of the scaled regressor matrix $\mathbf{A}$, according to (Gautier and Khalil, 1992), is 147, 68.4, and 58.0 for velocities of 1.0, 1.5, and 2.0 rad/s respectively. These moderately low condition numbers suggest that $\mathbf{A}$ is relatively well-conditioned in our experiments and that identification algorithms should converge. Table 3.3 shows the average accuracy of each algorithm (with PMD using $c_1 = 300$). In our experiments, PMD and GEO always produced physically consistent solutions while OLS and RTLS produced implausible solutions most of the time.

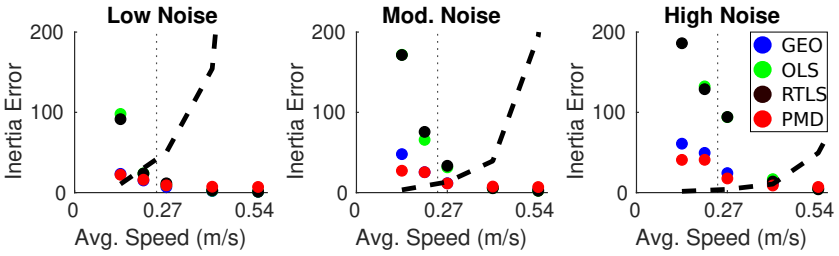


Figure 3.6: Average performance after 10 sec. for three noise levels and five velocities. The vertical line is ISO 10218-1 max. TCP velocity and the the bold dashed line is the SNR.

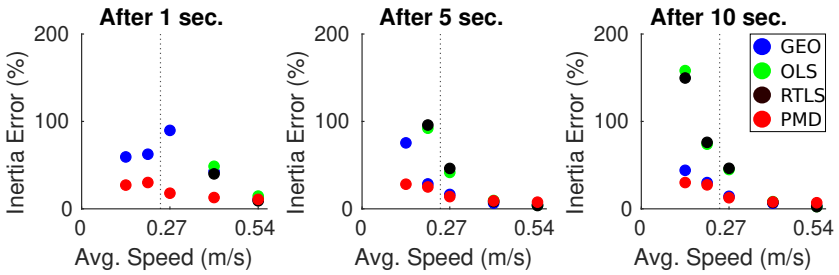


Figure 3.7: Average performance of each algorithm over time for the moderate noise level and at five end-effector velocities. The dotted line is ISO 10218-1 maximal TCP velocity.

As a sanity check, we tested PMD in the scenario where stop-and-go motions are performed (i.e., zero velocity and acceleration while recording data). In this setting, the errors for the mass and CoM were both lower than 0.1%.

Robot Experiments

In order to verify that the PMD algorithm performs well in real settings, we built a physical version of the modular test object to enable verification against ground truth. Experiments were carried out using a Robotiq FT-300 FT sensor sensor that was attached to the end-effector of a uFactory xArm 7 robotic arm. The test object was secured to the flange of the sensor (as seen in Figure 3.1) after weights were placed in one of the eight configurations shown in Figure 3.3. For each object configuration, the robot performed an oscillating motion in which the three robot joints proximal to the end-effector followed sinusoidal trajectories (similar to those in the simulation experiments). During the identification trajectory, the average angular velocity of the test object was 1.1 rad/s.

Akin to the simulation experiments, each identification algorithm was tested on data gathered with the eight object configurations, and the motion trajectory was limited to 1.5 seconds. PMD was tested with $c_1 = 5 \cdot 10^3$ (PMD-5K) and $c_1 = 25 \cdot 10^3$ (PMD-25K), and observations were used after the uncertainty estimates from the Kalman filters observing end-effector kinematics had stabilized. For this experiment, the condition number of $\mathbf{A}$ is 178—larger than in the simulations but still reasonably small. The average error across all

Table 3.3: Inertial parameter accuracy comparison after a limited number of observations have been made (i.e., ∼150) for three different velocities.

	Mass Error (%)			CoM Error (%)			Inertia Tensor Error (%)		
rad/s	1.0	1.5	2.0	1.0	1.5	2.0	1.0	1.5	2.0
OLS	1.36	2.88	3.22	34.4	44.0	36.1	>500	>500	>500
RTLS	1.84	**1.77**	2.41	97.5	37.7	38.2	>500	>500	>500
GEO	1.45	2.65	4.09	33.0	33.1	32.6	>500	252	109
PMD	**1.34**	2.15	**1.91**	**9.83**	**15.5**	**16.4**	**44.1**	**43.5**	**43.6**

Table 3.4: Average identification errors with the xArm 7 robotic arm and the FT-300 force-torque sensor.

	Mass (%)	CoM (%)	Inertia Tensor (%)
OLS	**3.18**	24.9	>500
RTLS	11.7	>500	>500
GEO	3.75	30.3	>500
PMD-5K	4.51	8.12	36.75
PMD-25K	4.45	**4.50**	**29.11**

studied configurations was computed for each algorithm and is shown in Table 3.4.

The estimated inertial parameters can be used to predict the effort required to move an object along a given trajectory (see Figure 3.8 for an example with PMD-5K) with reasonable accuracy, enabling effort-based motion planning as done in (Mavrakis et al., 2016).

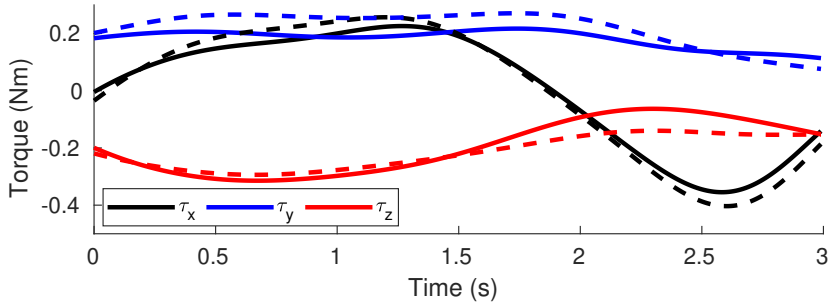


Figure 3.8: Torques predicted using the true (solid) and estimated (dashed) parameters for the real *Tee* object and for PMD-5K.

Discussion

As we can expect, results from our simulation experiments suggest that a greater density of point masses leads to greater accuracy, at the cost of longer compute time. Interestingly, the accuracy of the CoM estimate when using a density of 0.09 points/cm^3 was not significantly better than when using a density of 0.04 points/cm^3, suggesting that the latter density is a good compromise between accuracy and compute time. The dimnishing returns observed with greater point mass densities can be explained by the fact that we mainly use points masses as intermediates to enforce physical consistency rather than to accurately model the mass distribution of the object. Consequently, a small error in the object shape should have little to no impact on the accuracy of the inertial parameter estimates. In this work, we do not assume knowledge of the noise magnitude in the velocity and acceleration data. Instead, we tune the c_1 hyperparameter on a per-robot basis (e.g., we use $c_1 = 300$ with our simulated robot while we use $c_1 = 5000$ with the xArm 7). Future work could consider setting c_1 based on estimates of the noise power, such that a predetermined SNR is achieved

independently of the robot. We note that all of the algorithms benchmarked in this work are complementary and that each is suitable in a specific context:

- OLS (Atkeson et al., 1986) is the simplest algorithm and performs well with high-speed manoeuvres, but does not guarantee physical consistency;
- RTLS (Kubus et al., 2008) has the lowest asymptotic error for trajectories with large signal-to-noise ratios, but converges more slowly and does not guarantee physical consistency;
- GEO (Lee et al., 2019) performs better than RTLS under high noise and guarantees physical consistency, but requires a good initial guess and a well-tuned regularization factor;
- PMD is fast, more robust for small signal-to-noise ratios, and guarantees physical consistency, but requires knowledge of the shape and pose of the object.

3.1.5 Summary

In the first part of this chapter, we proposed an inertial parameter estimation method well suited to the velocity regime of cobots during object manipulation. Our algorithm uses a weighting scheme based on the *dynamism* of the motion; the estimator applies a reduced dynamics model when appropriate to yield a solution quickly. We rely on a discretization of the object shape, where the requisite information is known or is captured via the cobot's perception system, to guarantee physical consistency of the inertial parameters. Our proposed algorithm performs well in real-world experiments representative of practical manipulation tasks.

3.2 A Method Based on Visual Part Segmentation

The first part of this chapter introduced the problem of low SNRs faced by cobots when estimating the inertial parameters of manipulated objects. To alleviate this problem, we proposed to combine a reduced dynamics model with the full model via a weighting scheme based on motion dynamism. The resulting identification algorithm was shown to perform better in situations where the SNR is low, at the cost of introducing an approximation in the dynamics model. In the second part of this chapter, we propose an alternative identification method that relaxes the need for accurate kinematics estimates but can nonetheless determine the complete set of inertial parameters, under some mild assumptions.

Our technique uses information obtained from an RGB-D camera to improve the speed and accuracy of inertial parameter identification. At the centre of our proposed algorithm is the key observation that man-made objects are usually built from a few parts having approximately homogeneous densities.

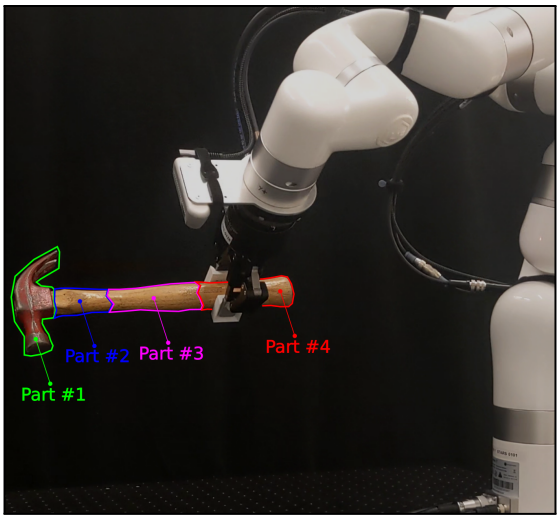


Figure 3.9: The shape of a manipulated object is segmented into parts that are assumed to be homogeneous. By identifying the mass of each part, the full set of inertial parameters can be estimated from safer "stop-and-go" motions.

Assumption 1 (Homogeneous Density of Parts)**.** *An object is made of N parts $\{r_1, \ldots, r_N\}$, each with a shape $V_j \subset \mathbb{R}^3$. The mass density is such that $\forall\, \mathbf{p} \in V_j$, $\rho(\mathbf{p}) = \rho_j$ where $\rho_j > 0$ is part r_j's constant density.*

Critically, Assumption 1 can be used to simplify the identification problem by exploiting measurements of the shape of a manipulated object. This is accomplished by noting that, once a cobot has determined the shape of an object, the object's inertial parameters depend solely on the mass of each of its homogeneous parts. We combine a surface-based point clustering method with a volumetric shape segmentation algorithm to quickly produce a part-level segmentation of a manipulated object; the segmented representation is then used by our algorithm to accurately estimate the object's inertial parameters. To benchmark our algorithm, we create and utilize a novel dataset consisting of realistic meshes, segmented point clouds, and inertial parameters for 20 common workshop tools. We refer to our overall approach as *Homogeneous Part Segmentation* (HPS).[2] Our main contributions are:

- a formulation of inertial parameter identification that incorporates Assumption 1;
- a method combining the algorithms proposed in (Attene et al., 2008) and (Lin et al., 2018) to improve part segmentation speed;
- a dataset of models of 20 common workshop tools with 3D meshes, point clouds, ground truth inertial parameters, and ground truth part-level segmentation for each object; and

[2] We provide code, our complete simulation dataset, and a video showcasing our algorithm at https://papers.starslab.ca/part-segmentation-for-inertial-identification/.

- experiments highlighting the benefits of HPS when compared to two benchmark algorithms.

We undertake a series of simulation studies with our novel dataset and carry out a real-world experiment to assess the performance of the proposed algorithm on a real cobot. We show that, under noisy conditions, our approach produces more accurate inertial parameter estimates than competing algorithms that do not utilize shape information. Finally, we demonstrate the real-world performance and accuracy of HPS by performing an intricate 'hammer balancing act' autonomously and online with a low-cost collaborative robotic arm.

3.2.1 Prior Work in Part-Level Object Segmentation

Although there is no formal definition of a "part" of a manipulated item, humans tend to follow the so-called *minima rule*: the decomposition of objects into approximately convex contiguous parts bounded by negative curvature minima (Hoffman and Richards, 1984). This rule has provided inspiration for many part segmentation methods reviewed in (Rodrigues et al., 2018), which benchmarks several techniques on the Princeton dataset (Chen et al., 2009) and divides approaches into surface-based, volume-based, and skeleton-based algorithms. Surface-based and volume-based mesh segmentation algorithms are reviewed in (Shamir, 2008), highlighting a tradeoff between segmentation quality and the number of parts produced by the algorithm. Skeleton-based segmentation methods for 3D shapes, which capture structural information as a lower-dimensional graph (i.e., the shape's *skeleton*), are reviewed in (Tagliasacchi et al., 2016). The approach in (Lin et al., 2020), which is based on the medial axis transform (Li et al., 2015), exploits prior knowledge of an object's skeleton to produce a *top-down* segmentation about 10 times faster than *bottom-up* methods that use local information only.

The method of Kaick et al. (2014) also segments incomplete point clouds into approximately convex shapes, but is prohibitively slow for manipulation tasks, with an average computation time of 127 seconds on the Princeton benchmark (Chen et al., 2009) as reported in (Lin et al., 2020). In contrast, learning-based methods for part segmentation can struggle to generalize to out-of-distribution shapes but are usually faster than geometric techniques (Dou et al., 2022; Lin et al., 2021; Rodrigues et al., 2018). Our approach to segmentation does not apply any learned components but is fast enough for real-time use by a collaborative robot.

A Hierarchical Tetrahedral Clustering (HTC) proposed in (Attene et al., 2008), enabled by fast tetrahedralization (Si, 2015) and quick convex hull computation (Barber et al., 1996), can perform well if a watertight mesh can be reconstructed from the input point cloud. This technique, which we describe in detail in Section 3.2.3, is an important element of our part segmentation pipeline. Another key component of our approach is the surface-based algorithm proposed in (Lin et al., 2018), which uses a dissimilarity metric to iteratively merge nearby mesh patches, taking special care to preserve part boundaries.

3.2.2 Inertial Parameters of Homogeneous Parts

In this section, we describe our inertial parameter identification technique, which assumes that an object has been segmented into its constituent parts. By determining the mass of each segment, we are able to identify the full set of inertial parameters, or to provide an approximate solution when Assumption 1 is not respected.

In the following, reference frames $\boldsymbol{\mathcal{F}}_b$ and $\boldsymbol{\mathcal{F}}_s$ are attached to the object and to the FT sensor, respectively. The reference frame $\boldsymbol{\mathcal{F}}_w$ is fixed to the base of the robot and is assumed to be an inertial frame, such that the gravity vector expressed in the sensor frame is given by $\mathbf{g}_s = {}_s\mathbf{R}_w[0, 0, -9.81]^\mathsf{T}$. The orientation of $\boldsymbol{\mathcal{F}}_b$ relative to $\boldsymbol{\mathcal{F}}_s$ is given by ${}_s\mathbf{R}_b$ and the origin of $\boldsymbol{\mathcal{F}}_b$ relative to $\boldsymbol{\mathcal{F}}_s$, expressed in $\boldsymbol{\mathcal{F}}_w$, is given by ${}^w_s\mathbf{p}_b$.

Formulation of the Optimization Problem

For a part r_j, under Assumption 1, the k-th moment of a mass distribution discretized into n point masses is given by

$$\int_V [{}_b\mathbf{p}_i]^k_\times \ \rho({}_b\mathbf{p}_i)dV \approx m_{r_j} \sum_i^n [{}_b\mathbf{p}_i]^k_\times \ , \tag{3.18}$$

where $\rho({}_b\mathbf{p}_i)$ is the mass density at ${}_b\mathbf{p}_i$ and m_{r_j} is the mass of part r_j. For a homogeneous mass density, the centre of mass is at the centroid

$${}_b\mathbf{p}_{r_j} = \frac{1}{n} \sum_i^n {}_b\mathbf{p}_i \ . \tag{3.19}$$

The inertia matrix of the discretized part computed relative to $\boldsymbol{\mathcal{F}}_s$ is given by

$$\begin{aligned} {}_s\mathbf{I}_{r_j} &= {}_s\mathbf{R}_b \, {}^b_s\mathbf{I}_{r_j} \, {}_s\mathbf{R}_b^\mathsf{T} && (3.20)\\ &= {}_s\mathbf{R}_b \left(\frac{m_{r_j}}{n} \sum_i^n \left[{}^b_s\mathbf{p}_i \right]^2_\times \right) {}_s\mathbf{R}_b^\mathsf{T} && (3.21)\\ &= {}_s\mathbf{R}_b \left(\frac{m_{r_j}}{n} \sum_i^n [{}_b\mathbf{p}_i + {}_b\mathbf{R}_s \, {}_s\mathbf{p}_b]^2_\times \right) {}_s\mathbf{R}_b^\mathsf{T} \ , && (3.22) \end{aligned}$$

where point-mass inertia matrix from (2.28) and the similarity transform in (2.38) were used. Hence, the part's vector of inertial parameters measured relative to $\boldsymbol{\mathcal{F}}_s$ is given by

$$\begin{aligned} {}_s\boldsymbol{\phi}_{r_j} &= \begin{bmatrix} m_{r_j} & m_{r_j} \, {}_s\mathbf{p}_{r_j}^\mathsf{T} & \operatorname{vech}\left({}_s\mathbf{I}_{r_j} \right)^\mathsf{T} \end{bmatrix}^\mathsf{T} && (3.23)\\ &= m_{r_j} \begin{bmatrix} 1 \\ {}_s\mathbf{p}_{r_j} \\ \operatorname{vech}\left({}_s\mathbf{R}_b \left(\frac{1}{n} \sum_i^n [{}_b\mathbf{p}_i + {}_b\mathbf{R}_s \, {}_s\mathbf{p}_b]^2_\times \right) {}_s\mathbf{R}_b^\mathsf{T} \right) \end{bmatrix} \ , && (3.24) \end{aligned}$$

where vech(·) is the *vector-half* operator (Henderson and Searle, 1979) that extracts the elements on and above the main diagonal. By keeping ${}_b\mathbf{p}_i$ fixed and by measuring ${}_s\mathbf{p}_b$ and ${}_s\mathbf{R}_b$ with the robot's perception system, the sole knowledge of the part's mass m_{r_j} is sufficient to fully define (3.24). Hence, assuming that the robot's perception system can provide ${}_b\mathbf{p}_i$ and that ${}_s\mathbf{p}_b$ and ${}_s\mathbf{R}_b$ are either known or measured, the inertial parameters of a homogeneous part depend solely on its mass. Similarly, the inertial parameters of a rigid object can be expressed as a function of the masses of its constituent parts.

For "stop-and-go" motions, where force measurements are taken while the robot is immobile, only the mass and CoM are identifiable (Nadeau et al., 2022). Nonetheless, a stop-and-go trajectory greatly reduces noise in the data matrix, because accurate estimates of the end-effector kinematics are not needed (Nadeau et al., 2022). Assuming that the manipulated object is built from up to four homogeneous parts[3], finding the mass of each part enables the identification of the complete set of inertial parameters even when stop-and-go trajectories are performed. This identification requires measuring the wrench $\mathbf{b}_k$ at time step k and relating the wrench to the masses $\mathbf{m}$ via

$$\underbrace{\begin{bmatrix}\mathbf{f}_s\\ \boldsymbol{\tau}_s\end{bmatrix}}_{\mathbf{b}_k} = \underbrace{\begin{bmatrix}\mathbf{g}_s[1 & 1 & 1 & 1]\\ -[\mathbf{g}_s]_\times[{}_s\mathbf{p}_{r_1} & {}_s\mathbf{p}_{r_2} & {}_s\mathbf{p}_{r_3} & {}_s\mathbf{p}_{r_4}]\end{bmatrix}}_{\widetilde{\mathbf{A}}_k} \underbrace{\begin{bmatrix}m_{r_1}\\ m_{r_2}\\ m_{r_3}\\ m_{r_4}\end{bmatrix}}_{\mathbf{m}}, \tag{3.25}$$

stacking K matrices such that $\mathbf{A} = [\widetilde{\mathbf{A}}_1^\mathsf{T}, ..., \widetilde{\mathbf{A}}_K^\mathsf{T}]^\mathsf{T}$ and $\mathbf{b} = [\mathbf{b}_1^\mathsf{T}, ..., \mathbf{b}_K^\mathsf{T}]^\mathsf{T}$. Minimizing the Euclidean norm leads to the convex optimization problem

$$\begin{aligned}\min_{\mathbf{m}\in\mathbb{R}^4} \quad & ||\mathbf{A}\mathbf{m} - \mathbf{b}||_2 \\ s.t. \quad & m_{r_j} \geq 0 \ \ \forall j \in \{1, \ldots, 4\},\end{aligned} \tag{3.26}$$

which can be efficiently solved with standard methods.

3.2.3 Visual Part Segmentation

In this section, we combine a part segmentation method that uses local information (e.g., surface normals) with a second method that relies on structural information (e.g., shape convexity). The Python implementation of our part segmentation algorithm, which is described by Algorithm 1, includes an open-source version of (Attene et al., 2008) as well as our variant of (Lin et al., 2018) that makes use of colour information.

Defining the shape of an object from a point cloud involves reconstructing the object

[3]For stop-and-go trajectories, the rank of the data matrix is four when using non-degenerate poses as described in the appendix of (Nadeau et al., 2022). Dynamic trajectories can increase the rank of $\mathbf{A}$ to 10, enabling mass identification of up to 10 unique homogeneous parts.

surface (i.e., shape reconstruction). In this work, the ball-pivoting algorithm (Bernardini et al., 1999) is used owing to its speed and relative effectiveness on point clouds of low density. Shape reconstruction can be challenging for objects with very thin parts (e.g., a saw) and a thickening of the object through voxelization is performed beforehand.

As stated by (3.18), the moments of a mass distribution are computed by integrating over the shape of the distribution. Hence, volumetric part segmentation is sufficient for identification of the true inertial parameters of an object. To obtain such a representation of the object from its surface mesh, tetrahedralization is performed via TetGen (Si, 2015) and the resulting tetrahedral mesh is supplied as an input to the part segmentation algorithm.

Our method makes use of the HTC algorithm (Attene et al., 2008), which iteratively merges clusters of tetrahedra such that the result is as convex as possible while also prioritizing smaller clusters. HTC maintains a graph with nodes representing clusters and edges representing adjacency, and with an edge cost based on the concavity and size of the connected nodes. The concavity of a cluster is computed by subtracting its volume from its convex hull and an edge cost is defined by

$$c_{ij} = \text{CVXHULL}\left(C_i \cup C_j\right) - \text{VOL}(C_i \cup C_j)$$
$$\text{Cost}(i,j) = \begin{cases} c_{ij} + 1, & \text{if } c_{ij} > 0 \\ \dfrac{|C_i|^2 + |C_j|^2}{N^2}, & \text{otherwise} \end{cases}, \tag{3.27}$$

where $|C_i|$ is the number of elements in cluster C_i and N is the total number of elements. The edge associated with the lowest cost is selected iteratively and the connected nodes are merged into a single cluster, resulting in the hierarchical segmentation.

To make part segmentation faster, we perform an initial clustering such that HTC requires fewer convex hull computations, which is by far the most expensive operation. The initial clustering is provided through a bottom-up point cloud segmentation algorithm (Lin et al., 2018) that clusters points together based on heuristic features and chooses a representative point for each cluster. Two adjacent clusters are merged if the dissimilarity between their representative points is lower than some threshold β (n.b. the λ symbol is used in (Lin et al., 2018)). The value of β increases with each iteration; the process halts when the desired number of clusters is obtained. In our implementation, the dissimilarity metric is defined as:

$$\text{D}(C_i, C_j) = \lambda_p \left\|\mathbf{p}_i - \mathbf{p}_j\right\| + \lambda_l \left\|\mathbf{l}_i - \mathbf{l}_j\right\| + \lambda_n \left(1 - |\mathbf{n}_i \cdot \mathbf{n}_j|\right) \tag{3.28}$$

where each λ is a tunable weight, $\mathbf{p}_i$ is the position of the representative point of cluster C_i, $\mathbf{l}_i \in \mathbb{N}^3$ is the RGB representation of its colour, and $\mathbf{n}_i$ is its local surface normal.

To enable accurate part segmentation, the initial clustering should not cross the boundaries of the parts that will subsequently be defined by the HTC algorithm. Therefore, initial clustering is stopped when the number of clusters (e.g., 50 in our case) is much larger than

the desired number of parts. The desired number of clusters does not need to be tuned on a per-object basis as long as it is large enough.

Algorithm 1: HTC (**2**) with initial clustering (**1**)

input : A point cloud
output: A segmented tetrahedral mesh
1 Initialize similarity threshold β as done in (Lin et al., 2018)
Initialize each point as a cluster
while *number of clusters* $>$ *desired number* **do**
  foreach *existing cluster* C_i **do**
    foreach *neighboring cluster* C_j **do**
      if $D(C_i, C_j) < \beta$ **then**
        Merge C_j into C_i
  $\beta \leftarrow 2\beta$
foreach *point* **p** *at the border of two clusters* **do**
  Merge **p** into the most similar cluster
2 Perform surface reconstruction and tetrahedralization
Associate each tet. to its nearest cluster
Initialize a node in the graph for each cluster
foreach *node* **do**
  if *two tet. from two nodes share a face* **then**
    Create an edge between the nodes and compute edge-cost with (3.27)
while *number of edges* > 0 **do**
  Find edge with lowest cost and merge its nodes
  Create a parent node that contains merged nodes
Label each tet. with its associated cluster number

3.2.4 Experiments

In this section, each component of the proposed method is evaluated on 20 objects with standard metrics, and our entire identification pipeline is benchmarked in 80 scenarios. The practicality of the proposed approach is also demonstrated through a real 'hammer balancing act' experiment using relatively inexpensive robot hardware (see Figure 3.12).

To conduct simulation experiments with realistic objects and evaluate the performance of part segmentation and identification, a dataset with ground truth inertial parameters and segments is needed. To the best of the authors' knowledge, no such dataset is freely available. From computer-aided design (CAD) files contributed by the community, we built a dataset containing 20 commonly-used workshop tools. For each object, our dataset contains a watertight mesh, a coloured surface point cloud with labelled parts, the object's inertial parameters, and a reference frame specifying where the object is usually grasped. This dataset of realistic objects enables the evaluation of shape reconstruction, part segmentation, and inertial parameter identification.

Table 3.5: Average computation time and segmentation error per object from our dataset with standard deviations in parentheses. Initial clustering significantly reduces the runtime with little impact on the part segmentation.

Algorithm	USE	GCE	Time (s)
HTC	0.1 (0.13)	0.05 (0.10)	9.73 (5.56)
HTC with Initial Clustering	0.1 (0.11)	0.07 (0.11)	3.48 (1.00)

(a) (b) (c)

Figure 3.10: Pictures of objects scanned by the RGB-D camera on the manipulator, next to their segmented meshes; points are located at the tetrahedra's centroids (screwdriver is rotated).

Experiments on Objects from Our Dataset

The quality of a shape reconstructed from point cloud data is evaluated by computing the Hausdorff distance between the ground truth mesh and the reconstructed mesh, both scaled such that the diagonal of their respective bounding box is one metre in length. Part segmentation performed with our proposed variation of HTC is evaluated via the undersegmentation error (USE) (Levinshtein et al., 2009), and with the global consistency error (GCE) (Martin et al., 2001; Chen et al., 2009), with results summarized in Table 3.6. The USE measures the proportion of points that are crossing segmentation boundaries, or *bleeding out* from one segment to another. The GCE measures the discrepancy between two segmentations, taking into account the fact that one segmentation can be more refined than the other. Both evaluation metrics represent dimensionless error ratios and are insensitive to over-segmentation, as discussed in Section 3.2.5.

Table 3.5 summarizes the runtime performance obtained by augmenting HTC with the initial clustering in Algorithm 1. While the average segmentation error is almost identical in both cases, Algorithm 1 executes in about a third of the time, owing to the smaller number of convex hull computations performed. The reconstruction and part segmentation can also be qualitatively evaluated on point clouds obtained from RGB-D images taken by a Realsense D435 camera at the wrist of the manipulator. To complete a shape that is partly occluded by the support table, the point cloud is projected onto the table plane, producing a flat bottom that might not correspond to the true shape of the object. For instance, the left

Table 3.6: Evaluation of part segmentation (USE and GCE) and shape reconstruction (Hausdorff). As expected, objects with a single part do not have GCE and USE errors while objects with many parts have larger segmentation errors.

Object	USE	GCE	Hausdorff (mm)	Mass (g)	#Parts
Allen Key	0	0	8.2	128	1
Box Wrench	0	0	10.9	206	1
Measuring Tape	0	0	11.4	136	1
Ruler	0	0	14.8	9	1
Screwdriver	0.013	0.001	12.1	30	2
Nut Screwdriver	0.002	0.002	16	81	2
Rubber Mallet	0.011	0.009	15.8	237	2
Bent Jaw Pliers	0.176	0.01	16.5	255	3
Machinist Hammer	0.018	0.012	13.6	133	2
Pliers	0.123	0.041	10.7	633	3
C Clamp	0.103	0.077	9.6	598	5
Adjustable Wrench	0.117	0.098	14.2	719	4
Hammer	0.08	0.098	14.7	690	3
File	0.057	0.102	17.9	20	3
Socket Wrench	0.141	0.123	7.4	356	5
Hacksaw	0.129	0.128	11.1	658	7
Clamp	0.259	0.181	17.6	340	7
Vise Grip	0.158	0.287	17.2	387	8
Electronic Caliper	0.42	0.316	9.1	174	14
Vise Clamp	0.296	0.373	15.1	225	9

side of the rotated screwdriver point cloud in Figure 3.10 is the result of such a projection.

The proposed identification algorithm is tested in simulation under four noise scenarios where zero-mean Gaussian noise is added to the signals. The standard deviations of the noise values are based on the specifications of the Robotiq FT-300 sensor as described in Table 3.7. The identification trajectory used to generate the regressor matrix $\mathbf{A}$ in (3.26) has the robot lift the object and successively orient it such that gravity is aligned with each axis of $\boldsymbol{\mathcal{F}}_b$, stopping for a moment at each pose and moving relatively quickly between poses, as shown in Figure 3.11. The average condition number (Gautier and Khalil, 1992) of the scaled regressor matrix for all simulated trajectories is 74, 92, and 99 for the low, moderate, and high noise scenarios, respectively. These relatively low condition numbers confirm that our identification trajectory is non-degenerate.

The performance of our proposed algorithm (HPS) is compared against the classical algorithm proposed in (Atkeson et al., 1986) (OLS) and to the more modern algorithm proposed in (Lee et al., 2019) (GEO). The latter was provided with a prior solution consisting of the true mass of the object, and the CoM and inertia tensor resulting from a homogeneous mass distribution for the object ($\alpha = 10^{-5}$ was used). HPS only uses measurement data when both the linear and angular accelerations are below 1 unit/s^2, corresponding to the

Table 3.7: Standard deviations of the zero-mean Gaussian noise added to accelerations and force signals in simulation.

Scenario	Ang. Acc.	Lin. Acc.	Force	Torque
Low Noise	0.25	0.025	0.05	0.0025
Moderate Noise	0.5	0.05	0.1	0.005
High Noise	1	0.1	0.33	0.0067

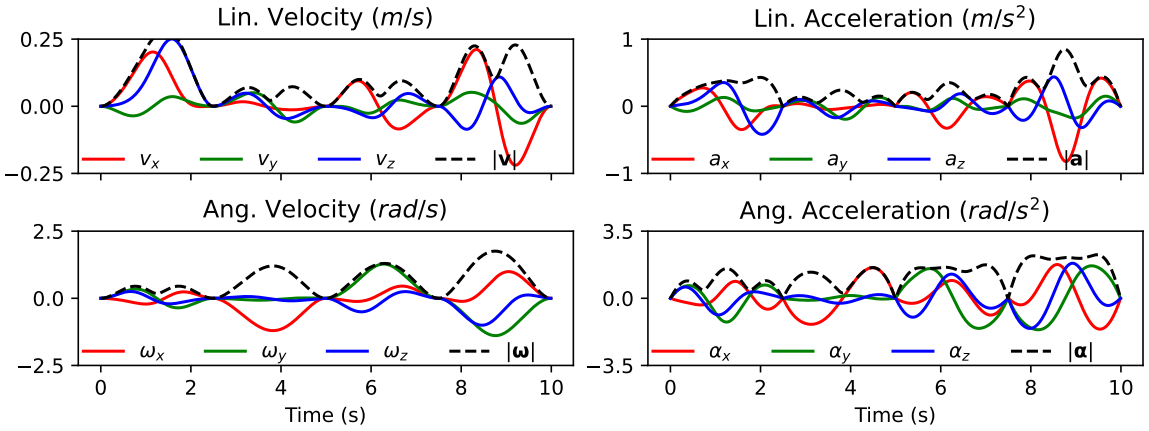


Figure 3.11: Kinematics of the identification trajectory performed with the Rubber Mallet. The norm of the velocity (dashed line) goes slightly above the standard maximum speed for a cobot.

stalled timesteps of the stop-and-go motion, whereas OLS and GEO use all data available.

The accuracy of inertial parameter identification is measured via the Riemannian geodesic distance (Lang, 1999)

$$e_{\text{Rie}} = \sqrt{\left(\frac{1}{2}\sum \boldsymbol{\lambda}\right)} \tag{3.29}$$

where $\boldsymbol{\lambda}$ is the vector of eigenvalues for

$$P({}^{s}\phi^{1})^{-1} P({}^{s}\phi^{2}) \tag{3.30}$$

and $P(\phi)$ is the *pseudoinertia* matrix

$$P(\phi) = \begin{bmatrix} \frac{\text{tr}(\mathbf{I})}{2}\mathbf{1}_3 - \mathbf{I} & m\mathbf{c} \\ m\mathbf{c}^{\mathsf{T}} & m \end{bmatrix} \tag{3.31}$$

that is required to be symmetric positive definite (SPD) (Wensing et al., 2017). The metric e_{Rie} is the distance between the estimated and ground truth inertial parameters in the space of SPD matrices. To assist interpretation of the identification error, we also compute the size-based error metrics proposed in (Nadeau et al., 2022), which use the bounding box and mass of the object to produce average *percentage* errors $\bar{\mathbf{e}}_m$, $\bar{\mathbf{e}}_c$, and $\bar{\mathbf{e}}_I$ associated, respectively, with the mass, centre of mass, and inertia tensor estimates. Table 3.8 reports

Table 3.8: Comparison between HPS (ours), OLS, and GEO with various levels of noise, indicating the percentage of solutions that were physically consistent (Cons).

Noise	Algo.	Cons. (%)	$\bar{\mathbf{e}}_m$(%)	$\bar{\mathbf{e}}_c$(%)	$\bar{\mathbf{e}}_I$(%)	e_{Rie}
No	OLS	100	**<0.1**	**<0.1**	**<0.1**	**0.03**
	GEO	100	<0.1	1.35	53.22	1.14
	HPS	100	0.27	0.1	10.28	0.72
Low	OLS	14	0.19	2.13	>500	N/A
	GEO	100	**0.18**	1.34	52.79	1.14
	HPS	100	0.40	**0.32**	**11.12**	**0.74**
Mod.	OLS	4.5	0.36	5.33	>500	N/A
	GEO	100	**0.35**	1.37	51.73	1.14
	HPS	100	0.74	**0.48**	**11.81**	**0.77**
High	OLS	2	0.69	10.11	>500	N/A
	GEO	100	**0.64**	1.58	48.49	1.13
	HPS	100	2.79	**1.07**	**15.00**	**0.87**

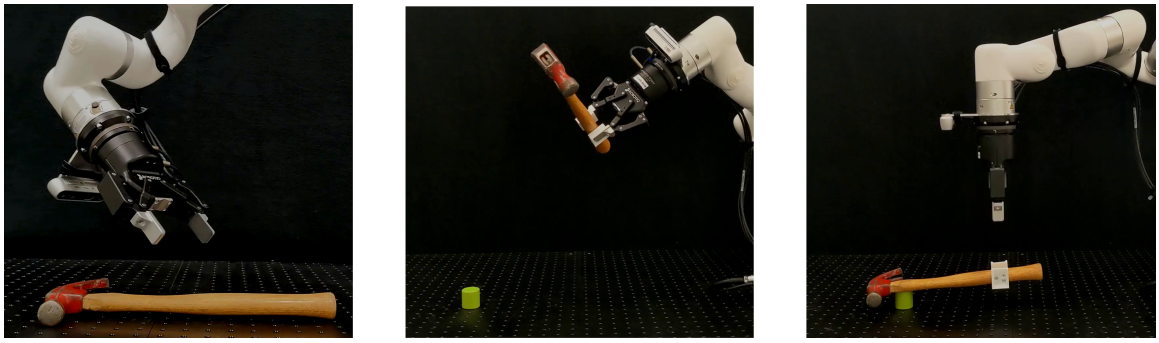
the mean of the entries of the error vector $\bar{\mathbf{e}}$ for each quantity of interest.

Demonstration in Real Settings

To test our proposed method in a realistic setting, we used a uFactory xArm 7 manipulator equipped with a RealSense D435 camera and a Robotiq FT-300 force-torque sensor, as shown in Figure 3.9. First, the hammer in Figure 3.10 was scanned with the camera, producing 127 RGB-D images of the scene in about 30 seconds. The object was then picked up at a predetermined grasping pose where the Robotiq 2F-85 gripper could fit into plastic holders attached to the object (enabling a stable grasp of the handle). A short trajectory that took about 10 seconds to execute was performed while the dynamics of the object were recorded at approximately 100 Hz. Point cloud stitching, mesh reconstruction, and part segmentation can be performed concurrently while the robot executes the trajectory, since these operations take 2.87, 0.24, and 2.82 seconds, respectively. Finally, our proposed method identified the inertial parameters in about 0.5 seconds with MOSEK (Andersen and Andersen, 2000). Using only its estimate of the CoM, the robot autonomously balanced the hammer on a cylindrical target with a radius of 17.5 mm. The entire process is shown in our accompanying video, with summary snapshots provided in Figure 3.12. In contrast, both OLS and GEO returned inaccurate parameter estimates, causing hammer balancing to fail.

3.2.5 Discussion

This work exploits object shape information to produce fast and accurate inertial parameter estimates. The object reconstructions used by our method to determine object shape are often already computed as components of planning and manipulation algorithms, limiting the computational burden introduced by our approach. The decomposition of objects into



(a) (b) (c)

Figure 3.12: The hammer is (a) scanned, (b) picked up for inertial identification, and (c) balanced onto the small green target.

parts also enables fast inertial parameter identification of bodies that have joints (e.g., hedge scissors), since the parameters can be trivially updated following a change in object configuration.

Wrong or inaccurate part segmentation may lead to erroneous parameter estimates. However, if Assumption 1 holds, *over*-segmenting an object does not affect the result of the identification as for a given part, (3.18) can be decomposed into

$$\int_{V_1} \mathbf{p}^k \rho_1(\mathbf{p}) dV_1 + \int_{V_2} \mathbf{p}^k \rho_2(\mathbf{p}) dV_2 = \int_{V} \mathbf{p}^k \rho(\mathbf{p}) dV, \tag{3.32}$$

which is true since $V = V_1 \cup V_2$ and $\rho_1(\mathbf{p}) = \rho_2(\mathbf{p})$. Similarly, if two conceptually distinct parts with the *same* mass density are erroneously combined into one, the result of the identification will remain unaffected. However, if parts with *different* mass densities are considered to be a single part by the algorithm, the identification will fail since (3.32) does not hold when $\rho_1(\mathbf{p}) \neq \rho_2(\mathbf{p})$.

The comparison in Table 3.8 suggests that OLS outperforms other algorithms in the noiseless scenario, which is expected since it is not biased by any prior information. However, OLS nearly always converges to physically inconsistent solutions and becomes inaccurate in the presence of even a small amount of sensor noise. The GEO algorithm performs similarly across noise levels, possibly due to the very good prior solution (i.e., correct mass, homogeneous density) provided, corresponding to the exact solution for objects that have a single part. On average, HPS outperforms OLS and GEO for the identification of CoM and $\mathbf{I}$ when the signals are noisy. The slightly higher $\bar{\mathbf{e}}_m$ for HPS may be caused by the approximation made when using stalling motions (Nadeau et al., 2022).

Experiments with objects from our dataset do not reveal any obvious trends relating the quality of the shape reconstruction (measured via the Hausdorff distance), the quality of the part segmentation (measured via USE and GCE), and the quality of the inertial parameter identification (measured via e_{Rie}). This can be explained by the fact that an object's mass and shape largely determine the signal-to-noise ratios that any identification

algorithm has to deal with. Additionally, a small shape reconstruction error should not impact the identification of the mass and have little impact on the CoM estimate since the optimization problem in (3.26) will naturally offset errors in part centroid locations by changing the mass distribution. In contrast, we expect that small errors in the shape of the object would have a greater impact on the inertia matrix due to the moment of inertia being dependent on the square of the distance between a point mass and the axis of rotation.

As demonstrated by experiments with objects that are mostly symmetrical (e.g., the screwdriver), if the shape of the object is such that the parts' centroids are coplanar, the optimizer will 'lazily' zero out the mass of some parts since they are not required to minimize (3.26). An improved version of HPS could use the hierarchy from HTC to intelligently define segments whose centroids are not coplanar.

3.2.6 Summary

In this section, we leveraged the observation that man-made objects are often built from a few parts with homogeneous densities. We proposed a method to quickly perform part segmentation and inertial parameter identification. We ran 80 simulations in which our approach outperformed two benchmark methods when noise levels are elevated, and we demonstrated real-world applicability by autonomously balancing a hammer on a small target. On average, our proposed algorithm performs well in noisy conditions and can estimate the full set of inertial parameters from 'stop-and-go' trajectories that can be safely executed by collaborative robots.

3.3 Conclusion and Future Work

In this chapter, we have presented two novel methods for the inertial parameter identification of objects in the context of collaborative robotics. The first method considers the signal-to-noise ratio of the terms in the equation of motion and defines a weighting scheme that reduces the impact of noisy signals on the identification. This work was presented at the International Conference on Robotics and Automation (ICRA) and is included in the conference proceedings as:

> **P. Nadeau**, M. Giamou and J. Kelly, *"Fast Object Inertial Parameter Identification for Collaborative Robots"*, 2022 International Conference on Robotics and Automation, Philadelphia, PA, USA, 2022, pp. 3560-3566, doi: 10.1109/ICRA46639.2022.9916213.

The second method proposes to make use of vision, a sensing modality that is ubiquitous in robotics, to enable a part segmentation of the manipulated object and simplify the identification problem. Under mild assumptions, this method can quickly and accurately identify the full set of inertial parameters, even for articulated objects. The results of this work were presented at ICRA 2023 and are included in the conference proceedings as:

P. Nadeau, M. Giamou and J. Kelly, *"The Sum of Its Parts: Visual Part Segmentation for Inertial Parameter Identification of Manipulated Objects"*, 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023, pp. 3779-3785, doi: 10.1109/ICRA48891.2023.10160394.

Building on our contributions, several avenues of future work are promising. Firstly, the motion trajectory performed by the robot to identify the object's inertial parameters could be optimized to minimize the uncertainty in the object shape and in the inertial parameters. This would enable maximum information to be extracted from a short motion. Ideally, the identification procedure would continuously refine the object's model as the robot handles it, allowing for better integration of the identification procedure into the handling operations. Additionally, expanding on our idea of integrating vision and force sensing through part segmentation, visual material recognition (e.g., wood, aluminium, plastic) could be used to provide an initial guess for the object's mass density distribution. This, in turn, would enable estimating the coefficients of friction when the object is placed in contact with other objects, which will be of crucial importance in the next chapter.

Chapter 4

Robustness Assessment of Assemblies in Frictional Contact

> Nature is thrifty in all its actions.
>
> — Pierre Louis Moreau de Maupertuis, 1744.

While Chapter 3 introduced methods for acquiring object models, we now rely on such models to assess the robustness of multi-object assemblies to external forces. The algorithms presented in this chapter lay the foundation for the object placement planner described in Chapter 5, which is henceforth referred to as (Nadeau and Kelly, 2025). Although (Nadeau and Kelly, 2025) precedes the work presented here in terms of publication date, it is logical for this thesis to first describe the robustness assessment algorithm before introducing the placement planner.

Our physically-grounded robustness assessment algorithm handles arbitrary structures made from rigid objects of any shape and mass distribution without relying on heuristics or approximations. The result is a method that provides a foundation for autonomous robot decision-making when interacting with objects in frictional contact. Our strategy relies on a contact interface graph representation to reason about instabilities and decouple sub-problems by making use of object shape information to improve efficiency. In practice, our algorithms can be used for assemblies of objects for which CAD models are available or have been reconstructed from sensor data. Compared to existing methods, our approach is more accurate and much faster, executing in less than 1 second in most practical scenarios. Although our method assumes object models to be exact and although running time grows quickly with the number of objects, we provide simple instructions to incorporate safety margins and to reduce compute time if needed.

4.1 Motivation

Assessing the capacity of multi-object assemblies to withstand external forces without becoming unstable is crucial for enabling robots to make informed decisions in real-world environments, where autonomy hinges on accurate and reliable world models. Mobile robots efficiently traversing cluttered environments (Heins and Schoellig, 2023; Flores and Kecskeméthy, 2013), robotic manipulators planning grasps (Omata and Nagata, 2000; Maeda et al., 2007; Bicchi and Kumar, 2000) or placing objects (Wang and Hauser, 2021; Srinivas et al., 2023; Jiang et al., 2012), autonomous construction robots (Wermelinger et al., 2021; Johns et al., 2023), and robots in many other contexts all need to answer the question:

What forces can assemblies of objects in frictional contact withstand before becoming unstable?

Answering this question accurately enables a robot to safely interact with its surroundings and make informed decisions when handling and manipulating objects.

This work contributes a novel method for evaluating the *robustness* of rigid object assemblies, that is, their capacity to withstand external forces without moving, as pictured in Figure 4.1. We also describe how our method can be useful in industrial object handling applications, with experimental results demonstrating its practical effectiveness. Due to the interplay of objects in contact, assessing the robustness of an assembly is complex and depends on the shapes, inertial parameters, frictional properties, and poses of all objects. Although heuristic methods can suffice in select applications (Wang and Hauser, 2019; Wang et al., 2010), a more general approach is needed to enable robots to handle diverse objects in a wide variety of scenarios. Previous efforts have resorted to computationally expensive methods, like dynamics simulations, to assess the robustness of assemblies (Lee et al., 2023; Chen et al., 2021; Saxena and Likhachev, 2023). In contrast, this work proposes a method producing an accurate solution for any rigid object assembly and does so much faster than existing approaches.

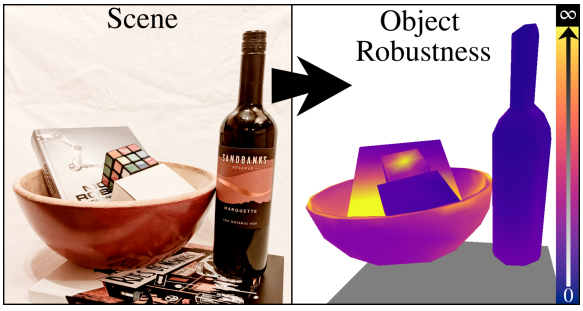


Figure 4.1: Using our method, a robot can assess the robustness of objects in complex scenes (lighter colours indicates higher robustness), allowing motion planners to avoid areas close to insecure objects (e.g., the wine bottle) and object placement planners to identify stable spots (e.g., the bottom of the book).

Given the poses, inertial parameters, and friction coefficients of known objects in an assembly, our algorithm can assess the maximum force that can be sustained by the assembly before becoming unstable. We validate our method against manually computed theoretical solutions and benchmark it against simulation-based and optimization-based approaches in terms of execution time and accuracy. We believe that our solution to the aforementioned problem can provide a foundation for autonomous decision-making when interacting with objects in frictional contact.

To highlight potential applications of our method, we present three key use cases (two of which are illustrated in Figure 4.2): stable object placement planning, safe assembly transportation, and disassembly planning. In the first application, we show that an existing stable object placement planner (Nadeau and Kelly, 2025) can produce better placements when relying on our proposed method. In the second application, we demonstrate how our method can be used to determine the acceleration that a mobile manipulator should not exceed when transporting an assembly of objects. Finally, we also expose how our method can be used to define disassembly sequences, enabling a robot manipulator to remove objects from an assembly without destabilizing it. Overall, our proposed method aims at improving the autonomy of mobile manipulators in industrial object handling tasks involving disassembling, transporting, and placing objects safely.

The remainder of this chapter is organized as follows. In Section 4.2, we highlight prior work on friction modeling, contact force resolution, and assembly stability assessment. The optimization problem used to compute contact forces is detailed in Section 4.3, and our proposed method for assessing the robustness of an assembly is presented in Section 4.4. A theoretical validation of our method is provided in Section 4.5, where we also benchmark three existing methods against ground truth solutions. In Section 4.6, we highlight three key applications of our method: stable placement planning, safe object transportation, and disassembly planning. Finally, an outlook on sources of errors, advantages, and limitations of our work is provided in Section 4.7.

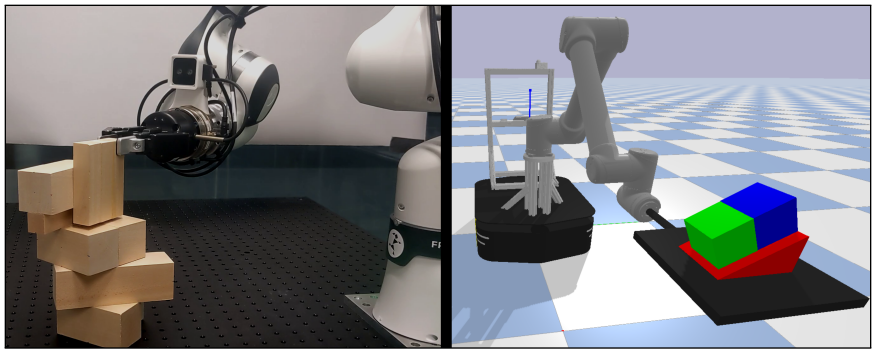


Figure 4.2: Two key applications of our method: object placement planning (Nadeau and Kelly, 2025) (left) and safe transportation (Heins and Schoellig, 2023) (right).

4.2 Related Work

4.2.1 Rigid Objects in Frictional Contact

Assuming that discrete contact points constitute the interface between two objects, the onset of slipping — which leads to instability — is governed by local forces at discrete contact points. For rigid objects in the elastic regime, (Shigley, 1972) and (Sinha and Abel, 1990) conclude that local forces at contact points on an interface between two objects are cumulative. They also show that the object deformation under load is such that the total potential energy of the system is minimized. Furthermore, results from (Otsuki and Matsukawa, 2013) indicate that, although the local stress at contact points can be momentarily released, all contact points are maximally stressed at the onset of bulk slipping. In (Bhushan, 2013), friction is described not as a material characteristic but as a systems response that is conditioned, in part, by (i) asperities in the contact surface (e.g., ploughing), (ii) adhesion between surfaces (e.g., chemical bonding), and (iii) lubrication of the contact surface (allowing a layer of atoms to slide relative to a deeper layer). Although friction depends on factors including contact time and temperature, experiments in (Bhushan, 1998) suggest that the tangential force required to trigger slipping is reasonably well approximated by the Coulomb friction model (Coulomb, 1821) for rigid objects in the elastic regime. Our proposed algorithm makes use of the findings from (Sinha and Abel, 1990) and (Otsuki and Matsukawa, 2013) to model force interactions for objects with multiple contact points. Also, we base our approach on the Coulomb friction model to make it practical in common robot applications.

For an assembly to be stable, all objects must be under force equilibrium; and enforcing this condition necessitates computing forces at the contact points. However, (Mattikalli et al., 1996) highlights that the equilibrium condition is, in general, insufficient to guarantee stability due to indeterminacies in the contact forces. Furthermore, (Mason, 1986) indicates that since the deformation of an object (assumed to be rigid) is much smaller than the uncertainty in the object's shape, the location of contact points is effectively indeterminate in practice. In this work, we avoid static indeterminacies by relaxing the assumption that objects are perfectly rigid, an approach that is coherent with the principle of virtual work (Sinha and Abel, 1990). The optimization problem we use to determine contact forces is also central to methods used in the computer-aided design of rigid structures (Whiting et al., 2012; Kao et al., 2022; Yao et al., 2017), and is equivalent to the solution obtained with finite-element methods (Shin et al., 2016).

Rigid-body dynamics for multi-object simulations is reviewed in (Stewart, 2000; Liu and Negrut, 2021), and a survey of techniques and challenges related to grasping and fixturing is presented in (Bicchi and Kumar, 2000).

4.2.2 Stability Assessment

Assessing the stability of objects in contact is useful in a breadth of applications like grasp planning (Omata and Nagata, 2000; Maeda et al., 2007; Bicchi and Kumar, 2000; Wermelinger et al., 2021), automated fixturing (Sugar and Kumar, 2000), object placement planning (Wang and Hauser, 2021; Srinivas et al., 2023), computer-aided design (Whiting et al., 2012; Kao et al., 2022), object transportation (Heins and Schoellig, 2023; Flores and Kecskeméthy, 2013). While some methods focus on a single object (e.g., planning how to grasp or fixture a given object), the algorithm proposed in this work can be used to assess the stability of a multi-object assembly and compute its robustness to external forces.

Abstracting away from contact forces, (Mojtahedzadeh et al., 2015) considers support relationships between objects to facilitate convex object placement planning. In turn, support relationships are determined through the use of geometric heuristics in (Kartmann et al., 2018) that do not consider inertial parameters or friction coefficients. As a result, the applicability of these methods are limited to simple assemblies for which heuristics can be defined. In contrast, our approach can be applied to any assembly of rigid objects.

Our method relies on a graph representation of the assembly, similar to the one used in (Lee and Moradi, 1999; Lee and Yi, 1994) to plan disassembly sequences. The graph representation in (Boneschanscher et al., 1988) uses edges to represent the set of forces that an object can exert onto another without making it move. However, since the set of forces that object A can exert onto object B is not equal to the set of forces that B can exert onto A, (Boneschanscher et al., 1988) is limited to assemblies that can be described with directed acyclic graphs (i.e. without loops). In contrast, the nature of the graph used in our work makes our proposed algorithm applicable to any assembly of rigid objects.

The assembly robustness criterion proposed in (Maeda et al., 2009) is defined as an optimization problem in which the magnitude of a given external wrench is maximized. For each external wrench query, (Maeda et al., 2009) tackles the full computation of contact forces and robustness assessment in a single optimization problem. Similarly, (Chen et al., 2021) performs stability analysis by solving an optimization problem for a large number of random force disturbances, resulting in large running times due to their stability assessment scheme. In contrast, our proposed approach decomposes the overall problem into subproblems and use object shape information to improve efficiency.

In (Nadeau and Kelly, 2025), an approximate robustness assessment method is used to find stable object placement poses much faster than with other methods. Compared to the heuristic used in (Nadeau and Kelly, 2025), the method defined in this work assesses the assembly robustness more accurately, albeit at a higher computational cost. In this work, we show that upgrading (Nadeau and Kelly, 2025) with our method can improve the robustness of the placement poses found and reduce the planning time when dealing with complex scenes. Furthermore, while (Nadeau and Kelly, 2025) only considered the capacity of an object to *withstand* external forces, this chapter proposes to also consider how external

forces might *improve* the overall assembly robustness.

4.3 Computing Contact Forces

In this work, we make the following assumptions:

1. Objects are almost perfectly rigid, with local deformations at discrete contact points only;
2. contact points are non-adhesive and deform elastically;
3. the Coulomb friction model is in effect; and
4. the shapes, inertial parameters, and coefficients of friction of all objects are known.

While these assumptions are commonly used in robotics (Lynch and Park, 2017; Mason, 2001; Boneschanscher et al., 1988), an extensive study of their practical applicability concluded that the Coulomb friction model is reasonably accurate but its coefficients vary across contact area (Yu et al., 2016). Expressed as an optimization problem, finding values for the contact forces in an assembly can yield no solutions, in which case the assembly is guaranteed to be unstable (Pang and Trinkle, 2000). Otherwise, a feasible set of contact forces stabilizing the assembly can be found.

4.3.1 Objective Function

For stiff materials, the force needed to slightly deform the contact points is governed by Young's modulus κ with $\|\mathbf{f}\| = \kappa x$, where x is the magnitude of the deformation. The energy stored through the deformation is given by

$$U = \int \|\mathbf{f}\|\, dx = \frac{\kappa x^2}{2} = \frac{\|\mathbf{f}\|^2}{2\kappa} \tag{4.1}$$

and the *principle of virtual work* dictates that an assembly will minimize this value (Sinha and Abel, 1990). Consequently, our objective function minimizes the sum of squared contact forces in the assembly.

4.3.2 Equilibrium Constraints

A necessary but insufficient condition for assembly stability (Mattikalli et al., 1996) is that all objects must be in a static equilibrium, which is achieved when the sum of all forces and torques acting on the object is zero. A contact force $\mathbf{f}$ at a contact point can be decomposed into a frictional component $\mathbf{f}_t$ and a normal component $\mathbf{f}_n$, with the frictional component further decomposed into two orthogonal components $\mathbf{f}_u$ and $\mathbf{f}_v$. A local reference frame can be defined as $\boldsymbol{\mathcal{F}}_i = \begin{bmatrix}\hat{\mathbf{u}} & \hat{\mathbf{v}} & \hat{\mathbf{n}}\end{bmatrix}$, where $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{n}}$ are the unit vectors of the local frame

with the inward surface normal given by $\hat{\mathbf{n}}$. The contact force at the contact point can be expressed as

$$\mathbf{f} = \mathbf{f}_t + \mathbf{f}_n = \begin{bmatrix} \mathbf{f}_u & \mathbf{f}_v & \mathbf{f}_n \end{bmatrix}^\mathsf{T} \tag{4.2}$$

in $\boldsymbol{\mathcal{F}}_i$, where $\mathbf{f}_t = \begin{bmatrix} \mathbf{f}_u & \mathbf{f}_v \end{bmatrix}^\mathsf{T} \in \mathcal{S}^1$ is the tangential force and $\mathbf{f}_n$ is the normal force. The contact force can be expressed in the global (i.e. world, inertial) frame $\boldsymbol{\mathcal{F}}_w$ as

$${}_w\mathbf{f}_i = {}_w\mathbf{R}_i\mathbf{f} \tag{4.3}$$

where ${}_w\mathbf{R}_i$ is the rotation matrix expressing the orientation of $\boldsymbol{\mathcal{F}}_i$ with respect to $\boldsymbol{\mathcal{F}}_w$. The wrench exerted by ${}_w\mathbf{f}_i$ on the object, and expressed in $\boldsymbol{\mathcal{F}}_w$, is

$${}_w\mathbf{w}_i = \underbrace{\begin{bmatrix} \mathbf{1}_3 \\ [{}_w\mathbf{p}_i]_\times \end{bmatrix}}_{\mathbf{A}_i} {}_w\mathbf{f}_i = \underbrace{\begin{bmatrix} \mathbf{1}_3 \\ [{}_w\mathbf{p}_i]_\times \end{bmatrix} {}_w\mathbf{R}_i}_{\mathbf{B}_i} \begin{bmatrix} \mathbf{f}_u \\ \mathbf{f}_v \\ \mathbf{f}_n \end{bmatrix} \tag{4.4}$$

where ${}_w\mathbf{p}_i$ is the position of the i-th contact point relative to $\boldsymbol{\mathcal{F}}_w$. The equilibrium of the jth object can be expressed as

$$\sum_k^{|K_j|} \mathbf{B}_k\mathbf{f}_k + {}_w\mathbf{w}_{g_j} = \mathbf{0}, \tag{4.5}$$

where ${}_w\mathbf{w}_{g_j}$ denotes the wrench due to the object being accelerated (e.g., by gravity) and K_j is the set of contact points on the object. In constant velocity or at rest,

$${}_w\mathbf{w}_g = m \begin{bmatrix} \mathbf{1}_3 & -[{}_w\mathbf{p}_c]_\times \end{bmatrix}^\mathsf{T} {}_w\ddot{\mathbf{p}}_g \tag{4.6}$$

where m is the object mass, ${}_w\ddot{\mathbf{p}}_g$ is gravity, and ${}_w\mathbf{p}_c$ is the position of the center of mass of the object.

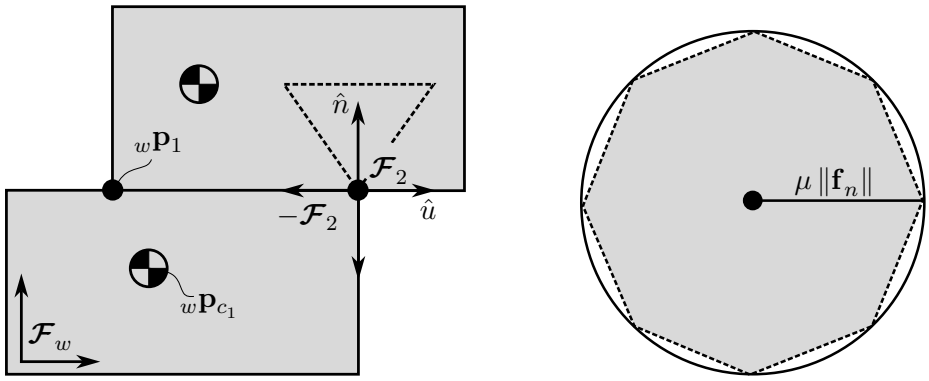
4.3.3 Friction Magnitude Constraints

The Coulomb friction model (Coulomb, 1821; Mason, 1986) states that, for a contact point to avoid slipping,

$$\|\mathbf{f}_t\| \leq \mu \|\mathbf{f}_n\| \tag{4.7}$$

where $\mu \in \mathbb{R}_+$ is the coefficient of friction, $\mathbf{f}_n$ is the normal force, and $\mathbf{f}_t$ is the tangential force at the contact point, as shown in Figure 4.3. Hence, in $\boldsymbol{\mathcal{F}}_i$, the friction force must lie within a circle whose radius is $\mu \|\mathbf{f}_n\|$. The inequality in (4.7) can be rewritten as the equality

$$c = \mu \|\mathbf{f}_n\| - \|\mathbf{f}_t\|, \tag{4.8}$$



(a) (b)

Figure 4.3: (a) Illustration of the notation used in this work. (b) Octagonal approximation to the Coulomb circle.

where $c \in \mathbb{R}_+$ is the *contact condition*, that can be interpreted as expressing how far a contact point is from slipping.

The Coulomb circle is commonly approximated (Mattikalli et al., 1996; Stewart and Trinkle, 2000) with an inscribed polygon, such as a square or an octagon, as shown in Figure 4.4. While an inscribed square covers only about 65% of the Coulomb circle, an inscribed octagon covers about 90%. In any case, the inequality

$$\mathbf{C} \begin{bmatrix} \mathbf{f}_u & \mathbf{f}_v & \mathbf{f}_n \end{bmatrix}^\mathsf{T} \leq \mathbf{0}_{N\times 1}, \tag{4.9}$$

where $\mathbf{C}$ is a $N \times 3$ matrix whose k-th row is given by

$$\mathbf{C}\,[k, 1:3] = \begin{bmatrix} \cos\frac{2\pi k}{N} & \sin\frac{2\pi k}{N} & -\mu\cos\frac{\pi}{N} \end{bmatrix}, \tag{4.10}$$

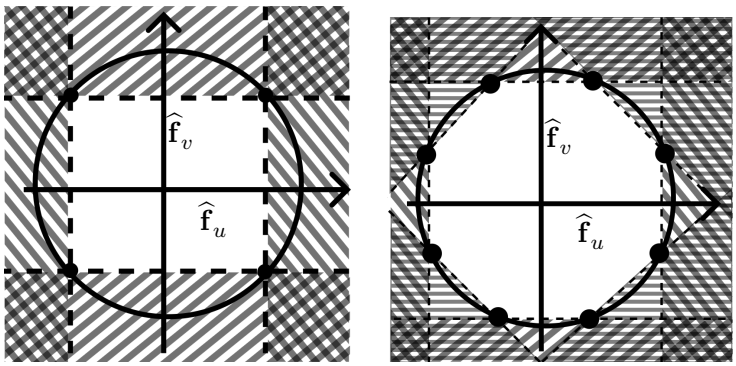
constrain the friction force to lie within an inscribed N-sided regular polygon, as shown in Figure 4.4.

4.3.4 Unilaterality Constraints

The assumption that contact points are non-adhesive can be enforced with

$$\mathbf{f}_n \geq 0 \tag{4.11}$$

assuming that $\hat{\mathbf{n}}$, along which $\mathbf{f}_n$ is directed, is the inward normal of the object surface at the contact point.



(a) (b)

Figure 4.4: Friction circle linearization with (a) $N = 4$ and (b) $N = 8$ by defining N half-plane constraints per contact point.

4.3.5 Optimization Problem

Given a set of J objects, let I be the set of all contact points across all objects and K_j be the set of contact points on object $j \in J$. The complete formulation of the optimization problem for computing contact forces in an assembly is

$$\min_{\mathbf{f}_i} \quad \sum_{i}^{|I|} \|\mathbf{f}_i\|^2 \tag{4.12}$$

subject to

$$\sum_{k}^{|K_j|} \mathbf{B}_k \mathbf{f}_k + {}_w\mathbf{w}_{g_j} = \mathbf{0} \qquad \forall j \in J \tag{4.13}$$

$$\mathbf{C}_i \mathbf{f}_i \leq \mathbf{0} \qquad \forall i \in I \tag{4.14}$$

$$\mathbf{f}_{i_n} \geq 0 \qquad \forall i \in I \tag{4.15}$$

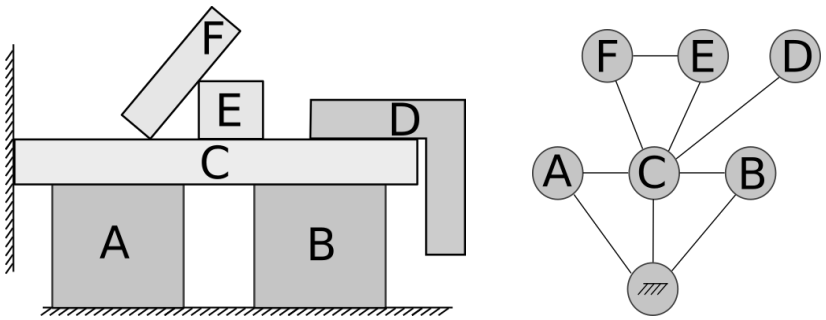
representing a quadratic program with linear constraints that can be efficiently solved with standard methods (Stellato et al., 2020).

4.4 Robustness Computation

In this work, we define *robustness* as the amount of force that can be applied in a given direction on a given point before relative motion occurs between objects in an assembly.

Definition 5. *With $\mathbb{R}$ being the set of real numbers and $\mathbb{S}^2$ being the unit sphere, robustness is formally defined as*

$$\mathrm{R} : \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}_+, \tag{4.16}$$



(a) (b)

Figure 4.5: (a) A structure composed of six non-fixed objects and two fixed objects (i.e. floor and wall). (b) The contact interface graph of the structure with the fixed nodes at the bottom.

which maps a pair $({}_w\mathbf{p}_i \in \mathbb{R}^3 \ , \ {}_w\hat{\mathbf{e}}_i \in \mathbb{S}^2)$ *to the maximum force magnitude* $\mathrm{R} \in \mathbb{R}_+$ *that can be exerted on point* ${}_w\mathbf{p}_i$ *in the direction* ${}_w\hat{\mathbf{e}}_i$ *before the onset of motion occurs, and where* $\mathbb{S}^2$ *is the unit sphere.*

In turn, this work also proposes a method to determine how applying an external force can improve or reduce the robustness of the assembly.

Definition 6. *The robustness improvement is the difference in robustness that results from applying an external force on a given point in a given direction, defined as*

$$\mathrm{RI} : \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}, \tag{4.17}$$

which maps a pair $({}_w\mathbf{p}_i \in \mathbb{R}^3 \ , \ {}_w\hat{\mathbf{e}}_i \in \mathbb{S}^2)$ *to the difference* $\mathrm{RI} \in \mathbb{R}$ *in robustness that results from applying an external force on point* ${}_w\mathbf{p}_i$ *in the direction* ${}_w\hat{\mathbf{e}}_i$.

4.4.1 Contact Interface Graph

An arbitrary assembly of rigid objects can be described with a *contact interface graph* (CIG), in which each node represents an object in the assembly and each edge represents a contact interface (Siciliano et al., 2008, Chapter 37) between two objects, as shown in Figure 4.5. Describing assemblies with this type of graph can be traced back to Lee (1994).

The information contained in the CIG can be stored as a list of interfaces, each defined by the two objects it connects, the locations of the contact points on the interface, and the forces at the contact points. To simplify the graph, nodes representing fixed objects (e.g., a floor, a wall) can be merged into a single node marked as being *fixed*. Since no object can be free-floating, the resulting graph is guaranteed to be connected such that a path exists from any node to the fixed node, and a *support relations graph* (Mojtahedzadeh et al., 2015) can easily be derived from the CIG. A *directional force graph* (Lee and Moradi,

1999) describing the forces required to extract an object from a fixtured assembly can also be derived from the CIG. Similar to the approach taken in (Boneschanscher et al., 1988), the CIG can be iteratively simplified through edge contraction to yield an equivalent CIG. However, in contrast to (Boneschanscher et al., 1988), edges in the CIG are undirected and the contraction operations are commutative, such that the presence of cycles in the graph is not a problem.

When a force is exerted on a node, all interfaces lying on simple paths between the node and the fixed node will incur additional stress since (4.1) dictates that stress should be spread as much as possible. Deformations occur when an object is constrained by a force and the environment (e.g., an object falling down under gravitational force will deform only when hitting the ground). Hence, objects not lying on the aforementioned paths are not further constrained by the applied force and will not incur additional stress.

Modeling Instabilities as Graph Cuts

At the onset of instability, a subset of objects in the assembly will move relative to the others. This phenomenon can be modelled as a graph cut of the CIG, where edges cut represent contact interfaces at which relative motion between objects happens. The relative motion can be translational or rotational, as illustrated in Figure 4.6. The former is associated with a slipping phenomenon while the latter is associated with a toppling phenomenon. While slipping occurs when friction constraints are violated, toppling happens when an external force disturbs the equilibrium of the assembly. The minimal force producing a given cut will either generate slipping or toppling, since a strictly greater force would be required to generate a toppling motion while objects are slipping. Crucially, this enables dividing the overall problem into two separate sub-problems: computing the robustness to slipping, and computing the robustness to toppling.

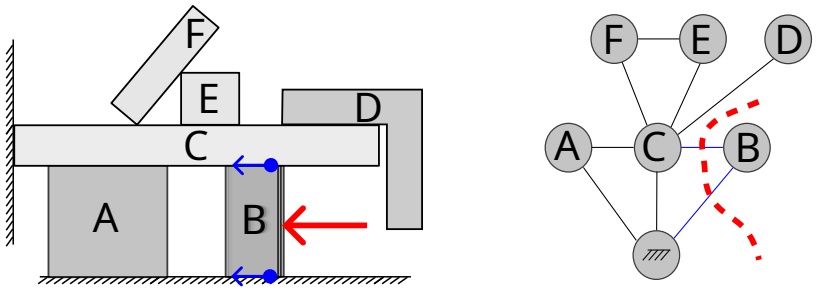
The overall robustness to a force exerted in direction ${}_w\hat{\mathbf{e}}_i$ on a point ${}_w\mathbf{p}_i$ is given by

$$\mathrm{R}\left({}_w\mathbf{p}_i, {}_w\hat{\mathbf{e}}_i\right) = \min\left(\mathrm{R}_{slip}\left({}_w\mathbf{p}_i, {}_w\hat{\mathbf{e}}_i\right), \mathrm{R}_{top}\left({}_w\mathbf{p}_i, {}_w\hat{\mathbf{e}}_i\right)\right) \tag{4.18}$$

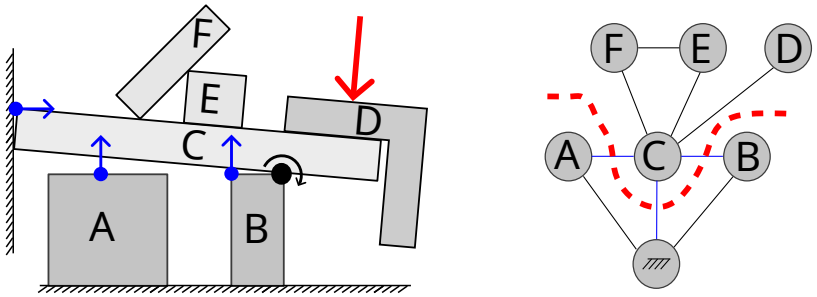
where $\mathrm{R}_{slip}(\cdot)$ and $\mathrm{R}_{top}(\cdot)$ are respectively the robustness to slipping and toppling. A key strategy in our approach is to deal with slipping and toppling separately, as they are fundamentally different phenomena, before combining their results.

4.4.2 Robustness to Slipping

In the following, we first determine the robustness to an external force when considering a single contact point. Then, we define how the robustness of several contact points on a common interface are combined. Finally, we explain how the robustness to slipping for a complete assembly can be obtained.



(a) Translational relative motion



(b) Rotational relative motion

Figure 4.6: Translational (a) and rotational (b) relative motions at the contact interfaces (small blue arrows) due to external forces (large red arrows). Both phenomena are modelled as a graph cut of the CIG, where edges cut represent contact interfaces at which relative motion happens.

At a Single Contact Point

Let the contact force $\mathbf{f}$ at a given contact point be expressed in a local frame as $\mathbf{f} = \begin{bmatrix}\mathbf{f}_u & \mathbf{f}_v & \mathbf{f}_n\end{bmatrix}^\mathsf{T}$ where $\mathbf{f}_t = \begin{bmatrix}\mathbf{f}_u & \mathbf{f}_v\end{bmatrix}^\mathsf{T} \in \mathcal{S}^1$ is the tangential force and $\mathbf{f}_n$ is the normal force, as pictured in Figure 4.3. Also, let an external force applied at the contact point be expressed in the local frame as $\mathbf{e} = s\hat{\mathbf{e}} = \begin{bmatrix}\mathbf{e}_u & \mathbf{e}_v & \mathbf{e}_n\end{bmatrix}^\mathsf{T}$ where $s \in \mathbb{R}_+$ is the magnitude of the external force and $\hat{\mathbf{e}} \in \mathbb{S}^2$ is its direction. We define the *contact condition* of the point as

$$c(s) = \mu \left\|\mathbf{f}_n + s\hat{\mathbf{e}}_n\right\| - \left\|\mathbf{f}_t + s\hat{\mathbf{e}}_t\right\| \tag{4.19}$$

that equals zero when

$$\mu \left\|\mathbf{f}_n + s\hat{\mathbf{e}}_n\right\| = \left\|\begin{bmatrix}\mathbf{f}_u + s\hat{\mathbf{e}}_u & \mathbf{f}_v + s\hat{\mathbf{e}}_v\end{bmatrix}\right\|, \tag{4.20}$$

at which point the contact point is on the verge of slipping. Intuitively, the contact condition is the amount of force that can be withstood in a given direction before slipping happens. Equation (4.20) is quadratic in s with solutions given by

$$s_m = \frac{\begin{bmatrix}\mathbf{f}_u & \mathbf{f}_v & -\mu^2\mathbf{f}_n\end{bmatrix}\hat{\mathbf{e}} - n}{d} \qquad \text{and} \qquad s_p = s_m + \frac{2n}{d} \tag{4.21}$$

where

$$n = \left(\mu^2 \left(\hat{\mathbf{e}}_n \mathbf{f}_u - \hat{\mathbf{e}}_u \mathbf{f}_n\right)^2 + \mu^2 \left(\hat{\mathbf{e}}_n \mathbf{f}_v - \hat{\mathbf{e}}_v \mathbf{f}_n\right)^2 - \left(\hat{\mathbf{e}}_v \mathbf{f}_u - \hat{\mathbf{e}}_u \mathbf{f}_v\right)^2\right)^{\frac{1}{2}} \quad \text{and} \tag{4.22}$$

$$d = \mu^2 \left\|\hat{\mathbf{e}}_n\right\|^2 - \left\|\hat{\mathbf{e}}_t\right\|^2 \tag{4.23}$$

are used to simplify the notation. The solution in (4.21) can be shown to be equivalent to the geometric problem of finding intersection points between an external force vector and the friction cone at a contact point, as derived in (Nadeau and Kelly, 2025). The curvature of the equation in (4.20) is given by d such that

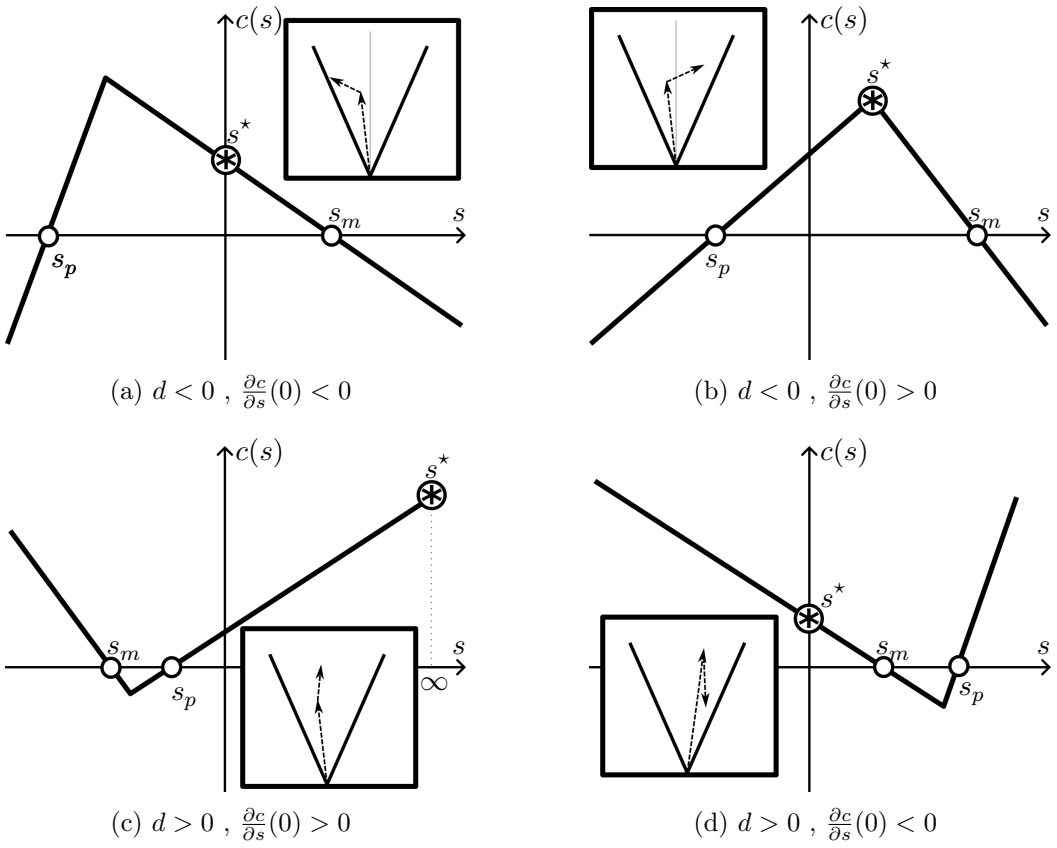
$$s_m = \frac{\left\|\mathbf{f}_t\right\|^2 - \mu^2 \left\|\mathbf{f}_n\right\|^2}{2\left(\mu^2 \left\|\hat{\mathbf{e}}_n\right\| \left\|\mathbf{f}_n\right\| - \hat{\mathbf{e}}_t^{\mathsf{T}} \mathbf{f}_t\right)} \quad \text{when} \quad d = 0 \tag{4.24}$$

is the solution to the linear equation obtained when $\mu^2 \left\|\hat{\mathbf{e}}_n\right\|^2 = \left\|\hat{\mathbf{e}}_t\right\|^2$. Assuming that $c(0) \geq 0$ (i.e. the assembly is stable when no external force is applied), four situations can be considered depending on the sign of d and on the sign of $\partial c / \partial s$ at $s = 0$. The four situations are shown in Figure 4.7, with inset figures representing the friction cone with contact and external force directions at the contact point. When positive, the solution s_m represents the maximum amount of force that can be exerted on the contact point in the direction $\hat{\mathbf{e}}$ before slipping occurs (s_p could equivalently be used for this purpose). When $s_m < 0$, an infinite amount of force can be exerted along $\hat{\mathbf{e}}$ without slip occurring. Hence, we define the robustness of the kth contact point to slipping when ${}_w\hat{\mathbf{e}}_i$ is exerted on point ${}_w\mathbf{p}_i$ as

$$\mathrm{R}_{k_{slip}}({}_w\mathbf{p}_i \ , \ {}_w\hat{\mathbf{e}}_i) = \begin{cases} s_m & \text{if } s_m \geq 0 \\ \infty & \text{if } s_m < 0 \end{cases}. \tag{4.25}$$

The situation depicted in Figure 4.7a represents the case where any non-zero external force (starting from $s = 0$) will reduce $c(s)$ (the contact condition). In Figure 4.7b, increasing the magnitude of the external force will first improve the contact condition by bringing the sum of forces closer to the axis of the friction cone, but increasing it further will monotonically decrease the contact condition. In Figure 4.7c, the external force is inside the friction cone at the contact point, and any amount of force will improve the contact condition. In Figure 4.7d, the external force is inside the cone that is opposed to the friction cone and increasing the magnitude of the external force will decrease the contact condition until $c(s) = 0$.

Taking the derivative of the contact condition in (4.19) with respect to the magnitude



(a) $d < 0$, $\frac{\partial c}{\partial s}(0) < 0$

(b) $d < 0$, $\frac{\partial c}{\partial s}(0) > 0$

(c) $d > 0$, $\frac{\partial c}{\partial s}(0) > 0$

(d) $d > 0$, $\frac{\partial c}{\partial s}(0) < 0$

Figure 4.7: The contact condition as a function of the external force magnitude. The roots of $c(s)$ are s_m and s_p while $s^\star$ is the magnitude that should be exerted to maximize c. Inset figures represent the friction cone with the external force starting at the tip of the contact force vector.

of the external force yields

$$\frac{\partial c}{\partial s} = \mu \hat{\mathbf{e}}_n - \frac{\hat{\mathbf{e}}_u \left(s\hat{\mathbf{e}}_u + \mathbf{f}_u\right) + \hat{\mathbf{e}}_v \left(s\hat{\mathbf{e}}_v + \mathbf{f}_v\right)}{\sqrt{\left(s\hat{\mathbf{e}}_u + \mathbf{f}_u\right)^2 + \left(s\hat{\mathbf{e}}_v + \mathbf{f}_v\right)^2}} \tag{4.26}$$

$$= \mu \hat{\mathbf{e}}_n - \frac{\hat{\mathbf{e}}_t^{\mathsf{T}} \left(\mathbf{f}_t + s\hat{\mathbf{e}}_t\right)}{\|\mathbf{f}_t + s\hat{\mathbf{e}}_t\|} \tag{4.27}$$

where the rightmost term can be interpreted as the projection of the tangential external force vector onto the total tangential force vector. From (4.27), the contact condition will be improved if the projection of the external force onto the contact point normal is above a threshold value determined by the coefficient of friction.

At a Contact Interface

Assuming that objects obey the laws of linear elasticity at the contact points, the stress incurred at each contact point is accumulated to produce the total frictional force (Sinha and Abel, 1990; Bhushan, 1998). Locally, the stress on a contact point can be momentarily released with another one taking over the load, but the gross effect is that the total frictional force is the sum of the frictional forces at each contact point (Otsuki and Matsukawa, 2013). This is consistent with the *principle of least action*: the system will naturally minimize its energy by distributing the forces as uniformly as possible on contact points, each respecting its traction limit. Consequently, the robustness of a contact interface t to an external force is given by

$$\mathrm{R}_{t_{slip}} \left(\mathbf{p}_i, \hat{\mathbf{e}}\right) = \sum_{k}^{I_t} \mathrm{R}_{k_{slip}} \left(\mathbf{p}_i, \hat{\mathbf{e}}\right), \tag{4.28}$$

where I_t is the set of contact points on the interface.

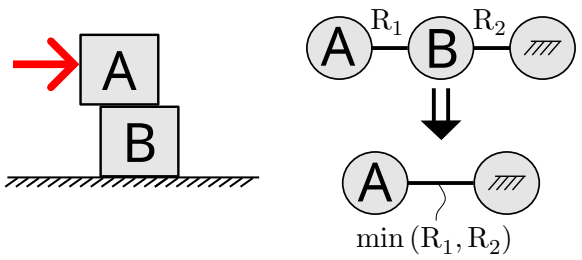
For a Complete Assembly

In general, an object will be in contact with several other objects through contact interfaces that have different maximal frictional forces (i.e. traction limit). For the reasons outlined in Section 4.4.2, we state the following assertion:

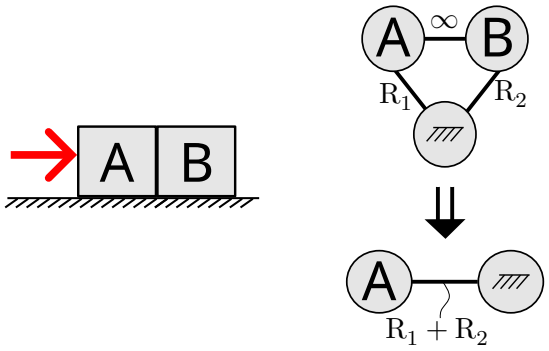
> **Assertion 1.** *The slipping robustness of several interfaces acting in parallel is given by the sum of the individual interface robustnesses.*

An example of this situation is shown in Figure 4.8b, where the total slipping robustness is given by the sum of the two interface robustnesses acting in parallel.

Furthermore, when two objects are in contact, the principle of action-reaction (i.e. Newton's third law) implies that the force acting on an interface is exerted onto both objects (with opposite directions). Hence, the maximum force that can be exerted at the interface by either object is given by the minimum traction of the two objects. In other words, in



(a) Interfaces in series



(b) Interfaces in parallel

Figure 4.8: Combining interfaces acting (a) in series, and (b) in parallel to resist an external force (red arrow). The individual interface robustnesses R_1 and R_2 are combined $\min(\cdot,\cdot)$ and $+(\cdot,\cdot)$ operators to combine interfaces in series and in parallel, respectively.

regards to slipping, a chain of objects will only be as strong as the weakest interface along the chain. This leads to the following assertion:

Assertion 2. *The slipping robustness of several interfaces acting in series is given by the minimum of the interface robustnesses along the series.*

An example of this situation is shown in Figure 4.8a, where the total slipping robustness is given by the minimum of the two interface robustnesses acting in series.

Given Assertion 1 and Assertion 2, the problem of determining the slipping robustness of a target object in an assembly can be cast as a *maximum flow* problem (Hillier and Lieberman, 2015). This is accomplished by defining a force flow network $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes in the CIG and $\mathcal{E}$ is the set of edges. In G,

1. the node associated with the object onto which the external force is exerted is labeled as the *source*;
2. the node associated with the fixed object is labeled as the *sink*; and
3. each edge is given a *capacity* equal to the corresponding interface's robustness, as computed with (4.28).

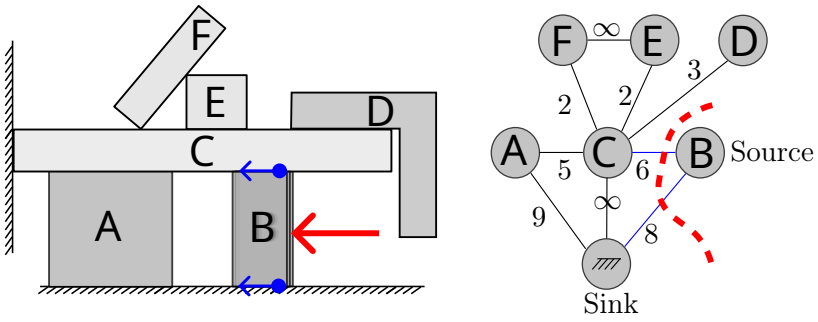


Figure 4.9: The subset of objects that would slip under the exertion of the external force (large arrow) is given by the minimum cut (dashed line) of the CIG. The robustness of the assembly to the external force applied on the *source* object is given by the maximum flow going into the *sink*. Edge *capacity* is given by the associated interface robustness.

The graph cut minimizing the total capacity of the edges cut delineates the subset of objects that would slip under the exertion of the external force, as shown in Figure 4.9. Moreover, the slipping robustness of the assembly to the external force is given by the *maximum flow* that can be routed from the source to the sink. Provided that a maximum flow exists, the slipping robustness of the assembly is given by

$$\mathrm{R}_{slip}(\mathbf{p}_i, \hat{\mathbf{e}}) = \sum \mathrm{flow}\left(\mathrm{e}\left[\mathrm{v}, \mathrm{sink}\right]\right) \qquad \forall\ \mathrm{v} \in \mathcal{V}, \tag{4.29}$$

where $e\left[v, \mathrm{sink}\right]$ represents the edge from node v to the sink if it exists, and flow(e) is the flow on edge e. Standard algorithms, whose worst-case running time is $O(|J|^3)$ (cubic in the number of objects) or better, can be used to compute the maximum flow and minimum cut simultaneously (Hillier and Lieberman, 2015). This type of network has been used in (Lee and Yi, 1994; Lee and Moradi, 1999) to study how clusters of parts in fixtured assemblies can be disassembled.

When planning the placement of an object, it can be beneficial to consider how the weight of the object can improve or reduce contact conditions in the assembly: the *slipping robustness improvement*. In multi-object assemblies, (4.41) can be computed for all interfaces lying on simple paths connecting the external force to the fixed object in the CIG, as described in Section 4.4.1, and summed to determine the global improvement in the object's robustness to toppling when the external force is applied. Since the contact condition of all contact points lying on simple paths between the source and the sink in the CIG will be influenced by the external force. Denoting the sub-graph produced from all simple paths between the source and the sink as $\mathcal{G}_s$, the total slipping robustness improvement is given by

$$\mathrm{RI}_{\mathrm{slip}}(\mathbf{p}_\mathrm{i}, \hat{\mathbf{e}}) = \sum_{\mathrm{I} \in \mathcal{G}_\mathrm{s}} \sum_{\mathrm{k} \in \mathrm{I}} \mathrm{RI}_{\mathrm{k}_{\mathrm{slip}}}(\mathbf{p}_\mathrm{i}, \hat{\mathbf{e}}), \tag{4.30}$$

where $\mathrm{RI}_{\mathrm{k_{slip}}}(\mathbf{p}_\mathrm{i}, \hat{\mathbf{e}})$ is given by (4.27).

4.4.3 Robustness to Toppling

While slipping occurs when an external force exceeds the traction of contact points in the assembly, toppling arises when the equilibrium of the assembly becomes impossible. In the following, we provide an algorithm to compute the toppling robustness of an assembly to external forces, that is, the maximal permissible magnitude that an external force applied at a specific point can have before a subset of objects in the assembly rotate about an axis. We start by defining feasible CIG cuts, and then determine axes about which motion can take place. The set of graph cuts and toppling axes is finally used to compute the robustness of the assembly to toppling.

Defining Feasible Graph Cuts

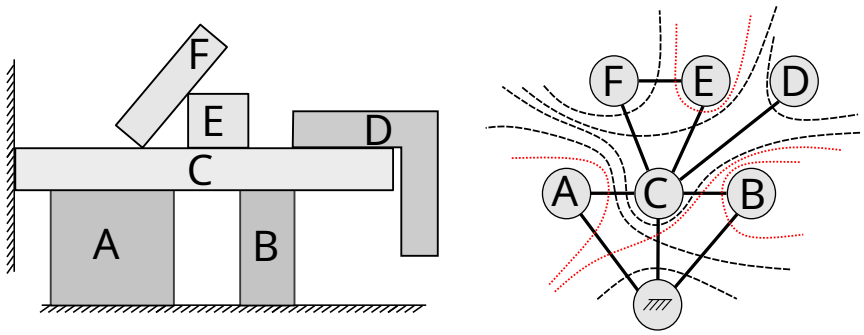
Similar to slipping, toppling can be modelled as a graph cut of the CIG, with the cut separating the objects that topple from those that remain stationary. Any valid cut separates the CIG into two sub-graphs, one containing the fixed object and the other containing the toppling objects. Some cuts are considered infeasible as they isolate a form-closed object subset (Bicchi, 1995) that cannot be toppled, as shown in Figure 4.10a. We define a *feasible cut* as a cut separating the fixed object from other objects that, together, are not form-closed. We denote the set of feasible cuts as $\mathcal{C}$ and the sub-graph (not containing the fixed node) produced by a cut $c \in \mathcal{C}$ as $\mathcal{G}_c$. The set of edges cut by $c \in \mathcal{C}$ is denoted as $\mathcal{E}_c$ and the set of interfaces cut by c is denoted as $\mathcal{T}_c$, as illustrated in Figure 4.10b. With the position of the i-th contact point being ${}_w\mathbf{p}_i$, and with the orientation of a frame ${}_w\mathbf{R}_i$ at the contact point being such that the z-axis is pointing into the super-object, define the grasp matrix $\mathbf{G}$ as

$$\mathbf{G}_i = \begin{bmatrix} {}_w\mathbf{R}_i & (-{}_w\mathbf{R}_i)_{[1:3,1:2]} \\ [{}_w\mathbf{p}_i]_\times \, {}_w\mathbf{R}_i & \left([{}_w\mathbf{p}_i]_\times \, {}_w\mathbf{R}_i\right)_{[1:3,1:2]} \end{bmatrix} \tag{4.31}$$

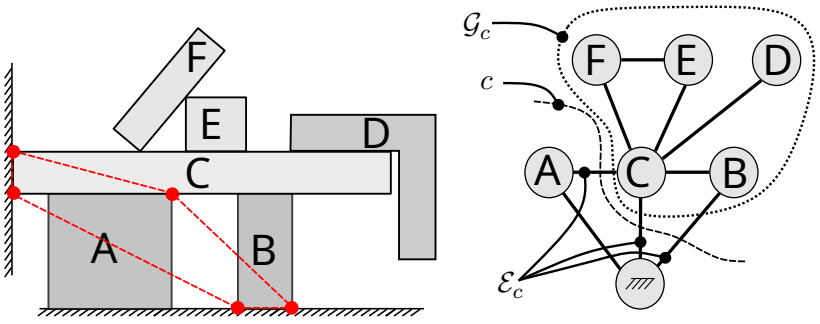
$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & \cdots & \mathbf{G}_i & \cdots & \mathbf{G}_I \end{bmatrix} \tag{4.32}$$

where each contact is modelled as a *hard-finger* (Siciliano et al., 2008), I is the number of contact points, and $\mathbf{X}_{[1:3,1:2]}$ is the submatrix built from the elements in the first three rows and first two columns of $\mathbf{X}$. The columns of $\mathbf{G}$ are the constraining wrenches exerted by the contact points. If $\mathrm{rank}(\mathbf{G}) < 6$, then the super-object is not form-closed (Bicchi, 1995) and a motion is possible. Otherwise, finding a solution to the linear program

$$\min_{\mathbf{x}} \quad \sum_i \mathbf{x}_i \quad \text{s.t.} \quad \mathbf{G}\mathbf{x} = \mathbf{0} \quad \text{and} \quad \mathbf{x}_i \geq 1 \tag{4.33}$$



(a) Cuts of the CIG



(b) Toppling Axes

Figure 4.10: (a) Graph cuts of an assembly's CIG. Red dotted lines indicate infeasible cuts that isolate a form-closed object subset. (b) The edges of the convex hull over contact points (left side) located on interfaces cut (right side) define potential toppling axes (dots indicate through-page axes).

determines that the super-object is form-closed, in which case it cannot topple about any axis and the next super-object can be immediately considered (Lynch and Park, 2017).

Defining Toppling Axes

For a given cut $c \in \mathcal{C}$, the edges of the convex hull over contact points $\{i \in I_t \; \forall \; t \in \mathcal{T}_c\}$ define potential toppling axes, as shown in Figure 4.10b. In the particular case where all $t \in \mathcal{T}_c$ are parallel, an additional axis is defined normal to the contact interfaces and passing through the centre of friction (Mason, 1986). We denote the set of all potential toppling axes as $\mathcal{A}_c$.

Defining Toppling Torque

For the ith contact point, we define the moment vector about toppling axis $a \in \mathcal{A}_c$ as

$$ {}_a\mathbf{v}_i = {}_{a_s}\mathbf{p}_i \times \hat{\mathbf{a}}, \tag{4.34} $$

where ${}_{a_s}\mathbf{p}_i$ is the position of the contact point relative to the toppling axis origin. The countering moment exerted by the contact point about the toppling axis is given by

$$ {}_a\tau_i = \begin{cases} \left(\mathrm{R}_{k_{slip}}({}_w\mathbf{p}_i, \hat{\mathbf{e}})\hat{\mathbf{e}} + {}_w\mathbf{f}_i\right) \cdot {}_a\mathbf{v}_i & \text{if } {}_a\mathbf{v}_i \cdot \hat{\mathbf{n}}_i = 0, \\ \infty & \text{if } {}_a\mathbf{v}_i \cdot \hat{\mathbf{n}}_i < 0, \\ {}_w\mathbf{f}_i \cdot {}_a\mathbf{v}_i & \text{if } {}_a\mathbf{v}_i \cdot \hat{\mathbf{n}}_i > 0, \end{cases} \tag{4.35} $$

where $\mathrm{R}_{k_{slip}}({}_w\mathbf{p}_i, \hat{\mathbf{e}})$ is computed from (4.25), $\hat{\mathbf{n}}_i$ is the surface normal at the contact point, and ${}_w\mathbf{f}_i$ is the contact force. When ${}_a\mathbf{v}_i \cdot \hat{\mathbf{n}}_i = 0$, the toppling axis is perpendicular to the contact interfaces and friction forces are counteracting the toppling motion, while when ${}_a\mathbf{v}_i \cdot \hat{\mathbf{n}}_i < 0$, the contact force kinematically prevents toppling about the axis. Otherwise, the contact force contributes to the equilibrium about the toppling axis and is used to compute the toppling torque.

For a given cut $c \in \mathcal{C}$ and a given toppling axis $a \in \mathcal{A}_c$, the total toppling torque exerted by contact points in I_t about a is given by

$$ {}_a\tau_{I_t} = \sum_{i \in I_t} {}_a\tau_i, \tag{4.36} $$

where I_t is the set of contact points on the interface $t \in \mathcal{T}_c$. The total toppling torque exerted about an axis $a \in \mathcal{A}_c$ is given by

$$ {}_a\tau_c = \sum_{t \in \mathcal{T}_c} {}_a\tau_{I_t}. \tag{4.37} $$

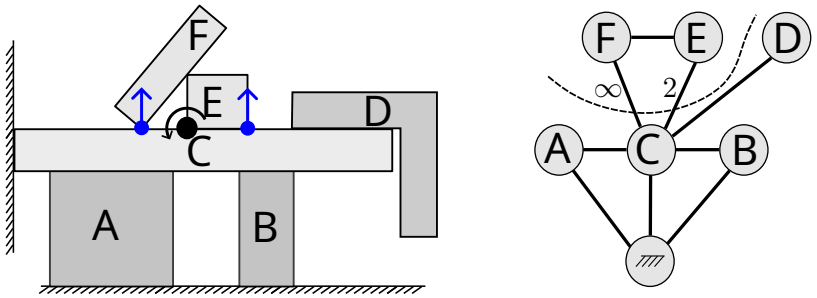
A toppling motion is deemed infeasible if ${}_a\tau_c$ is infinite, as is the case in Figure 4.11a. For feasible axes, each edge's toppling torque indicates the contribution of contact forces at the edge's interface in counteracting toppling, as shown in Figure 4.11b.

Defining Toppling Robustness

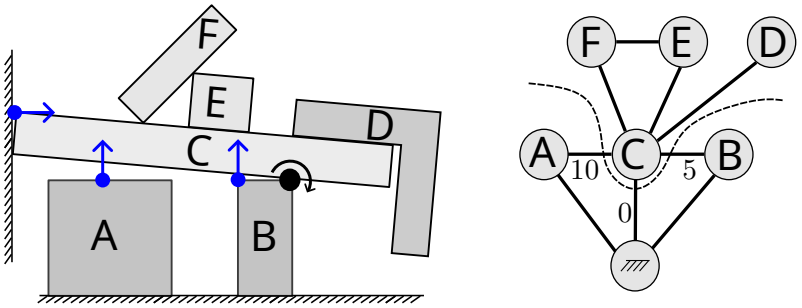
The toppling robustness for a set of point-direction tuples $\mathcal{P} = \{(\mathbf{p}_i, \hat{\mathbf{e}}_i)\}$ is computed with Algorithm 2. The initialization consists in setting the toppling robustness of each element of $\mathcal{P}$ to infinity, and in determining the feasible cut set $\mathcal{C}$ as described in Section 4.4.3. For each cut $c \in \mathcal{C}$, a set of toppling axes $\mathcal{A}_c$ over the interfaces that are cut is determined following Section 4.4.3. For each toppling axis $a \in \mathcal{A}_c$, the toppling torque ${}_a\tau_c$ is computed with (4.37) and the toppling robustness of every point in $\mathcal{P}$ that is also in $\mathcal{G}_c$ is updated with

$$ \mathrm{R}_{top}\left(\mathbf{p}_i, \hat{\mathbf{e}}_i\right) = \min \left\{ \mathrm{R}_{top}\left(\mathbf{p}_i, \hat{\mathbf{e}}_i\right), \frac{{}_a\tau_c}{\left({}^w_{a_s}\mathbf{p}_i \times {}_w\hat{\mathbf{e}}_i\right) \cdot \hat{\mathbf{a}}} \right\}. \tag{4.38} $$

The equation in (4.38) keeps the minimal toppling robustness, since the onset of instability will be triggered by the weakest force producing toppling.



(a) Infeasible toppling



(b) Feasible toppling

Figure 4.11: (a) A toppling axis (black dot) deemed infeasible due to one of the edges cut contributing an infinite countering moment (right side). (b) The total toppling torque of edges cut (here equals to 15) is the torque required to produce toppling about the axis.

Algorithm 2: Computing Toppling Robustness

input : CIG G, set of tuples $\mathcal{P} = \{(\mathbf{p}_i, \hat{\mathbf{e}}_i)\}$
output: toppling robustness $\mathrm{R}_{top}(\mathbf{p}_i, \hat{\mathbf{e}}_i)\ \forall\ i \in \mathcal{P}$
1. Initialize $\mathrm{R}_{top}(\mathbf{p}_i, \hat{\mathbf{e}}_i) \leftarrow \infty\ \forall\ i \in \mathcal{P}$
2. Determine the feasible cut set $\mathcal{C}$
foreach $c \in \mathcal{C}$ **do**
  foreach $a \in \mathcal{A}_c$ **do**
    3. Compute ${}_a\tau_c$ with (4.37)
    4. $\forall i \in \mathcal{P} \cap \mathcal{G}_c$: Update $\mathrm{R}_{top}(\mathbf{p}_i, \hat{\mathbf{e}}_i)$ with (4.38).

The worst-case running time of Algorithm 2 is $O\left(2^{|J|}\right)$, exponential in the number of objects in the assembly and proportional to the maximum number of cuts in the CIG. Although the average running time can be much lower in practice, the theoretical worst-case requires to consider axes between every object combinations.

At the onset of toppling,

$$\underbrace{s\left({}^{w}_{a_s}\mathbf{p}_i \times {}_{w}\hat{\mathbf{e}}_i\right)}_{{}_{a}\tau_e} = {}_{a}\tau_c \tag{4.39}$$

for some positive magnitude s and some toppling axis a where ${}_{a}\tau_e$ is the torque exerted by the external force $\hat{\mathbf{e}}_i$ about toppling axis a. The robustness to toppling can be improved if the total torque countering toppling, given by

$${}_{a}\tau_c - s \cdot {}_{a}\tau_e, \tag{4.40}$$

is increased. When considering multiple axes simultaneously, the global toppling robustness improvement is given by

$$\mathrm{RI}_{\mathrm{top}}(\mathbf{p}_{\mathrm{i}}, \hat{\mathbf{e}}) = \frac{d}{ds} \sum_{a \in \mathcal{A}_c} \left({}_{a}\tau_c - s \cdot {}_{a}\tau_e\right)^2 \tag{4.41}$$

$$= \sum_{a \in \mathcal{A}_c} \frac{d}{ds} \left({}_{a}\tau_c^2 - 2s \cdot {}_{a}\tau_c \, {}_{a}\tau_e + s^2 \, {}_{a}\tau_e^2\right) \tag{4.42}$$

$$= \sum_{a \in \mathcal{A}_c} 2\left(s \cdot {}_{a}\tau_e^2 - {}_{a}\tau_c \, {}_{a}\tau_e\right), \tag{4.43}$$

which can be positive (improvement) or negative (deterioration).

In multi-object assemblies, (4.41) can be computed for all axes lying on simple paths connecting the external force to the fixed object in the CIG, as done when computing slipping robustness improvement in (4.30), and summed to determine the global improvement in the object's robustness to toppling when the external force is applied.

4.5 Validation and Benchmark Experiments

In the following, we compare the result of our method to three other methods and to a manually computed ground truth. The comparison is done on three simple assemblies, shown in Figure 4.12, for which it is feasible to manually compute the robustness of the assembly to external forces. The first assembly consists in a single cube resting on a plane, the second assembly is a serial chain of three stacked cubes, and the third assembly consists in slab resting on two legs and forming a parallel kinematic structure. In these experiments, the robustness of the assemblies is computed for forces applied normally to the surface of the assembly.

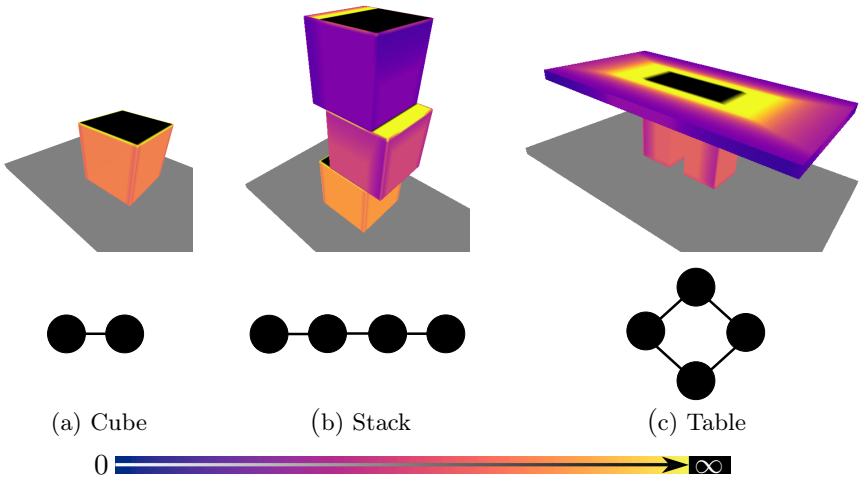


Figure 4.12: Assemblies used to benchmark the accuracy of robustness assessment methods with their respective CIG. The colour of each surface point indicates the robustness to a normal force at that point according to the scale (bottom) with black indicating infinity.

4.5.1 Benchmarked Methods

We benchmark our algorithm against a method making use of commonly available simulators, a method iteratively solving optimization problems, and an approximate robustness heuristic.

Simulation-based

Off-the-shelf dynamics simulators have been used to get a sense of an assembly robustness and to verify stability (Lee et al., 2023; Sun et al., 2024; Yoneda et al., 2023). These physics simulators can be used with general rigid body assemblies and are simple to set up. To determine the robustness of an assembly to an external force applied at a given point, a binary search for the maximum force magnitude that can be sustained by the assembly before motion occurring is performed, as described in Algorithm 3. To do so, initial searching bounds are set (i.e. $[0, 100]$ Newtons) and a force whose magnitude is in the centre of the bounds is iteratively applied to the assembly. Following the force simulation, assembly motion is determined by computing its kinetic energy and comparing it to a threshold value (i.e. 10^{-4} Joules). Whether the assembly moved or not dictates if the upper or lower bound is updated to the force magnitude. The process is repeated a number of times (i.e. 100) before returning the centre of the bounds as the robustness of the assembly to the external force.

In our experiments, we used the PyBullet simulator (Coumans and Bai, 2021) with the full Coulomb friction cone and with no force restitution to avoid bouncing effects. The robustness search is done over 100 iterations, with 0–100 Newtons initial bounds, and a motion threshold of 10^{-4} Joules.

Algorithm 3: Simulator-based Robustness Search

input : Query tuple $(\mathbf{p}_i, \hat{\mathbf{e}}_i)$
output: Robustness $\mathrm{R}(\mathbf{p}_i, \hat{\mathbf{e}}_i)$
Initialize bounds $B_l, B_u \leftarrow [0, 100]$
for *max_iterations* **do**
  $s \leftarrow (B_l + B_u)/2$
  Apply $s\hat{\mathbf{e}}_i$ at $\mathbf{p}_i$ and simulate one time step
  Compute kinetic energy E_{kin}
  if $E_{kin} > threshold$ **then**
    $B_u \leftarrow s$
  else
    $B_l \leftarrow s$
$\mathrm{R}(\mathbf{p}_i, \hat{\mathbf{e}}_i) \leftarrow (B_l + B_u)/2$

Optimization-based

In (Maeda et al., 2009; Chen et al., 2021), the optimization problem in Section 4.3.5 is slightly modified into (4.44)–(4.48) to yield the robustness of an assembly to an external force. The objective function of the original optimization problem is changed to the one of maximizing the magnitude s of an external force applied at a given point $\mathbf{p}$ in the assembly. The equilibrium constraint in (4.46) for the object onto which the force is applied is modified to include the external force, while other constraints remain unchanged. The resulting linear program

$$\max_{\mathbf{f}_i, s} \quad s \tag{4.44}$$

subject to

$$\mathbf{w}_j = \begin{cases} {}_w\mathbf{w}_{g_j} + s\mathbf{B}_e\hat{\mathbf{e}} & \text{if } \mathbf{p} \text{ on object } j \\ {}_w\mathbf{w}_{g_j} & \text{otherwise} \end{cases} \qquad \forall j \in J \tag{4.45}$$

$$\sum_{k}^{|K_j|} \mathbf{B}_k \mathbf{f}_k + \mathbf{w}_j = \mathbf{0} \qquad \forall j \in J \tag{4.46}$$

$$\mathbf{C}_i \mathbf{f}_i \leq 0 \qquad \forall i \in I \tag{4.47}$$

$${}_i\mathbf{f}_{i_n} \geq 0 \qquad \forall i \in I \tag{4.48}$$

is solved for each query $\mathrm{R}(\mathbf{p}, \hat{\mathbf{e}})$.

To solve (4.44)–(4.48) in our experiments, we run the IPOPT solver (Wächter and Biegler, 2006) for a maximum of 1,000 iterations. If an optimal solution to the problem in (4.44)–(4.48) is found, we set $\mathrm{R}(\mathbf{p}, \hat{\mathbf{e}}) = s$. When the solver determines that the problem is infeasible, we set $\mathrm{R}(\mathbf{p}, \hat{\mathbf{e}}) = 0$, and when the solver determines that the problem is unbounded or the maximum number of iterations is reached, we set $\mathrm{R}(\mathbf{p}, \hat{\mathbf{e}}) = \infty$. We perform experiments with square and octagonal approximations of the friction circle. Every contact

interface is discretized into 20 discrete contact points whose positions are selected using an approximate farthest point sampling strategy where the point, from a subset of 10 randomly selected points, farthest from any previously selected contact point is chosen.

Approximate

The approximate robustness assessment proposed in (Nadeau and Kelly, 2025), in which every object is treated in isolation, is also used as a comparison point. Contact interfaces are discretized into 20 contact points and contact forces are computed by solving the optimization problem in Section 4.3.5.

Ground Truth

For simple scenes, it is feasible to manually compute the robustness of the assembly to external forces. This is done through a geometrical analysis of the assembly and by reasoning about contact forces. To do so, the surface of the objects onto which external forces are applied is divided into areas. For each area, the force necessary to produce slipping is computed, and the axes about which the assembly would topple under the influence of the external force are determined. The torque necessary to produce toppling about each axis is determined by computing the average moment arm of contact interfaces and considering the forces at the interfaces. Compared to other approaches, the result thereby obtained is not subject to errors due to contact interface detection or discretization.

This Work

Our proposed robustness assessment method relies on a discretization of contact interfaces into 20 contact points, selected through farthest point sampling, where contact forces are applied. This is the same procedure used for the other methods we benchmark. Areas of contact between objects are detected with a collision detection algorithm.

4.5.2 Evaluation criteria

We evaluate the performance of the methods in terms of running time and accuracy. Running time includes the time spent detecting contact areas, computing contact forces, and computing the robustness of the assembly. Accuracy is evaluated by comparing the robustness obtained with each method to the manually computed ground truth and expressing the difference as a relative error. Points for which the robustness is infinite are not considered in the accuracy evaluation as they result in an undefined relative error.

4.5.3 Results

Ten experiments are performed for each method/assembly combination, for a total of 120 runs. The running time and accuracy results are averaged over the runs and are shown in

Table 4.1. For the optimization-based method, the results are shown for two different polygonal approximations of the friction cone, with 4 and 8 sides, as each type of approximation yields different results.

Method / Scene	**Cube**	**Stack**	**Table**
Approximate			
Time (s)	**0.07**	0.23	0.24
Error (%)	**0**	30	44
Simulation-based			
Time (s)	0.80	10.1	41.7
Error (%)	22	26	67
Optimization-based (4)			
Time (s)	10.9	151	118
Error (%)	16	17	23.6
Optimization-based (8)			
Time (s)	23.6	205	118
Error (%)	11	10	9
This Work			
Time (s)	0.13	**0.20**	**0.23**
Error (%)	**0.00**	**1.00**	**1.20**

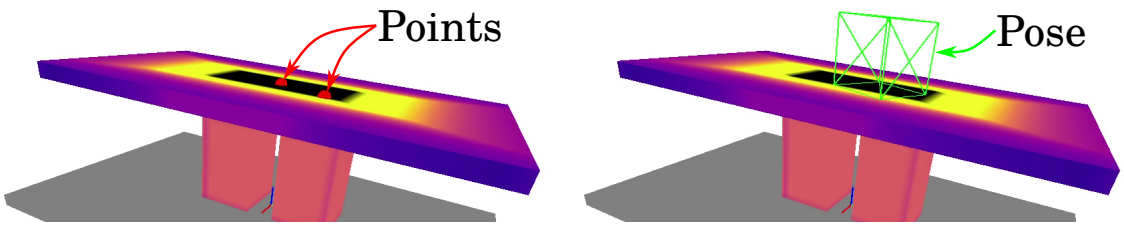
Table 4.1: Accuracy and running time comparison between several methods used to compute assembly robustness. The benchmark is done on three scenes for which the ground truth robustness is manually computed. Numbers between parentheses indicate the number of sides of the polygonal friction cone approximation used.

4.6 Applications

In the following, we demonstrate how our robustness assessment method can be used for stable object placement planning, safe assembly transportation, and disassembly planning.

4.6.1 Object Placement Planning

The problem of stably placing an object in an existing assembly is challenging since feasible poses must (1) make contact with the assembly while avoiding inter-penetrations, and (2) exert forces upon all objects in the assembly while preserving the force equilibrium of the system. Consequently, a very small region of the solution space is feasible, and searching for stable placements by iterating over a discretized set of poses is inefficient and computationally expensive. As will be described with greater details in Chapter 5, the robustness of points on objects in the assembly can be used to efficiently guide the search for stable poses by sampling robust contact points and determining poses that would solicit the selected contact points. An example of the overall process is illustrated in Figure 4.13. While the approximate approach that will be defined in Chapter 5 (henceforth referred to as *Approx*)



(a) Contact point sampling (b) Pose definition

Figure 4.13: Example of object placement planning from contact point robustness. (a) Robustness is used to bias contact point sampling on the assembly. (b) The sampled contact points are used to define a pose for the object to be placed (a small cube).

reduces computational complexity by considering each object in isolation, the method proposed in this chapter can be used to compute the robustness of an assembly more accurately, albeit at a higher computational cost. Hence, our approach can be seamlessly integrated into the planner that defined in Chapter 5 to improve the reliability of the placements found.

The six benchmarking scenes from (Nadeau and Kelly, 2025) (S1–S6) were used to measure the impact of using our robustness assessment method in a placement planning application. The scenes range from a simple cube resting on a flat surface to more complex assemblies that require reasoning about orientation and about using multiple objects as support. Our proposed algorithm was evaluated as a drop-in replacement for *Approx* by reusing the same planning parameters but computing the assembly robustness with the method proposed in this work. A total of 1,200 experiments were performed across the six scenes and two methods, with the average evaluation criteria of each scene-method pair indicated in Table 4.2. For each experiment, the time taken to find a stable placement was recorded, and the minimal robustness of the assembly following the placement was computed to evaluate the quality of the placement.

We also benchmark our proposed method against Approx on three additional scenes, shown in Figure 4.14, for which stable placement planning requires reasoning about the complete assembly. We refer to scenes in the top, middle, and bottom rows of Figure 4.14 as *Slant*, *Balance*, and *Cantilever*, respectively. We perform 100 placement planning experiments on each scene, with the same methods and parameters as in the previous experiments, for a total of 600 runs, and we report the average results in Table 4.3.

4.6.2 Stable Assembly Transportation

A mobile robot transporting a set of objects, while aiming to minimize the transportation time, should also avoid accelerating too quickly to prevent any object from being destabilized. Hence, determining the acceleration for which the assembly is on the verge of moving

	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**Avg.**
Approx							
Time (s)	**0.387**	2.28	0.551	**2.94**	**2.43**	1.90	**1.75**
Rob. (N)	2.43	1.85	**0.817**	0.416	1.53	2.28	1.55
Exact							
Time (s)	0.41	**1.95**	**0.550**	3.26	2.73	**1.84**	1.79
Rob. (N)	**2.45**	**2.14**	0.801	**0.478**	**1.64**	**2.47**	**1.66**
Relative							
Time (%)	+6	**− 14**	+ 0	+ 11	+ 12	**− 3**	+ 2
Rob. (%)	**+ 1**	**+ 16**	− 2	**+ 15**	**+ 7**	**+ 8**	**+ 7**

Table 4.2: Placement planning performance comparison between the approximate and exact methods on their six benchmarking scenes. The relative improvement of our method is also indicated (bottom).

	Slant	**Balance**	**Cantilever**	**Avg.**
Approx.				
Time (s)	4.5	3.0	0.713	2.7
Rob. (N)	0.301	0.524	0.496	0.4
Exact				
Time (s)	3.92	1.07	0.417	1.8
Rob. (N)	0.424	0.716	0.546	0.5
Relative				
Time (%)	**− 13**	**− 63**	**− 43**	**− 40**
Rob. (%)	**+ 33**	**+ 40**	+ 0	**+ 24**

Table 4.3: Results from placement planning experiments on our three more complex scenes from robustness obtained with the approximate and exact methods. The relative improvement of our exact method is also indicated (bottom).

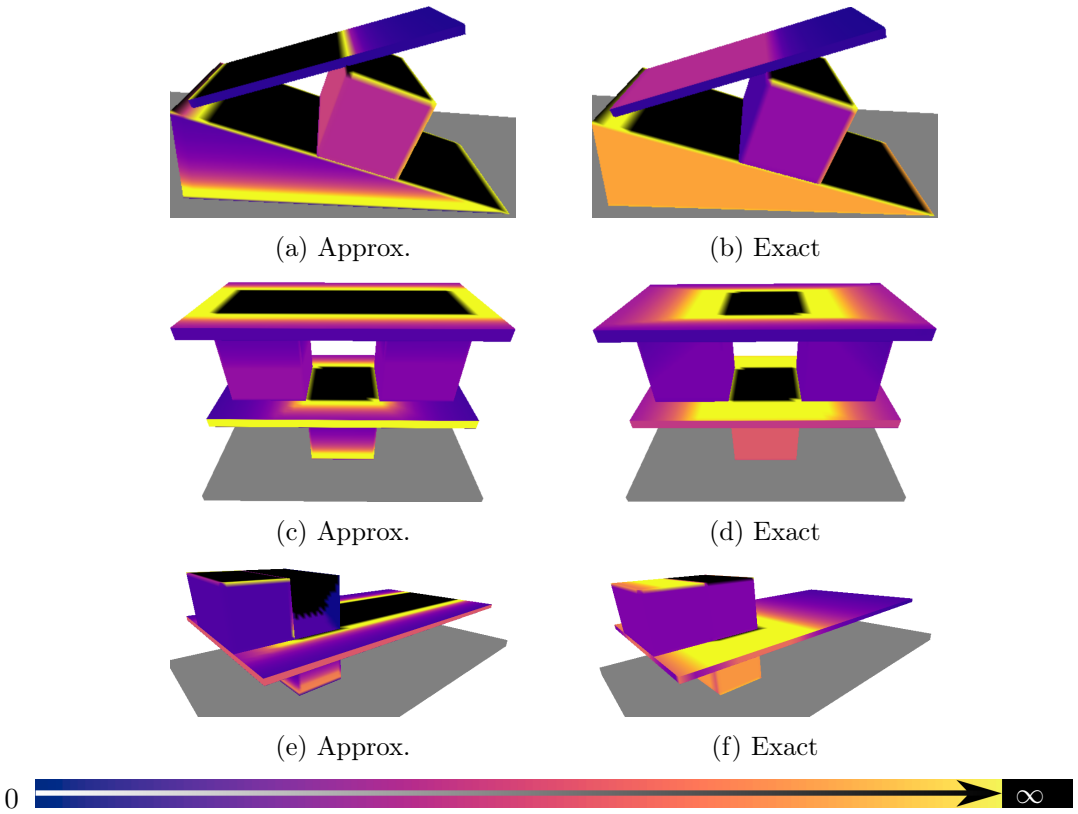
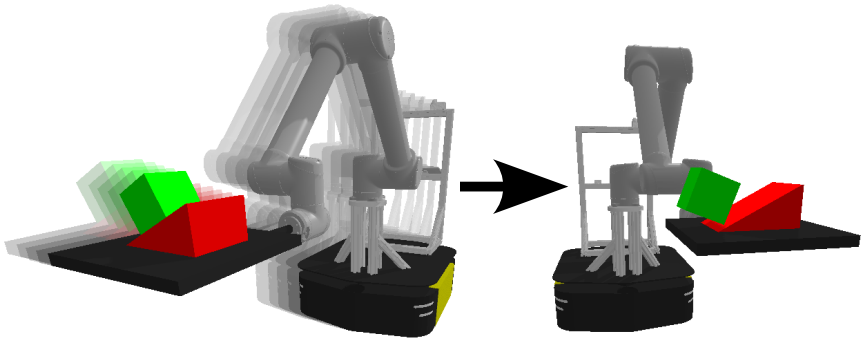


Figure 4.14: The approximate method (left side) produces robustness maps that can misguide the planner into selecting unstable placements. Our method (right side) is more accurate and can better guide the planner.



(a) Mobile robot accelerating (b) MSA Exceeded

Figure 4.15: An example from our simulations illustrating a mobile robot (a) accelerating below the MSA and (b) exceeding the MSA when travelling in the $+Y$ direction.

relative to the support — the maximal sustainable acceleration (MSA) — is key to ensuring the safety and efficiency of the transport. We propose a method that makes use of our robustness criterion to compute the MSA, and we compare our results to simulation-based and optimization-based methods with two simple examples.

According to D'Alembert's principle (Goldstein et al., 2002), an accelerating assembly can be treated as an equivalent static system, with a fictitious force applied at the center of mass in the direction opposite to the acceleration vector. The MSA of an assembly can be determined from its robustness with Algorithm 4, by iterating over all feasible cuts of the CIG and computing the MSA along desired acceleration vectors for each cut. The cut with the smallest MSA will be the first to move when the assembly is accelerated in the direction of the corresponding acceleration vector.

Our algorithm is showcased in Figure 4.16, in which the MSA of two assemblies is computed for 100 acceleration vectors in the horizontal plane. The result from our algorithm is compared to the MSA computed using the simulation environment from (Heins and Schoellig, 2023), in which a mobile manipulator carries an assembly as shown in Figure 4.15. Simulation experiments are performed with the PyBullet physics engine (Coumans and Bai, 2021) at 900 Hz, with a contact error reduction parameter set to 0.2 for Figure 4.16a and 0.15 for Figure 4.16b. The MSA is assumed to be reached when a relative velocity of 0.05 m/s between objects in the assembly is exceeded. We also compare to two variations of the optimization-based approach defined in Section 4.5.1: one with a square approximation of the friction circle (Optim. (4)) and one with an octagonal approximation (Optim. (8)). For each acceleration direction, the optimization problem in (4.44)–(4.48) is solved with $\hat{\mathbf{e}}$ set to the acceleration direction and applied at the centre of mass of each object. The time taken to compute the MSA with the simulation-based approach, optimization-based approach, and robustness assessment method for the example in Figure 4.16b is 38, 3.6 and 0.63 seconds, respectively.

In the example pictured in Figure 4.16a, the assembly can sustain a large acceleration in

the $-Y$ direction (i.e., to the left) due to large coefficients of friction and due to the fact that the inertial force exerted on the cube will tend to push it against the slant and improve the contact conditions. At some point (i.e., 11.8 m/s^2), however, the inertial force will overcome the frictional forces and the assembly will slide to the right. In comparison, a much smaller acceleration (i.e., 3.27 m/s^2) can be sustained in the $+Y$ direction before the cube topples to the left under the influence of the inertial force. Adding a second cube behind the first one, as shown in Figure 4.16b, prevents the first cube from toppling to the left and gives rise to a 30% MSA increase in the $+Y$ direction. This comes at the cost of a MSA decrease in the $\pm X$ directions, as the second cube elevates the centre of mass of the assembly.

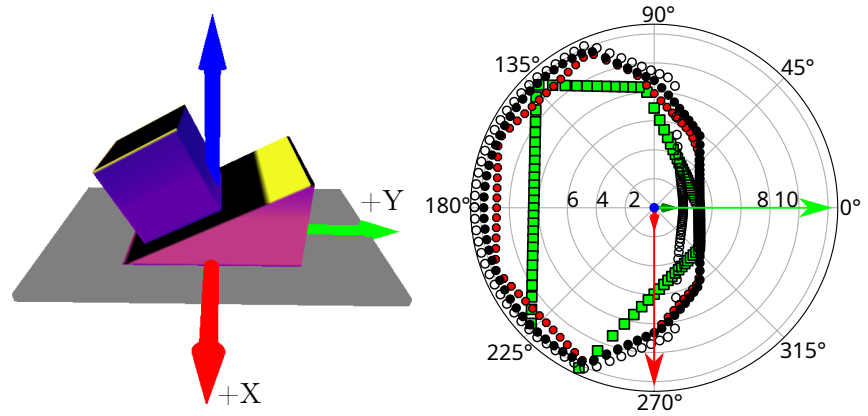
In practice, the MSA can be used to inform kinodynamic motion planners (Donald et al., 1993) on the maximal accelerations that can be safely reached by a mobile robot transporting an assembly (Kunz and Stilman, 2014; LaValle and Kuffner Jr, 2001; Webb and Van Den Berg, 2013; Lau et al., 2009).

Algorithm 4: Maximal Sustainable Accelerations

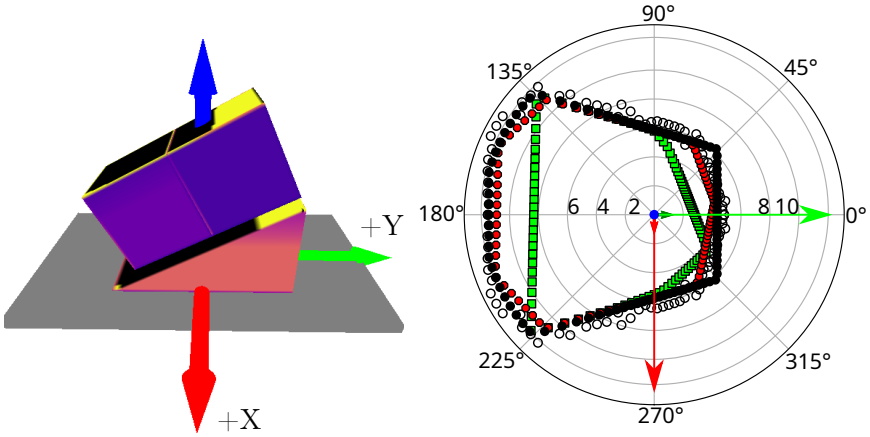
input : Set of 3D acceleration directions $\mathcal{D}$
output: Set of maximal sustainable accelerations $\mathcal{X}$
1. Initialize all $x_i \in \mathcal{X}$ to ∞
2. Determine the feasible CIG cut set $\mathcal{C}$
foreach $c \in \mathcal{C}$ **do**
  3. Compute the mass m and centre of mass $\mathbf{c}$ of c
  foreach $\mathbf{d}_i \in \mathcal{D}$ **do**
    4. Compute the MSA in $\mathbf{d}_i$ with $x_i \leftarrow \min\{x_i, \mathrm{R}(\mathbf{c}, -\mathbf{d}_i)/m\}$

4.6.3 Disassembly Planning

Determining the order in which objects in contact can be removed without causing instabilities in the assembly can enable robots to automate disassembly operations that are requisite when a subset of the objects in the assembly have to be moved. The tasks of disassembly sequencing and backward assembly planning (i.e., assembly by disassembly), which recursively decompose a given assembly into simpler subassemblies, can be solved by considering cuts in the kinematic structure (Lee and Yi, 1994; Lee and Moradi, 1999). Since our proposed approach is based on a graph representation of the assembly (i.e., the CIG), it can directly be used for disassembly planning. To this end, the robustness of the assembly to forces exerted at the centre of mass of each object and directed opposite to their acceleration (i.e. gravity) is computed. An object can be safely removed when the force required to move it is strictly opposed by its inertia (e.g., its weight). A lower or higher robustness at the centre of mass would indicate that the assembly depends on the object to remain stable. An example of disassembly planning is shown in Figure 4.17, where the order in which objects can be safely removed is determined by iteratively computing the robustness of the assembly to forces exerted at the centre of mass of each object and directed opposite to gravity (i.e.,



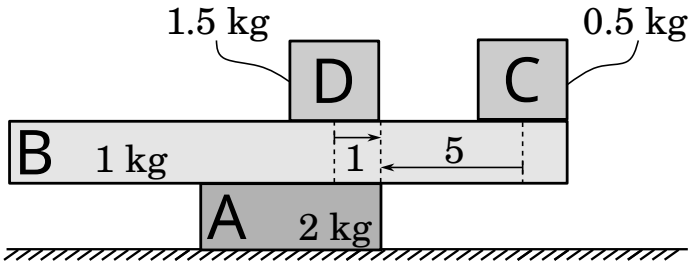
(a)



(b)

Simulation Robustness Optim. (8) Optim. (4)

Figure 4.16: The maximum sustainable acceleration (in m/s^2) of an assembly in the horizontal plane (right side) when a single cube is on the slant (top), and when a second cube is placed behind the first one (bottom). Filled dots result from our algorithm while empty dots result from dynamics simulations. Red octagons and green squares respectively result from an optimization-based approach with octagonal and square Coulomb circle approximations.



Step / Object	A	B	C	D
1	5	3	(0.5)	0.5
2	4.5	2.5	—	(1.5)
3	3	(1)	—	—
4	(2)	—	—	—

Figure 4.17: Disassembly planning of a structure of four objects (top). The order in which objects can be safely removed is determined by iteratively computing the robustness of the assembly to forces exerted at the centre of mass of each object and directed opposite to their acceleration (table values). An object can be safely removed when this value is equal to its weight. In this example, the disassembly order is: C, D, B, A.

upward).

4.7 Discussion

4.7.1 Robustness Assessment: Sources of Errors

In this work, our proposed method is compared to three other methods used for computing the robustness of assemblies: an optimization-based approach, a simulation-based approach, and a heuristic. All methods rely on discretizing contact interfaces, thereby violating the assumption of uniform pressure distribution and leading to erroneous estimates of the centre of friction estimate. Nonetheless, contact interface discretization is reasonable with rigid objects whose coefficient of friction has been shown to vary significantly across contact areas (Yu et al., 2016), thereby introducing uncertainty in the location of the centre of friction.

According to our validation results in Table 4.1, the simulation-based approach yields a large average relative robustness error of 38% on simple scenes. This can be explained by the fact that dynamics simulations do not assume static equilibrium, consider few contact points at every time step, and make various approximations to speed up the process.

In contrast, optimization-based approaches can yield more accurate results but are lim-

ited by the Coulomb friction circle approximation. On average, the distance between any point on a circle and its closest point on an inscribed square is about 20% of the circle radius. For an inscribed octagon, the average distance is about 10% of the circle radius. These values correspond to the average relative robustness errors obtained with the square and octagonal approximations in Table 4.1, suggesting that optimization-based approaches can yield accurate robustness estimates when using a large number of sides to approximate the Coulomb circle.

Our proposed method solves the quadratic equation in (4.20) when computing robustness instead of relying on linear approximations of the Coulomb circle. However, like the other approaches, the accuracy of our method is limited by the discretization of contact interfaces, which is likely responsible for the small relative robustness errors shown in Table 4.1.

Finally, like our method, the approximation that will be described in Chapter 5 does not rely on linear approximations of the Coulomb circle and yields accurate results for isolated objects. However, for assemblies whose CIG has a depth greater than one, the approximate method will be inaccurate, as suggested by our validation results.

4.7.2 Performance in Object Handling Applications

Assessing the robustness of assemblies to external forces can be used in various object handling operations, such as object placement, safe transportation, and disassembly planning. In Section 4.6, we provided examples of how our method can be used for these purposes. This section discusses the performance of our method in these applications.

Comparing the performance of the object placement planner with approximate dynamics (Approx) to a variation of the same planner that uses our robustness assessment method, we observe that our method is competitive in terms of planning time (+2%) and produces slightly more stable placements (+7%). On more complex scenes, however, results in Table 4.3 suggest that using our method can significantly improve the planning time while yielding more robust placements. On three examples of such scenes, our method is 40% faster in finding a stable placement and yields assemblies that are 24% more robust on average. Although the computational complexity of our robustness assessment method exceeds that of Approx, these results can be explained by the fact that the robustness assessment only has to be performed once per placement while a large number of placement planning iterations can be required to find a stable placement.

For efficiently transporting assemblies, our results in Figure 4.16 clearly illustrate the importance of carefully selecting the orientation of the assembly when transporting it. While simulation-based or optimization-based approaches can be used to compute the maximal sustainable acceleration, our method does not require parameter tuning, is faster, and more accurate.

Finally, we also showed how an accurate robustness assessment can be used to determine the order in which objects can be safely removed from an assembly, enabling planning

disassembly sequences.

4.7.3 Practical Considerations: Approximations and Safety

The worst-case running time of Algorithm 2 is proportional to the number of feasible cuts in the CIG and exponential in the number of objects in the assembly. A practical approach to limit the running time of our method consists in considering that heavy objects are in fact immobile. This can greatly reduce the number of feasible cuts in the CIG and lower the computational complexity. In practice, the CIG can be easily analyzed to determine whether such an approximation is needed or applicable.

When assembling or transporting objects, it might be beneficial to reduce the odds of instability occurring due to object model inaccuracies. A simple strategy to achieve this consists in considering that the mass of objects is smaller than their estimated value. Since robustness linearly increases with object mass, underestimating the mass of objects will lead to a conservative estimate of the assembly robustness.

4.7.4 Limitations and Future Work

Our proposed method for computing the robustness of assemblies to external forces assumes that object pose, friction coefficients, and object shapes are known with certainty. While object shapes can be accurately provided by CAD models, practical applications will require estimating the pose of objects in the assembly. Furthermore, friction coefficients have been shown to vary significantly across contact areas (Yu et al., 2016). Future work could consider propagating uncertainty from object poses and friction coefficients into the robustness assessment, thereby enabling robust and reliable object handling operations. Moreover, while our method assumes that a single external force is exerted on the assembly, bimanual manipulation tasks may require considering the influence of multiple external forces acting on the assembly. Considering the influence of multiple simultaneous external forces on the assembly is an interesting avenue for future work.

4.8 Conclusion

In this work, we proposed a method to compute the robustness of an assembly to external forces, facilitating autonomous decision-making in tasks involving rigid objects in contact. Our method does not rely on heuristics or approximations, making it dependable in a broad range of scenarios involving multi-object assemblies. Through theoretical validation and experimental assessment, we demonstrated that our method is more accurate and much more efficient than existing approaches for computing the robustness of assemblies. We showed that our method can be used to plan stable object placements in complex assemblies, compute the maximal sustainable acceleration of mobile robots transporting assemblies, and

plan disassembly sequences. We expect our algorithms to be valuable in various applications involving forceful interactions with the environment, particularly in the context of autonomous object handling.

Chapter 5

Stable Object Placement Planning

> Philosophers have hitherto only interpreted the world in various ways; the point is to change it.
>
> Karl Marx, 1845.

This chapter introduces a planner designed to guide robot manipulators in stably placing objects within intricate scenes. Our proposed method reverses the traditional approach to object placement: our planner selects contact points first and then determines a placement pose that solicits the selected points. This is instead of sampling poses, identifying contact points, and evaluating pose quality. Our algorithm facilitates stability-aware object placement planning, imposing no restrictions on object shape, convexity, or mass density homogeneity. We propose to use an approximation to the static robustness assessment introduced in Chapter 4 to guide the planner in selecting contact points that are more likely to lead to stable placements while avoiding combinatorial computational complexity. Our proposed stability heuristic enables our planner to find a solution about 20 times faster when compared to the same algorithm not making use of the heuristic and eight times faster than a state-of-the-art method that uses the traditional sample-and-evaluate approach. Our proposed planner is also more successful in finding stable placements than the five other benchmarked algorithms. Derived from first principles and validated in ten real robot experiments, our planner offers a general and scalable method to tackle the problem of object placement planning with rigid objects.

5.1 Motivation

Robots with the capability to stably place objects in contact with one another hold the potential to reduce the need for human intervention in various tasks, spanning home chores, industrial operations, and work in outdoor environments (Alterovitz et al., 2016). In do-

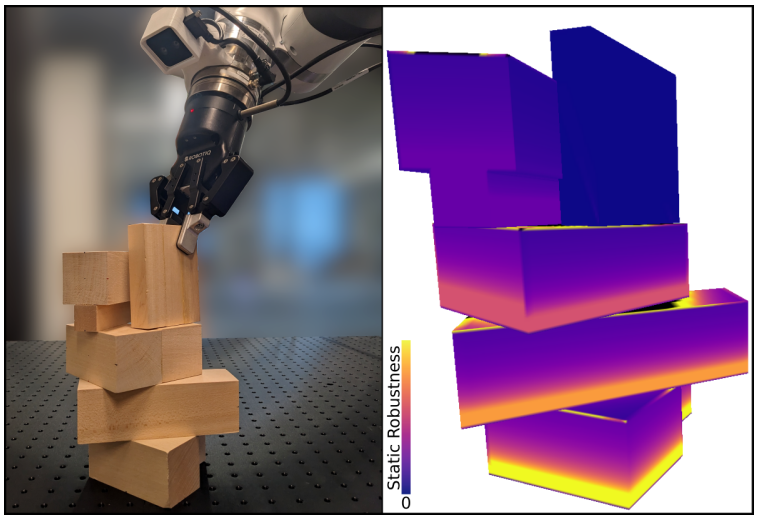


Figure 5.1: Intricate stable structure generated by our object placement planner. The coloured surface, indicating the relative robustness of surface points to a normal force, is used to guide the planner in selecting contact points that are more likely to lead to stable placements.

mestic settings, tasks like tidying the living space by rearranging objects or organizing the garage storage could be automated (Srinivas et al., 2023). Effective planning for stable object placement could facilitate safe palletizing of mixed products or enable the autonomous loading of trucks in industrial settings. In outdoor environments, autonomous excavators could be used in disaster relief through debris removal (Wermelinger et al., 2021). However, stably packing or rearranging rigid objects in contact is challenging due to the subtle yet influential force interactions between objects that determine their stability. Since the problem of determining force interactions is known to be NP-hard (Baraff, 1991), many approaches resort to planning heuristics based on shape information only and do not generalize well across tasks. Moreover, while existing methods have typically considered geometry and dynamics in two separate steps (Chen et al., 2021; Haustein et al., 2019) we propose to **(i)** combine them through a *static robustness map* built from an approximate variation of the robustness assessment introduced in Chapter 4, as shown in Figure 5.1, and **(ii)** define a planner that leverages this map to generate stable placements more efficiently.

A general approach to placement planning requires considering the potential motion of objects under forceful interactions. In static assemblies, the mass and centre of mass (henceforth referred to as the inertial parameters), as well as the friction coefficients, are sufficient to characterize an object's inertia—its resistance to acceleration. We call, *inertia-aware*, a planner that operates based on these parameters, with knowledge of the second mass moments not being required in static scenarios. When dealing with rigid objects, a pose is sufficient to determine the position of all object points and is the primary representation used to define placements due to its simplicity. Object placement planning consists of selecting the pose that a given object should take in a given assembly. Crucially, a *valid* placement

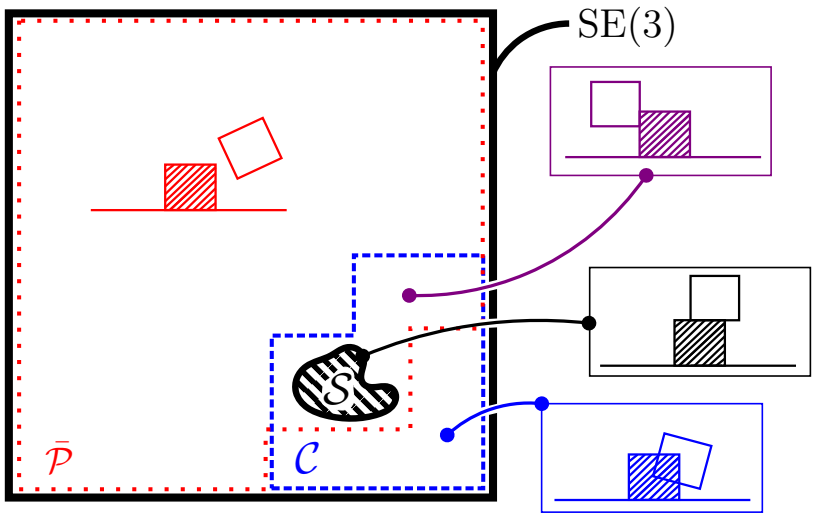


Figure 5.2: The set of stable placement poses $\mathcal{S}$ (hashed area) for the white cube is a very small subset of SE(3) such that a random sample of SE(3) has very little chance of being in $\mathcal{S}$. Any element of $\mathcal{S}$ must also be included in the set of non-penetrating poses $\bar{\mathcal{P}}$ (dotted) and the set of poses for which the object being placed is in contact with the scene $\mathcal{C}$ (dashed).

pose should not violate object solidity (i.e., an object cannot penetrate another one) and assembly stability (i.e., no objects should move after placing the object). In essence, the problem of finding valid placement poses is difficult due to the fact that a very small subset of all poses are valid, as pictured in Figure 5.2. A search over SE(3) is thus highly inefficient, and an approach based on random sampling is unlikely to find a valid pose. In contrast, the approach proposed in this chapter only considers poses for which the object is in contact with the scene, a small subset of SE(3), and make use of the robustness assessment introduced in Chapter 4 to increase the likelihood of finding stable placements.

In sum, this chapter introduces an inertia-aware object placement planning algorithm, leveraging a physically-grounded heuristic to generate stable placements of known objects, with scalability to tackle large-scale problems. The remainder of this chapter is organized as follows. Section 5.2 reviews the literature on assembly stability and object placement planning, highlighting the novelty of our proposed algorithm. In Section 5.3, the inertia-aware object placement planning problem is formally defined, while Section 5.4 introduces a physically-grounded heuristic to assess an assembly's capacity to withstand external forces. Our proposed planner is detailed in Section 5.5, and the results of more than 1,500 experiments involving six scenes and six algorithms are presented in Section 5.6, demonstrating the benefits of our method. In Section 5.7, practical validation of our proposed algorithm is carried out through 10 real-world experiments involving 50 placements, in which a robot manipulator builds assemblies while recording any failures that occur. Finally, Section 5.8 concludes with insights gained from our simulation and real robot experiments.

5.2 Related Work

Early methods to produce stable orientations of an assembly under gravity, for fixturing purposes, include (Mattikalli et al., 1993), which ignores friction. In (Mattikalli et al., 1996), friction is taken into account but stability is not guaranteed due to indeterminacies in the distribution of contact forces (Pang and Trinkle, 2000), a common issue. Selecting a set of feasible contact forces can be done by resorting to optimization-based methods (Whiting et al., 2012; Kao et al., 2022) that make use of the principle of virtual work (Goldstein et al., 2002), and integrating kinematic constraints to ensure that the solution is physically plausible. Such methods have been shown to be equivalent to finite element methods (FEM) (Shin et al., 2016) for rigid objects, and, similar to FEM, are computationally expensive and non-deterministic. Stability assessment under a large number of random force disturbances is perfomed in (Chen et al., 2021) by solving a *min-max* optimization problem for every perturbation, making it the primary bottleneck of the planning algorithm. Meanwhile, this chapter proposes a physically sound heuristic that is quick to compute by avoiding combinatorial complexity. Also, in contrast to (Chen et al., 2021), our approach considers stability first instead of verifying it only at the end.

Classical methods used to assess the stability of a grasp (Borst et al., 1999; Ferrari et al., 1992) can provide a quantitative evaluation of the stability of a workpiece, while newer algorithms can produce stable multi-object grasps (Agboh et al., 2023) with rigid convex objects. Although these methods offer a comparative basis for different placements, their application to object placement planning is limited. This is because they do not take into account the fact that fixed rigid objects can withstand very large forces, nor do they guide the planner on how to position an object on a stable structure—two aspects that our proposed algorithm addresses.

In general, the assembly task planning problem has been shown to be NP-complete (Kavraki et al., 1993), which explains why task and motion planning methods (Toussaint, 2015) have only been applied to small-scale problem instances. Learning-based methods that do not resort to handcrafted heuristics have been proposed for assembly planning (Xu et al., 2019; Zhu et al., 2021) but also struggle to generalize across tasks (Jiang et al., 2012). While most object planning algorithms ignore the inertial parameters of the objects, the work in (Haustein et al., 2019) make use of the centre of mass of the object being placed but is limited to isolated placements on fixed horizontal surfaces. Through our experiments, we show examples where this common approach may produce unstable placements. As an attempt at a more general object placement planning algorithm, physics simulators have been integrated into planning algorithms (Lee et al., 2023) but are prohibitively slow to generate valid plans, even for small-scale problems. In contrast, this chapter proposes a general method derived from first principles that avoids the overhead of dynamics simulators to better scale to large problems.

Planning stable placements not only requires determining reaction forces but also the set

of contact points forming the interfaces between objects, which is a complex task in general. Hence, previous works have simplified the problem by limiting the shape of the objects to rectangular workpieces (Chen et al., 2021; Motoda et al., 2022), or by segmenting the scene into simple shapes like cylinders and cuboids (Kartmann et al., 2018). In contrast, our method can handle any shape described by a triangular mesh, with the placement planning time growing sub-quadratically with the number of vertices in the mesh.

5.3 Inertia-aware Object Placement Planning

Let $\mathcal{O}$ be a set of N objects with known shape $\mathcal{V}_i \subset \mathbb{R}^3$ and material composition $\rho_i(\mathbf{p}) \; \forall \; \mathbf{p} \in \mathcal{V}_i$ for $i \in \{1, \dots, N\}$. Let an assembly $\mathcal{O}_a \subset \mathcal{O}$ be a set of objects in frictional contact, and $\mathcal{O}_f \subset \mathcal{O}_a$ be the subset of objects fixed in the limited space of the scene. The problem is to find a sequence of penetration-free placement poses that maximizes the force needed to displace any object. Hence, different from typical assembly planning problems, the goal configuration is not given in terms of the pose of the objects in the assembly, but rather in terms of an objective that is to be maximized.

5.4 Approximate Static Robustness Assessment

When placing an object amongst others, it is usually desirable that the object be unlikely to move after being placed. For rigid polyhedral objects, ultimately, instability will occur when an object slips or topples. In both cases, the event is triggered when an applied force exceeds a threshold that we call the *static robustness* (SR): the maximum amount of force that can be exerted along a given direction, on a given point, before an object in the assembly moves. In Chapter 4, we introduced an algorithm that computes the SR for all points on the surface of objects in and assembly. We termed *static robustness map* the scalar field

$$\mathrm{SR}(\mathbf{p}, \hat{\mathbf{n}}) \quad \forall \; \mathbf{p} \in \mathcal{P}, \tag{5.1}$$

where $\mathcal{P}$ is the set of points on the objects, and $\hat{\mathbf{n}}$ is the normal to the surface at $\mathbf{p}$. The stability of an object being determined by its supporting contact forces, it follows that planning for stable placement involves reasoning about contact forces in the assembly. However, the optimization problem introduced in Chapter 4 to solve for contact forces is computationally expensive due to its non-linear constraints.

5.4.1 Contact Force Computation

To reduce the computational complexity of contact force computation in (4.12), we relax non-linear constraints and approximate the friction cone with a four-sided pyramid. The

resulting optimization problem

$$\min_{\mathbf{f}_i \ \forall \ i,j} \quad \sum_{i}^{N} \|\mathbf{f}_i\|^2 \tag{5.2}$$

subject to

$$\sum_{k}^{K_j} \mathbf{B}_k \mathbf{f}_k + {}_w\mathbf{w}_{g_j} = \mathbf{0} \qquad \forall j \in J \tag{5.3}$$

$$\mathbf{C}_i \mathbf{f}_i \leq 0 \qquad \forall i \in I \tag{5.4}$$

$${}_i\mathbf{f}_n \geq 0 \qquad \forall i \in I \tag{5.5}$$

is a linearly-constrained quadratic program that can be solved in polynomial time by standard solvers (Wächter and Biegler, 2006). In the above, $\mathbf{B}_k$ maps $\mathbf{f}_k$ (the reaction force at the k-th contact point of the j-th object) to a world-frame wrench, $\mathbf{C}_i$ enforces the friction constraints at the i-th contact point, and ${}_w\mathbf{w}_{g_j}$ is the wrench exerted by inertial forces on the j-th object. See Section 4.3.5 for more details about the definition of $\mathbf{B}_k$, $\mathbf{C}_i$, and ${}_w\mathbf{w}_{g_j}$.

Alternatively, an unconstrained solution can be obtained via a QR decomposition (Golub and Van Loan, 2013) of the data matrix, with

$$\mathbf{Q}, \mathbf{R} = \text{qr}\left(\begin{bmatrix} \mathbf{B}_1 & \dots & \mathbf{B}_j & \mathbf{B}_{K_j} \end{bmatrix}\right) \ , \tag{5.6}$$

by solving

$$\mathbf{R}^{\mathsf{T}}_{1:6,1:6}\mathbf{z} = \mathbf{b} \tag{5.7}$$

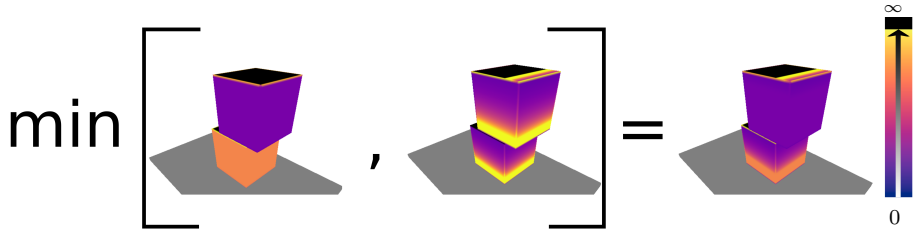
for $\mathbf{z}$, where $\mathbf{R}_{1:6,1:6}$ is the upper-left 6×6 sub-matrix of $\mathbf{R}$. The reaction forces can then be obtained with

$$\mathbf{f} = \mathbf{Q}_{:,1:6}\mathbf{z} \ , \tag{5.8}$$

where $\mathbf{Q}_{:,1:6}$ is built from the first six columns of $\mathbf{Q}$. This approach is expected to be the fastest but is unconstrained and might violate the friction cone and the non-tensile force constraints. For placement planning, the QR-based approach can be used to quickly get an approximate solution but the validity of promising solutions should be verified with the optimization-based approach.

5.4.2 Approximate Static Robustness Computation

In general, computing the static robustness map of an assembly involves computations that grow exponentially with the number of objects. This is essentially due to the fact that a force applied to an object can be transmitted to all other objects in the assembly. To avoid this combinatorial complexity, we propose to consider each object in isolation and compute the static robustness map for each object separately. Hence, to compute an object's slipping robustness, only contact points on the object are considered, with no need to solve the



(a) (b) (c)

Figure 5.3: Combining approximate slipping (a) and toppling (b) robustnesses to obtain the overall static robustness (c) with values colour coded according to the scale on the right (black being infinite).

maximum flow problem defined in Section 4.4.2 to determine the exact slipping SR.

An isolated object onto which an external force is applied will topple about an edge of the convex hull of the contact points on the object. Any other axis would require a greater amount of force to rotate the object since the lever arm would be smaller. Hence, the convex hull of the object contact points is computed via the QuickHull algorithm (Barber et al., 1996), whose complexity is $O(|K_j|^2)$ in the worst case, to produce a list of axes about which the object could topple. The last two steps of the algorithm defined in Algorithm 2 are then performed to yield the toppling SR for the object. Slipping and toppling robustnesses may be combined as

$$\mathrm{SR} = \min\left\{\mathrm{SR}_{\mathrm{slip}}, \mathrm{SR}_{\mathrm{top}}\right\} \tag{5.9}$$

to obtain the overall static robustness of the object, as pictured in Figure 5.3. The result of the mapping can be seen in Figure 5.1, Figure 5.3, and Figure 5.8, where the colour indicates the relative magnitude of the maximal force that can be applied on a point in the direction normal to the surface. A lighter colour indicates that a greater force can be applied before the object slips or topples, with black indicating that an infinite force can be applied. The robustness function obtained via the use of our approximative approach is unlikely to be exact in multi-objects assemblies. However, it is quick to compute because it avoids any operation whose computational complexity grows exponentially with the number of objects in the assembly.

5.5 Placement Planning From Static Robustness

Ideally, the normal contact force at each contact point is much greater than the tangential force such that the object is far from slipping. Therefore, our proposed planner uses the robustness to normal forces to plan stable placements. In the following, we assume that a known object has to be placed in a given assembly of objects for which the robustness in the normal direction at all points on the surface of the objects in the assembly is known. Also, placements are assumed to be performed in a linear (i.e., a single object is moved at a time), monotone (i.e., placing does not involve rearranging), and sequential (i.e., a single

gripper is used) fashion (Wang et al., 2021).

By placing an object in an assembly, at least one contact interface between objects is created, which fixes some of the degrees of freedom (DoFs) of the object. Any nondegenerate[1] contact interface can be categorized into (Xiao, 1993; Ji and Xiao, 2001):

- a face/face plane contact that fixes 3 DoFs,
- a face/edge line contact that fixes 2 DoFs,
- an edge/edge point contact that fixes 1 DoF, and
- a face/vertex point contact that fixes 1 DoF.

At least three contact interfaces are necessary to constrain the object to a specific pose by fixing all 6 DoFs (Ji and Xiao, 2001). However, to place an object successfully without inducing a collision, at least one DoF must remain unconstrained. With two non-collinear contact interfaces, from one (e.g., with face/face interfaces) to four (e.g., with face/vertex interfaces) DoFs will remain unconstrained, ensuring that the object can be placed without inducing a collision. A third contact interface might be needed to ensure stability under gravity, but the interface does not necessarily need to be selected prior to the computation of the placement pose. Furthermore, in a face/face contact situation, the normal vectors on each side of the contact interface must act in direct opposition. In a face/edge contact, the normal vector of the face must be orthogonal to the contact edge.

Our approach is summarized by Algorithm 5, with each step described in greater detail in the following sections. Our algorithm is based on sampling points on the assembly and on the object to place, and then defining a pose for the object based on the two sets of points. With this sampling-based approach, it is expected that many candidates will turn out to be invalid, and that the process will need to be iterated several times before a stable, non-penetrating pose is found. Each iteration therefore needs to be performed as quickly as possible.

5.5.1 Contact Point Selection

The selection of candidate contact points on the surface of objects in the assembly is performed in an iterative, probabilistic manner based on the robustness of the objects to normal forces. At each iteration of the planning algorithm, two points are sampled according to the probability distribution in (5.10). With $\mathrm{r}(\mathbf{p}, \hat{\mathbf{n}})$ being the robustness of the object in the normal direction at point $\mathbf{p}$, the probability of sampling this point is given by

$$\mathrm{P}(\mathbf{p}) = \frac{\min\left(\mathrm{r}(\mathbf{p}, \hat{\mathbf{n}}), Q\right)}{\sum_{\mathbf{p}_i \in \mathcal{P}} \min\left(\mathrm{r}(\mathbf{p}_i, \hat{\mathbf{n}}_i), Q\right)} , \tag{5.10}$$

[1] In (Xiao, 1993), edge/vertex, vertex/vertex, and parallel edge/edge contacts are defined as being degenerate.

Algorithm 5: Placement Planning From SR

input : Assembly $\mathcal{O}_a$, object to place $\mathcal{O}_i$, static robustness map $\mathrm{r}(\mathbf{p}, \hat{\mathbf{n}}) \ \ \forall\ \mathbf{p} \in \mathcal{P}$
output: Stable pose ${}_w\mathbf{T}_o$ of $\mathcal{O}_i$ if found

```
while max. number of iterations not reached do
    Sample two points ₘpₐ and ₘp_b on O_a with (5.10)
    Find two points ₒp_q and ₒp_r on O_i support geometry whose normals oppose the
      ones on the assembly and whose relative position coincides
    Define ₘT_o from ₘpₐ, ₘp_b, ₒp_q, and ₒp_r
    if ₘT_o does not result in inter-penetration then
        Solve (5.8) for reaction forces (QR-based)
        if tension forces are below threshold then
            Solve (5.2) for reaction forces (QP-based)
            if forces are equilibrated then
                return ₘT_o
```

where Q is the robustness above which the odds of being sampled are set to be constant. In practice, Q can be defined such that the largest finite robustness in the assembly has a small probability of being sampled (e.g., 10%). This can be done by initializing Q to a larger value $Q_0 > 0$ that is exponentially decayed with the number of iterations k at a rate $\lambda < 1$ with $Q_k = Q_0\lambda^k$. This way, the probability of sampling any point in $\mathcal{P}$ converges to

$$\lim_{k\to\infty} \mathrm{P}\left(\mathbf{p}\right) = \frac{Q_k}{|\mathcal{P}|Q_k} = \frac{1}{|\mathcal{P}|}\ , \tag{5.11}$$

where $|\mathcal{P}|$ is the cardinality of $\mathcal{P}$, the set of surface contact points. With such a scheme, the odds of sampling a point with a large robustness are initially higher, but decrease with the number of iterations. All points are eventually considered with almost equal probability, and any valid placement pose is eventually considered.

Extension: Sampling on Fixed Supports

Sometimes, it might be desirable to place the object on a fixed support (e.g. table) instead of amongst the objects in the assembly. For instance, if placing the object on the assembly would result in a weakened assembly, it might be preferable to place the object on a fixed support whose static robustness is, by definition, infinite. In this case, however, it is usually desired that the object be placed in a compact manner. The probability function in (5.10) can be easily extended to accomodate this requirement. For instance, the set of scene points

$\mathcal{P}$ can be augmented with a set of points $\mathcal{P}_f$ uniformly distributed on the fixed support with

$$w\left(\mathbf{p}\right) = \begin{cases} P(\mathbf{p}) \ \forall \mathbf{p} \in \mathcal{P} \\ \max_{\mathbf{p} \in \mathcal{P}} \left(\mathrm{P}\left(\mathbf{p}\right)\right) \ \forall \mathbf{p} \in \mathcal{P}_f \end{cases} , \tag{5.12}$$

$$w\left(\mathbf{p}\right) = w\left(\mathbf{p}\right) e^{-\frac{\gamma}{s^2}^k \left\| {}_w\mathbf{p}_p - {}_w\mathbf{p}_c \right\|^2} \ \forall \mathbf{p} \in \mathcal{P} \cup \mathcal{P}_f \ , \tag{5.13}$$

$$P\left(\mathbf{p}\right) = \frac{w\left(\mathbf{p}\right)}{\sum w\left(\mathbf{p}\right)} \ \forall \mathbf{p} \in \mathcal{P} \cup \mathcal{P}_f \ , \tag{5.14}$$

where (5.13) increases the probability of sampling points closer to the centroid ${}_w\mathbf{p}_c$ of the scene and (5.14) ensures that the sum of the probabilities is equal to one. In (5.13), γ is the rate at which the probability of sampling a point close to the centroid of the scene decreases and s is the scale of the scene.

5.5.2 Object Point Selection

To define a placement pose, a pair of object features (i.e., faces or edges) is considered. One point per feature is selected such that the distance between the object points corresponds to the distance between the scene contact points. To favour stability, the object points are selected such that the line joining the points is as close as possible to the centre of mass of the object. Prior to point selection, candidate pairs of features on the object are validated by considering the relative orientation of the feature normals to ensure that they can *afford* the support geometry of the contact points in the assembly. Each feature pair can either be *face-face* (parallel, coplanar, or intersecting), *face-edge* (parallel, or intersecting), or *edge-edge* (parallel, intersecting, or skew).

In the following, we assume that the contact points ${}_w\mathbf{p}_a$ and ${}_w\mathbf{p}_b$ are sampled on object faces in the scene with normals ${}_w\hat{\mathbf{n}}_a$ and ${}_w\hat{\mathbf{n}}_b$ such that $\left\| {}_w\mathbf{p}_a - {}_w\mathbf{p}_b \right\| = L$. The relative orientation of ${}_w\hat{\mathbf{n}}_a$ and ${}_w\hat{\mathbf{n}}_b$ can be obtained with

$$\hat{\boldsymbol{\omega}} = {}_w\hat{\mathbf{n}}_a \times {}_w\hat{\mathbf{n}}_b \ , \tag{5.15}$$

$$\theta = \arccos\left({}_w\hat{\mathbf{n}}_a \cdot {}_w\hat{\mathbf{n}}_b\right) \ , \tag{5.16}$$

$${}_a\mathbf{R}_b = \mathbf{1}_3 + \sin\left(\theta\right)\left[\hat{\boldsymbol{\omega}}\right]_\times + \left(1 - \cos\left(\theta\right)\right)\left[\hat{\boldsymbol{\omega}}\right]_\times^2 \ , \tag{5.17}$$

where $\hat{\boldsymbol{\omega}}$ is the axis of rotation from ${}_w\hat{\mathbf{n}}_a$ to ${}_w\hat{\mathbf{n}}_b$, θ is the angle of rotation about $\hat{\boldsymbol{\omega}}$ from ${}_w\hat{\mathbf{n}}_a$ to ${}_w\hat{\mathbf{n}}_b$, and ${}_a\mathbf{R}_b$ is the rotation matrix from ${}_w\hat{\mathbf{n}}_a$ to ${}_w\hat{\mathbf{n}}_b$.

Validating Feature Pairs A face will afford another face if its normal is parallel and opposed to the normal of the other face. In 3D, an edge ${}_o\mathbf{g}_i$ joins two adjacent faces whose normals are denoted by ${}_o\hat{\mathbf{n}}_{e_1}$ and ${}_o\hat{\mathbf{n}}_{e_2}$. The vectors directed from the edge to the inside of

the face, along the face and orthogonal to the edge are given by

$$ {}_o\hat{\mathbf{s}}_{e_1} = \left({}_o\mathbf{g}_i \times {}_o\hat{\mathbf{n}}_{e_1}\right) / \left\| {}_o\mathbf{g}_i \times {}_o\hat{\mathbf{n}}_{e_1} \right\| \ , \tag{5.18} $$

$$ {}_o\hat{\mathbf{s}}_{e_2} = \left({}_o\mathbf{g}_i \times {}_o\hat{\mathbf{n}}_{e_2}\right) / \left\| {}_o\mathbf{g}_i \times {}_o\hat{\mathbf{n}}_{e_2} \right\| \ , \tag{5.19} $$

and termed the *side vectors* (see Figure 5.4b).

A face with outward normal ${}_o\hat{\mathbf{n}}$ will afford an edge if the normal is orthogonal to the edge direction ${}_o\mathbf{u}_e$ such that

$$ {}_o\mathbf{u}_e \cdot {}_o\hat{\mathbf{n}} = 0 \ , \tag{5.20} $$

and if the normal lies between the two side vectors of the edge. Intuitively, a planar surface with infinitesimal area should touch the edge without touching the faces on either side. A simple test to check if the normal lies between ${}_o\hat{\mathbf{s}}_{e_1}$ and ${}_o\hat{\mathbf{s}}_{e_2}$ is defined with

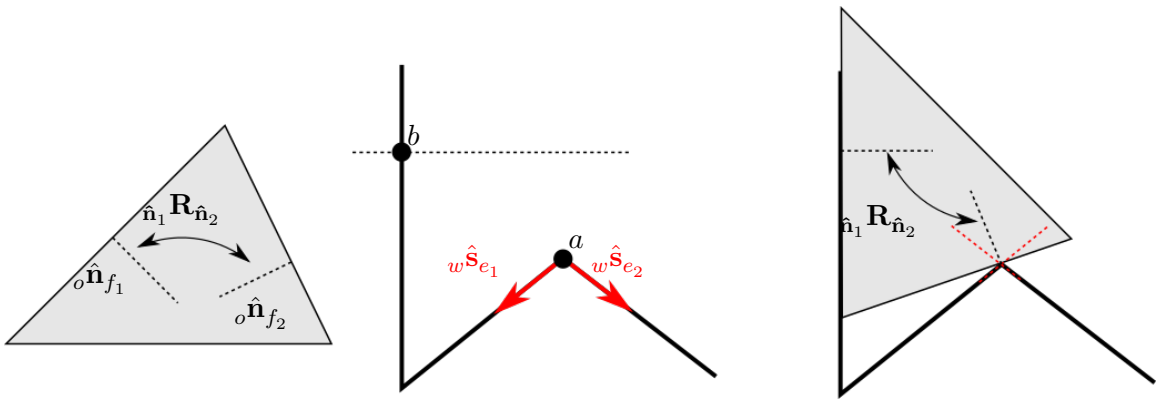
$$ u = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \left({}_o\hat{\mathbf{s}}_{e_1} \times {}_o\hat{\mathbf{s}}_{e_2}\right) \ , \tag{5.21} $$

$$ v = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \left({}_o\hat{\mathbf{n}} \times {}_o\hat{\mathbf{s}}_{e_2}\right) \ , \tag{5.22} $$

$$ w = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \left({}_o\hat{\mathbf{n}} \times {}_o\hat{\mathbf{s}}_{e_1}\right) \ , \tag{5.23} $$

such that the normal lies between the two side vectors if $uv \leq 0$ and $uw \geq 0$.

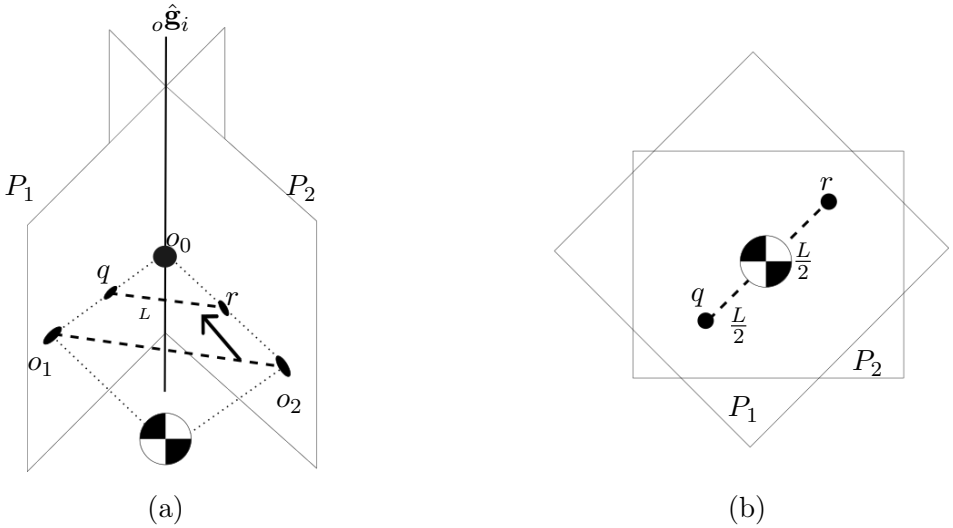
Since the normals sampled in the scene are expressed in a different frame than the ones on the object, the components of the normal vectors cannot be simply compared. Instead, the relative orientation of the object normals is considered to check if the scene can afford the object normals. For instance, assuming that the normal to the first feature selected on the object opposes ${}_w\hat{\mathbf{n}}_b$, the normal of the second feature needs to lie between the side vectors around point ${}_w\mathbf{p}_a$ when ${}_o\hat{\mathbf{n}}_{f_1}$ opposes ${}_w\hat{\mathbf{n}}_b$, as shown in Figure 5.4.



(a) Object (b) Scene (c) Object Afforded

Figure 5.4: Affordance test to see if ${}_o\hat{\mathbf{n}}_{f_2}$ would lie between ${}_w\hat{\mathbf{s}}_{e_1}$ and ${}_w\hat{\mathbf{s}}_{e_2}$ when ${}_o\hat{\mathbf{n}}_{f_1}$ opposes ${}_w\hat{\mathbf{n}}_b$.

Several cases are considered depending on the relative orientation of the features. In



(a) (b)

Figure 5.5: Face-face pair with (a) intersecting faces and (b) coplanar faces.

each case, the points on the object are selected such that the distance between them is L and that the line joining them is as close as possible to the centre of mass of the object.

A. Face-Face Pair Two faces sampled on the object can be parallel, coplanar, or intersecting, as shown in Figure 5.5.

Let two planes be defined by

$$P_1(d_1) : {}_o\hat{\mathbf{n}}_{f_1} \cdot \mathbf{p} = d_1 \ , \tag{5.24}$$

$$P_2(d_2) : {}_o\hat{\mathbf{n}}_{f_2} \cdot \mathbf{p} = d_2 \ , \tag{5.25}$$

where $\mathbf{p}$ is a point on the plane, ${}_o\hat{\mathbf{n}}_f$ is the normal of the face plane, and d is the distance of the plane from the origin. If the faces are parallel but not coplanar, the distance between the faces $(d_1 - d_2)\,{}_o\hat{\mathbf{n}}_{f_2}$ would need to be exactly equal to L for the match to be valid. This is almost never the case and would produce infinitesimal interpenetrations (i.e., friction) when placing the object.

A.1. Coplanar Faces If the faces are coplanar, there is an infinite number of point pairs that are distanced by L. Amongst the possible pairs, we select one that minimizes the distance between the centre of mass of the object and the line joining the two points to maximize the odds of sampling a stable pose. Let ${}_o\mathbf{u}$ be in the direction given by the extent of the object projected onto the common plane of the two faces. Selecting points along the direction vector that are at a distance $\pm L/2$ from the projection of the centre of mass onto

the face plane is done with

$$ {}_o\mathbf{p}_{o_1} = d_1 \, {}_o\hat{\mathbf{n}}_{f_1} + {}_o\mathbf{p}_c - \left({}_o\mathbf{p}_c \cdot {}_o\hat{\mathbf{n}}_{f_1} \right) {}_o\hat{\mathbf{n}}_{f_1} \ , \tag{5.26} $$

$$ {}_o\mathbf{p}_q = {}_o\mathbf{p}_{o_1} + \frac{L}{2} \, {}_o\mathbf{u} \ , \tag{5.27} $$

$$ {}_o\mathbf{p}_r = {}_o\mathbf{p}_{o_1} - \frac{L}{2} \, {}_o\mathbf{u} \ , \tag{5.28} $$

where ${}_o\mathbf{p}_c$ is the centre of mass of the object expressed in the object frame. In (5.26), ${}_o\mathbf{p}_{o_1}$ is the projection of the centre of mass onto the face plane, while ${}_o\mathbf{p}_q$ and ${}_o\mathbf{p}_r$ are respectively the point on the first and second face being considered.

A.2. Intersecting Faces If the faces are intersecting, they share a common edge whose direction is given by

$$ {}_o\hat{\mathbf{g}}_e = {}_o\hat{\mathbf{n}}_{f_1} \times {}_o\hat{\mathbf{n}}_{f_2} / \left\| {}_o\hat{\mathbf{n}}_{f_1} \times {}_o\hat{\mathbf{n}}_{f_2} \right\| \ , \tag{5.29} $$

and whose origin ${}_o\mathbf{p}_e$ is defined by

$$ \mathbf{u} = {}_o\hat{\mathbf{n}}_{f_1} \times {}_o\hat{\mathbf{g}}_e \ , \tag{5.30} $$

$$ {}_o\mathbf{p}_e = {}_o\mathbf{p}_{o_1} + \frac{d_2 - {}_o\hat{\mathbf{n}}_{f_2} \cdot {}_o\mathbf{p}_{o_1}}{{}_o\hat{\mathbf{n}}_{f_2} \cdot \mathbf{u}} \mathbf{u} \ , \tag{5.31} $$

where ${}_o\mathbf{p}_{o_1}$ is the projection of the centre of mass onto the plane of the first face, as defined previously. The plane orthogonal to the common edge and passing through the centre of mass of the object intersects ${}_o\hat{\mathbf{g}}_e$ at

$$ {}_o\mathbf{p}_{o_0} = {}_o\mathbf{p}_e + \left({}_o\mathbf{p}_c \cdot {}_o\hat{\mathbf{g}}_e \right) {}_o\hat{\mathbf{g}}_e \tag{5.32} $$

such that a triangle is formed by ${}_o\mathbf{p}_{o_0}$, ${}_o\mathbf{p}_{o_1}$, and ${}_o\mathbf{p}_{o_2}$ in the plane passing through ${}_o\mathbf{p}_c$. A similar triangle connecting ${}_o\mathbf{p}_{o_0}$, ${}_o\mathbf{p}_q$, ${}_o\mathbf{p}_r$ whose leg joining both faces has length L can be defined with

$$ {}_o\mathbf{p}_q = {}_o\mathbf{p}_{o_0} + \frac{L}{\left\| {}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{o_2} \right\|} \left({}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{o_0} \right) \ , \tag{5.33} $$

$$ {}_o\mathbf{p}_r = {}_o\mathbf{p}_{o_0} + \frac{L}{\left\| {}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{o_2} \right\|} \left({}_o\mathbf{p}_{o_2} - {}_o\mathbf{p}_{o_0} \right) \ , \tag{5.34} $$

where ${}_o\mathbf{p}_{o_2}$ is the projection of the centre of mass onto the plane of the second face, and ${}_o\mathbf{p}_q$ and ${}_o\mathbf{p}_r$ are respectively the point on the first and second face being considered.

B. Face-Edge Pair In the situation where one interface is between two faces and the other interface is between a face and an edge, the line vector connecting the two contact points on the object will join a point on the face to a point on the edge, as shown in

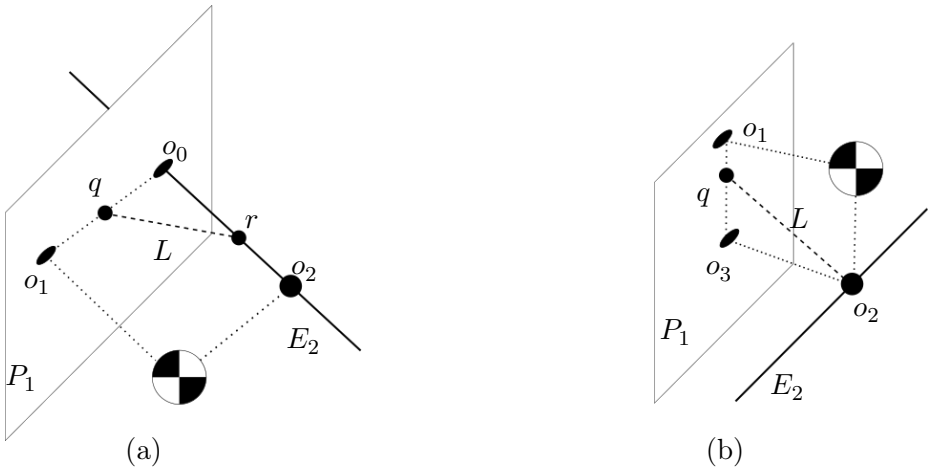


Figure 5.6: Face-edge pair: (a) intersecting, (b) parallel.

Figure 5.6. Let the plane of the face/face interface be defined by

$$P_1(d) : {}_o\hat{\mathbf{n}}_f \cdot {}_o\mathbf{p}_q = d \ , \tag{5.35}$$

and the edge be defined by

$$E_2(t) : {}_o\mathbf{p}_r = {}_o\mathbf{p}_e + t\, {}_o\mathbf{u}_e \ , \tag{5.36}$$

such that **q** is constrained to lie on the plane of the face/face interface and **r** is constrained to lie on the edge.

The edge can either be parallel to the face plane or intersecting it. In both cases, a triangle is defined on a plane that is passing through the centre of mass of the object to minimize the odds of sampling an unstable pose, as done in the face-face pair case.

B.1. Intersecting If the edge intersects the plane of the face, the intersection point ${}_o\mathbf{p}_{o_0}$ is given by

$$t = \frac{d - {}_o\hat{\mathbf{n}}_f \cdot {}_o\mathbf{p}_e}{{}_o\hat{\mathbf{n}}_f \cdot {}_o\mathbf{u}_e} \tag{5.37}$$

$${}_o\mathbf{p}_{o_0} = {}_o\mathbf{p}_e + t\, {}_o\mathbf{u}_e \ . \tag{5.38}$$

The projections of the centre of mass onto the face and edge are given by

$${}_o\mathbf{p}_{o_1} = d\, {}_o\hat{\mathbf{n}}_f + {}_o\mathbf{p}_c - \left({}_o\mathbf{p}_c \cdot {}_o\hat{\mathbf{n}}_f\right) {}_o\hat{\mathbf{n}}_f \ , \tag{5.39}$$

$${}_o\mathbf{p}_{o_2} = {}_o\mathbf{p}_e + \left(({}_o\mathbf{p}_c - {}_o\mathbf{p}_e) \cdot {}_o\mathbf{u}_e\right) {}_o\mathbf{u}_e \ , \tag{5.40}$$

respectively, such that a triangle is formed by ${}_o\mathbf{p}_{o_0}$, ${}_o\mathbf{p}_{o_1}$, and ${}_o\mathbf{p}_{o_2}$ in the plane passing through ${}_o\mathbf{p}_c$. The method employed in the face-face pair case can then be used to find ${}_o\mathbf{p}_q$ and ${}_o\mathbf{p}_r$.

B.2. Parallel If the edge is parallel to the face plane, ${}_o\mathbf{p}_r$ is set to the projection of the centre of mass onto the edge with

$$ {}_o\mathbf{p}_r = {}_o\mathbf{p}_{o_2} \ . \tag{5.41}$$

In the case of the edge not lying in the plane of the face, ${}_o\mathbf{p}_{o_2}$ is moved to the closest point on the face with

$$ {}_o\mathbf{p}_{o_3} = d\,{}_o\hat{\mathbf{n}}_f + {}_o\mathbf{p}_{o_2} - \left({}_o\mathbf{p}_{o_2} \cdot {}_o\hat{\mathbf{n}}_f\right) {}_o\hat{\mathbf{n}}_f \ , \tag{5.42}$$

where ${}_o\mathbf{p}_{o_1}$ and ${}_o\mathbf{p}_{o_2}$ are the projections of the centre of mass onto the face and edge respectively as defined previously. The location of the point on the face is then given by

$$ {}_o\mathbf{p}_q = {}_o\mathbf{p}_{o_3} + \frac{\sqrt{\left(L^2 - \left\|{}_o\mathbf{p}_{o_3} - {}_o\mathbf{p}_{o_2}\right\|^2\right)}\left({}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{o_3}\right)}{\left\|{}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{o_3}\right\|} \tag{5.43}$$

such that the projection of the centre of mass onto the face plane lies on the line joining ${}_o\mathbf{p}_q$ to ${}_o\mathbf{p}_r$, as desired.

C. Edge-Edge Pair The line vectors of two sampled edges can either be parallel, intersecting, or skew, as shown in Figure 5.7. For two edge lines defined by their parametric equations

$$ E_1(t_1) : {}_o\mathbf{p}_q = {}_o\mathbf{p}_{e_1} + t_1\,{}_o\mathbf{u}_{e_1} \ , \tag{5.44}$$

$$ E_2(t_2) : {}_o\mathbf{p}_r = {}_o\mathbf{p}_{e_2} + t_2\,{}_o\mathbf{u}_{e_2} \ , \tag{5.45}$$

the lines are parallel if ${}_o\mathbf{u}_{e_1} \times {}_o\mathbf{u}_{e_2} = \mathbf{0}$.

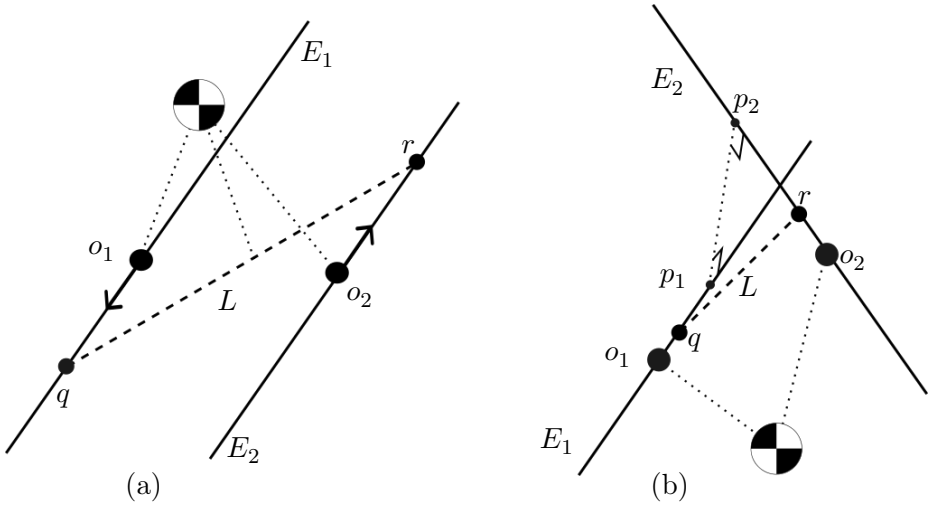


Figure 5.7: Edge-edge pair: (a) parallel, (b) skew.

Unless edges are parallel, there are two points, ${}_o\mathbf{p}_{p_1} = E_1(t_1)$ and ${}_o\mathbf{p}_{p_2} = E_2(t_2)$, that

are a minimum distance apart. These points are given for parameters

$$t_1 = \frac{({}_o\mathbf{u}_{e_1} \cdot {}_o\mathbf{u}_{e_2})({}_o\mathbf{u}_{e_2} \cdot {}^{o}_{e_2}\mathbf{p}_{e_1}) - ({}_o\mathbf{u}_{e_1} \cdot {}^{o}_{e_2}\mathbf{p}_{e_1})}{({}_o\mathbf{u}_{e_1} \cdot {}_o\mathbf{u}_{e_2})(1 - {}_o\mathbf{u}_{e_1} \cdot {}_o\mathbf{u}_{e_2})} , \tag{5.46}$$

$$t_2 = \frac{({}_o\mathbf{u}_{e_2} \cdot {}^{o}_{e_2}\mathbf{p}_{e_1}) - ({}_o\mathbf{u}_{e_1} \cdot {}^{o}_{e_2}\mathbf{p}_{e_1})({}_o\mathbf{u}_{e_1} \cdot {}_o\mathbf{u}_{e_2})}{({}_o\mathbf{u}_{e_1} \cdot {}_o\mathbf{u}_{e_2})(1 - {}_o\mathbf{u}_{e_1} \cdot {}_o\mathbf{u}_{e_2})} , \tag{5.47}$$

respectively, where ${}^{o}_{e_2}\mathbf{p}_{e_1} = {}_o\mathbf{p}_{e_1} - {}_o\mathbf{p}_{e_2}$ is the vector joining the origins of the two edges (Ericson, 2004). If the distance between the two closest points is non-zero, the two edges are skew. The projection of the centre of mass onto E_1 and E_2 is given by

$${}_o\mathbf{p}_{o_1} = {}_o\mathbf{p}_{e_1} + \left(({}_o\mathbf{p}_c - {}_o\mathbf{p}_{e_1}) \cdot {}_o\mathbf{u}_{e_1}\right) {}_o\mathbf{u}_{e_1} , \tag{5.48}$$

$${}_o\mathbf{p}_{o_2} = {}_o\mathbf{p}_{e_2} + \left(({}_o\mathbf{p}_c - {}_o\mathbf{p}_{e_2}) \cdot {}_o\mathbf{u}_{e_2}\right) {}_o\mathbf{u}_{e_2} , \tag{5.49}$$

respectively.

C.1. Parallel Edge Lines Similar to previous scenarios, the objective is to select points such that the distance between the object centre of mass and the line joining the two points is minimized, to increase the odds of sampling a stable pose.

For parallel lines, the points ${}_o\mathbf{p}_q$ and ${}_o\mathbf{p}_r$ are selected by being equally distanced from their respective centre of mass projections in opposite directions with

$${}_o\mathbf{p}_q = {}_o\mathbf{p}_{o_1} + \frac{1}{2}\sqrt{L^2 - \left\|{}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{o_2}\right\|^2}\, {}_o\mathbf{u}_{e_1} , \tag{5.50}$$

$${}_o\mathbf{p}_r = {}_o\mathbf{p}_{o_2} - \frac{1}{2}\sqrt{L^2 - \left\|{}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{o_2}\right\|^2}\, {}_o\mathbf{u}_{e_1} , \tag{5.51}$$

such that the two points are distanced by L. If $L < \left\|{}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{o_2}\right\|$, ${}_o\mathbf{p}_q$ and ${}_o\mathbf{p}_r$ are set to the projections of the centre of mass onto the edges.

C.2. Intersecting Edge Lines If the edges are intersecting, the intersection point will form a triangle with points ${}_o\mathbf{p}_q$ and ${}_o\mathbf{p}_r$ on the edges. However, the two edges are not necessarily part of the same face on non-convex objects.

The intersection point of the two edges ${}_o\mathbf{p}_{o_0}$ can be obtained with (5.46) by setting either t_1 or t_2 in its respective line equation. The points ${}_o\mathbf{p}_q$ and ${}_o\mathbf{p}_r$ can be obtained with (5.33) from ${}_o\mathbf{p}_{o_0}$, and the points ${}_o\mathbf{p}_{o_1}$, ${}_o\mathbf{p}_{o_2}$ in (5.48) and (5.49).

C.3. Skew Edge Lines With skew edges, the closest point on each edge to the other edge, given by (5.46), define a line vector ${}^{o}_{p_2}\mathbf{p}_{p_1} = {}_o\mathbf{p}_{p_1} - {}_o\mathbf{p}_{p_2}$ that is orthogonal to both edges. Hence, edges lie in parallel planes that are orthogonal to ${}^{o}_{p_2}\mathbf{p}_{p_1}$ and separated by a distance $\left\|{}^{o}_{p_2}\mathbf{p}_{p_1}\right\|$. Therefore, the distance between $E_1(t_1)$ and $E_2(t_2)$ is given by

$$\|E_2(t_2) - E_1(t_1)\| = \sqrt{D^2 + \left\|{}^{o}_{p_2}\mathbf{p}_{p_1}\right\|^2} , \tag{5.52}$$

where D is the distance along the parallel planes, such that the distance between ${}_o\mathbf{p}_{o_1}$ and

${}_o\mathbf{p}_{o_2}$ along the parallel planes is given by $\sqrt{\left\|{}^o_{o_2}\mathbf{p}_{o_1}\right\|^2 - \left\|{}^o_{p_2}\mathbf{p}_{p_1}\right\|^2}$. Similar to how the position of the contact points were defined in (5.33), the points ${}_o\mathbf{p}_q$ and ${}_o\mathbf{p}_r$ are defined with

$$ {}_o\mathbf{p}_q = {}_o\mathbf{p}_{p_1} + \frac{\left({}_o\mathbf{p}_{o_1} - {}_o\mathbf{p}_{p_1}\right)\sqrt{L^2 - \left\|{}^o_{p_2}\mathbf{p}_{p_1}\right\|^2}}{\sqrt{\left\|{}^o_{o_2}\mathbf{p}_{o_1}\right\|^2 - \left\|{}^o_{p_2}\mathbf{p}_{p_1}\right\|^2}} , \tag{5.53} $$

$$ {}_o\mathbf{p}_r = {}_o\mathbf{p}_{p_2} + \frac{\left({}_o\mathbf{p}_{o_2} - {}_o\mathbf{p}_{p_2}\right)\sqrt{L^2 - \left\|{}^o_{p_2}\mathbf{p}_{p_1}\right\|^2}}{\sqrt{\left\|{}^o_{o_2}\mathbf{p}_{o_1}\right\|^2 - \left\|{}^o_{p_2}\mathbf{p}_{p_1}\right\|^2}} , \tag{5.54} $$

that will produce a solution unless $\left\|{}^o_{p_2}\mathbf{p}_{p_1}\right\| > L$, in which case the edges are too far apart to produce a valid match.

5.5.3 Determination of the Object Pose

Let ${}_o\hat{\mathbf{n}}_{e_1}$ be equal to the average of the two face normals that E_1 is joining. By definition, ${}_o\hat{\mathbf{n}}_{e_1}$ will be in the plane orthogonal to E_1. For a placement to be valid, the normal at the scene contact point ${}_w\hat{\mathbf{n}}_a$ must also lie in the plane orthogonal to the edge. Moreover, the angle between ${}_o\hat{\mathbf{n}}_{e_1}$ and ${}_w\hat{\mathbf{n}}_a$ must be less than 90° for the placement to be non-penetrating. Unless the two edges are collinear, we have that ${}^w_b\mathbf{p}_a \times {}_w\hat{\mathbf{n}}_a \neq 0$ such that there exists a vector

$$ {}_w\hat{\mathbf{u}} = \frac{{}^w_b\mathbf{p}_a \times {}_w\hat{\mathbf{n}}_a}{\left\|{}^w_b\mathbf{p}_a \times {}_w\hat{\mathbf{n}}_a\right\|} \tag{5.55} $$

that is orthogonal to the scene face and to ${}^w_b\mathbf{p}_a$, the direction of the line vector connecting the two contact points. The vector in the object frame that is analogous to ${}_w\hat{\mathbf{u}}$ is given by

$$ {}_o\hat{\mathbf{u}} = -\frac{{}^o_r\mathbf{p}_q \times {}_o\hat{\mathbf{n}}_{e_1}}{\left\|{}^o_r\mathbf{p}_q \times {}_o\hat{\mathbf{n}}_{e_1}\right\|} , \tag{5.56} $$

such that

$$ {}_w\hat{\mathbf{u}} = {}_w\mathbf{R}_o \, {}_o\hat{\mathbf{u}} \tag{5.57} $$

relates the two vectors.

Defining basis vectors

$$ {}_w\hat{\mathbf{w}} = \frac{{}_w\hat{\mathbf{u}} \times {}^w_b\mathbf{p}_a}{\left\|{}_w\hat{\mathbf{u}} \times {}^w_b\mathbf{p}_a\right\|} , \tag{5.58} $$

$$ {}_o\hat{\mathbf{w}} = \frac{{}_o\hat{\mathbf{u}} \times {}^o_r\mathbf{p}_q}{\left\|{}_o\hat{\mathbf{u}} \times {}^o_r\mathbf{p}_q\right\|} , \tag{5.59} $$

a frame fixed in the scene can be defined with

$$ \boldsymbol{\mathcal{F}}_s = \begin{bmatrix} {}_w\hat{\mathbf{u}} & \frac{{}^w_b\mathbf{p}_a}{\left\|{}^w_b\mathbf{p}_a\right\|} & {}_w\hat{\mathbf{w}} \end{bmatrix}^\mathsf{T} . \tag{5.60} $$

A frame fixed in the object can be obtained with

$$\boldsymbol{\mathcal{F}}_o = \begin{bmatrix} {}_o\hat{\mathbf{u}} & \frac{{}^o_r\mathbf{p}_q}{\left\|{}^o_r\mathbf{p}_q\right\|} & {}_o\hat{\mathbf{w}} \end{bmatrix}^{\mathsf{T}} , \tag{5.61}$$

such that

$${}_w\mathbf{R}_o = \boldsymbol{\mathcal{F}}_s \boldsymbol{\mathcal{F}}_o^{\mathsf{T}} \tag{5.62}$$

is the orientation of the object in the world frame. With the knowledge of ${}_w\mathbf{R}_o$, the position of the object can subsequently be obtained with

$${}_w\mathbf{p}_o = {}_w\mathbf{p}_a - {}_w\mathbf{R}_o \, {}_o\mathbf{p}_q , \tag{5.63}$$

where ${}_w\mathbf{p}_a$ is the position of the first scene contact point.

5.5.4 Pose Validation

Once a pose is determined, it undergoes a series of checks to ensure that the object does not interpenetrate other objects and does not destabilize the assembly. First, a collision detection algorithm verifies that the object avoids penetrating other assembly components, and detects contact interfaces on which the object is resting. To determine the location of contact points, we use the approach from (Mattikalli et al., 1996) and select the vertices of the convex hull over the contact interface as contact points. Reaction forces at the contact points are then computed using the QR-based method in (5.8) and the maximal tension force is compared to a threshold value (e.g., 5 Newtons) to avoid performing the more computationally expensive QP-based method when the placement is clearly unstable. Finally, the forces at the contact points are computed using (5.2), the equilibrium of the assembly is verified, and the static robustness of the assembly is updated.

5.6 Simulation Experiments

We perform a series of experiments across six different scenes, shown in Figure 5.8, where the task is to place a cube such that it stably rests on other objects. Each scene is designed to challenge the algorithms in different ways. The *Stack* scene has many points on vertical surfaces, *Pyramids* has few strong contact points, *Table* is prone to toppling, *Sawteeth* has no horizontal surfaces, and *Canyon* only affords placements resting on both objects. The benchmarked algorithms and evaluation criteria are described in Section 5.6.1 and Section 5.6.2 respectively, and the results are reported in Table 5.1. For all sample-based planners, fifty consecutive experiments are performed for each scene with a maximum of 500 attempts per experiment. We evaluate all algorithms in terms of planning time, assembly robustness, packing density, and success rate.

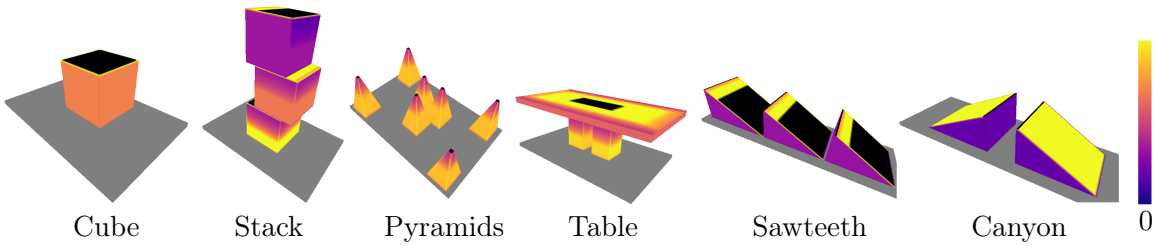


Figure 5.8: The six scenes used for the experiments. The surface colour indicates the relative magnitude of the maximal normal force that can be sustained, with a lighter colour indicating a larger force and black indicating an infinite force.

5.6.1 Benchmarked Algorithms

Planner Using Static Robustness (Ours-SR)

We compare to other methods our proposed algorithm defined in Section 5.5 and using (5.10) to sample points, with the initial probability of sampling the most robust point set to 10% and the exponential decay rate λ set to 0.99. In all experiments involving contact point sampling, if multiple pairs of features can be matched to the sample points, a single random pair is selected.

Planner With Uniform Sampling (Ours-Uniform)

In essence, our algorithm defined in Section 5.5 uses the proposed static robustness heuristic to increase the odds of sampling promising contact points in the scene. Insights about the effectiveness of the proposed algorithm can be obtained by comparing it with a variation for which the probability of sampling a point is uniform. If there is a significant benefit to using the static robustness heuristic, the proposed algorithm should perform better when the probability distribution in (5.10) is used, and worst when it is uniform.

Dynamics Simulation From Above (Sim-Above)

A dynamics simulator is also used to provide a standard for comparison. Object placement from simulations is performed in two phases during which the restitution coefficient is set to 0, and the dynamics are heavily damped to reduce any bouncing phenomena. In the first phase, the object is dropped in a random orientation with its centre of mass at a random position slightly above the non-fixed objects in the scene. The gravitational acceleration is set to 0.1 $\mathrm{m/s^2}$ and the downward velocity of the object is set to 0.01 m/s to reduce the kinetic energy of the object when it collides with the scene. The simulation is run until the object comes to rest and is aborted if the object falls off the scene floor. In the second phase, gravity is increased to 9.81 $\mathrm{m/s^2}$ and the simulation is ran for 100 iterations to ensure that the placement is stable under standard gravitational acceleration. If the placed object

rests on at least one other object, and no object has moved significantly, the placement is considered to be successful.

Dynamics Simulation From A Random Pose (Sim-Random)

Performing simulations from initial poses above the scene objects will not allow the planner to find placement poses below other objects. A more general approach is to perform simulations from random poses, where the pose of the object is initialized randomly within the vicinity of the scene objects. When sampling positions, the extent of the scene was defined from the bounding box of the objects in the scene and points around the scene, up to a distance equal to the radius of the sphere circumscribing the object, were considered. Otherwise, all parameters are kept the same as in the *Sim-above* algorithm.

Heightmap Minimization (HM)

The algorithm defined in (Wang and Hauser, 2021) is implemented and used as a benchmark. A summary of the algorithm is provided below. Planning with the *HM* algorithm is performed in four steps: (i) candidate pose generation, (ii) scoring, (iii) stability evaluation, and potentially (iv) refined search. The first step uses (Goldberg et al., 1999) to generate four object orientations that are likely stable on planar surfaces, and perturb the orientations with rotations about the vertical axis. The scene is voxelized and a set of candiate poses is generated by finding the lowest point in the voxel grid for each object orientation. The second step scores the candidate poses with a heightmap minimization heuristic proposed in (Wang and Hauser, 2021) that encourages dense and stable placements. The third step iterates over the highest scoring poses and evaluates their stability by solving (5.2) and returning the first stable pose found. Finally, in the event that no stable pose is found, the search is refined by generating a larger set of candidate poses through rotation perturbations about orthogonal axes in the horizontal plane. Our implementation of the *HM* algorithm uses the same parameters and the same voxel resolution as in (Wang and Hauser, 2021).

Random Pose (Chance)

To provide a baseline for comparison, a random pose within the vicinity of the scene objects is selected for the object, and the stability of the placement is evaluated. Although this method is expected to be the slowest, it should be capable of finding stable placements if enough iterations are performed (although a maximum of 500 attempts are made in the experiments).

5.6.2 Evaluation Criteria

Five criteria are used to evaluate the performance of the algorithms.

Placement Time. For each set of experiments, the average time required to find a collision-free stable placement pose is measured. Since some algorithms may require more iterations but less compute time per iteration, all algorithms are compared in terms of the average compute time per valid placement.

Assembly Robustness. The objective of inertia-aware object placement planning is to find placements that maximize the force required to displace any object in the assembly. Hence, the minimal magnitude of the external wrench that would destabilize the assembly is computed according to (Maeda et al., 2009; Chen et al., 2021), where the wrench directions are defined as the orthogonal axes in the 6D wrench space.

Minimum Static Robustness. Our static robustness heuristic defined in Section 5.4 is computed for points on all objects in the scene following a valid placement. The minimum static robustness, representing the smallest force that can be applied to the surface of an object such that the assembly is destabilized, is computed for each experiment.

Packing Density. Since it is often desired to place objects such that the assembly is dense, the packing density of the assembly is computed as the volume of the oriented bounding box of the assembly following the placement.

Success Rate. The proportion of experiments in which a valid placement was found is reported for each algorithm.

		Chance	Sim-Random	Sim-Above	HM	Ours-Uniform	Ours-SR
Cube	Time	2.85	11.23	0.79	0.88	3.60	**0.38**
	Rob.	—	1.72	1.99	**3.21**	1.64	1.81
	Min. SR	—	2.29	2.64	**4.44**	2.21	2.47
	Volume	—	30.0	28.4	**17.2**	30.1	28.0
Stack	Time	4.88	54.5	7.76	**0.56**	120	1.51
	Rob.	—	0.48	0.41	**0.80**	0.53	0.56
	Min. SR	—	1.76	1.88	**2.40**	2.10	2.00
	Volume	—	70.6	66.7	**47.3**	58.0	61.1
Table	Time	5.13	20.8	6.59	0.83	13.8	**0.40**
	Rob.	—	1.76	1.85	1.11	1.86	**1.95**
	Min. SR	—	2.10	2.10	2.10	1.65	2.10
	Volume	—	105	103	109	**88.3**	101
Pyramids	Time	6.47	97.25	41.74	**1.10**	18.62	1.71
	Rob.	—	0.78	0.75	**0.85**	0.49	0.40
	Min. SR	—	**1.69**	1.10	1.54	0.41	0.42
	Volume	—	80.1	**72.4**	128	96.8	135
Sawteeth	Time	4.30	9.43	8.74	30.0	10.4	**2.86**
	Rob.	—	0.71	0.64	—	0.91	0.77
	Min. SR	—	1.91	1.80	—	1.58	1.73
	Volume	—	**136**	139	—	161	159
Canyon	Time	4.02	55.2	45.7	31.6	9.85	**1.30**
	Rob.	—	0.85	0.88	—	1.40	**1.68**
	Min. SR	—	0.57	0.29	—	0.52	**0.72**
	Volume	—	109	**108**	—	130	128
Average Time		4.61	41.40	18.55	10.83	29.38	**1.36**
Success Rate		0%	93.3%	99.7%	66.7%	95.3%	**100%**

Table 5.1: Average placement time in seconds, robustness to external wrench (Rob.) in Newtons, minimum static robustness (Min. SR) in Newtons, and volume of the assembly in cubic metres for each algorithm across our six scenes. Entries with — indicate that no valid placement was found in the experiments.

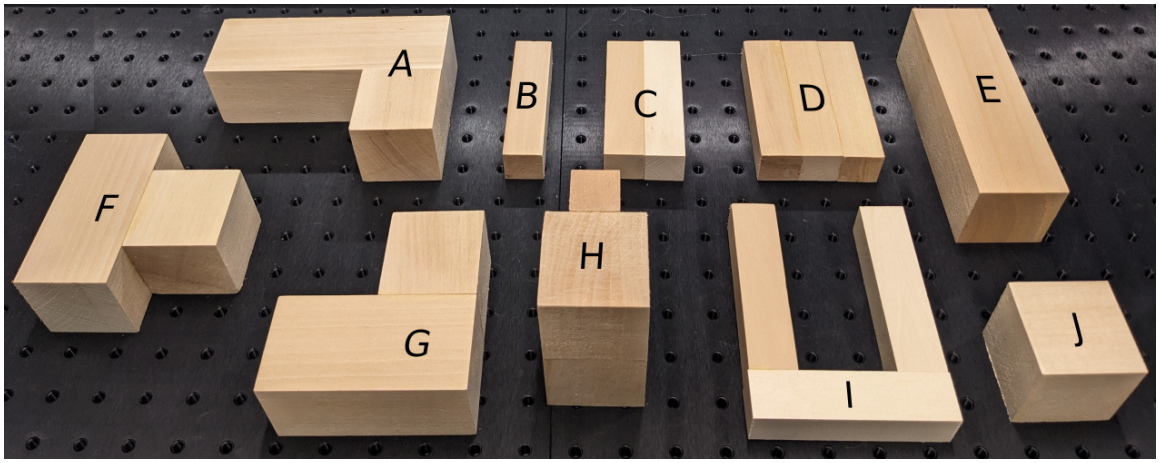


Figure 5.9: The set of 10 basswood objects used for the real robot experiments.

5.7 Real Robot Experiments

A practical placement planner should produce stable assemblies under the presence of uncertainty in the pose, shape, and inertial parameters of the objects. To evaluate the effectiveness of the proposed algorithm in such a scenario, we performed experiments with a Franka Research 3 robot arm equipped with a Robotiq 2F-85 gripper. A set of 10 non-convex basswood objects, pictured in Figure 5.9, is used for this experiment, where each object is built from manually-glued cuboids resulting in shapes that are slightly different from the models used by the planner. The mass density of the objects is assumed to be 415 kg/m^3, which is expected to differ from the actual value, introducing errors in the inertial models of the objects. In our experiments, the friction coefficient was assumed to be 0.5 at interfaces between our basswood objects, which is expected to be an optimistic value. In practical applications where stability is critical, the friction coefficient can be assumed to be lower than its nominal value such that the planner acts more conservatively. Uncertainty in the initial pose of the objects, slippage, and Cartesian trajectory errors are expected to create a discrepancy between the planned and executed placements, potentially resulting in unstable assemblies.

A total of 50 placements from 10 experiments pictured in Figure 5.10 (E1 to E10) are performed with the robot. Each experiment consists of iteratively placing objects in the scene until five objects are placed. For each placement, the proposed planner is ran 10 times, each time executing at most 20 iterations with an object selected randomly without replacement and with odds proportional to its mass (i.e., heavier objects are more likely to be placed first). For these experiments, sampling on the fixed support is allowed and (5.14) is used with $\gamma = 0.5$ and with other parameters set to the same values as in the simulation experiments. If more than one placement results in the same minimum SR, the one with the largest median SR is selected in E1 – E5, and the one with the smallest assembly volume is selected in E6 – E10.

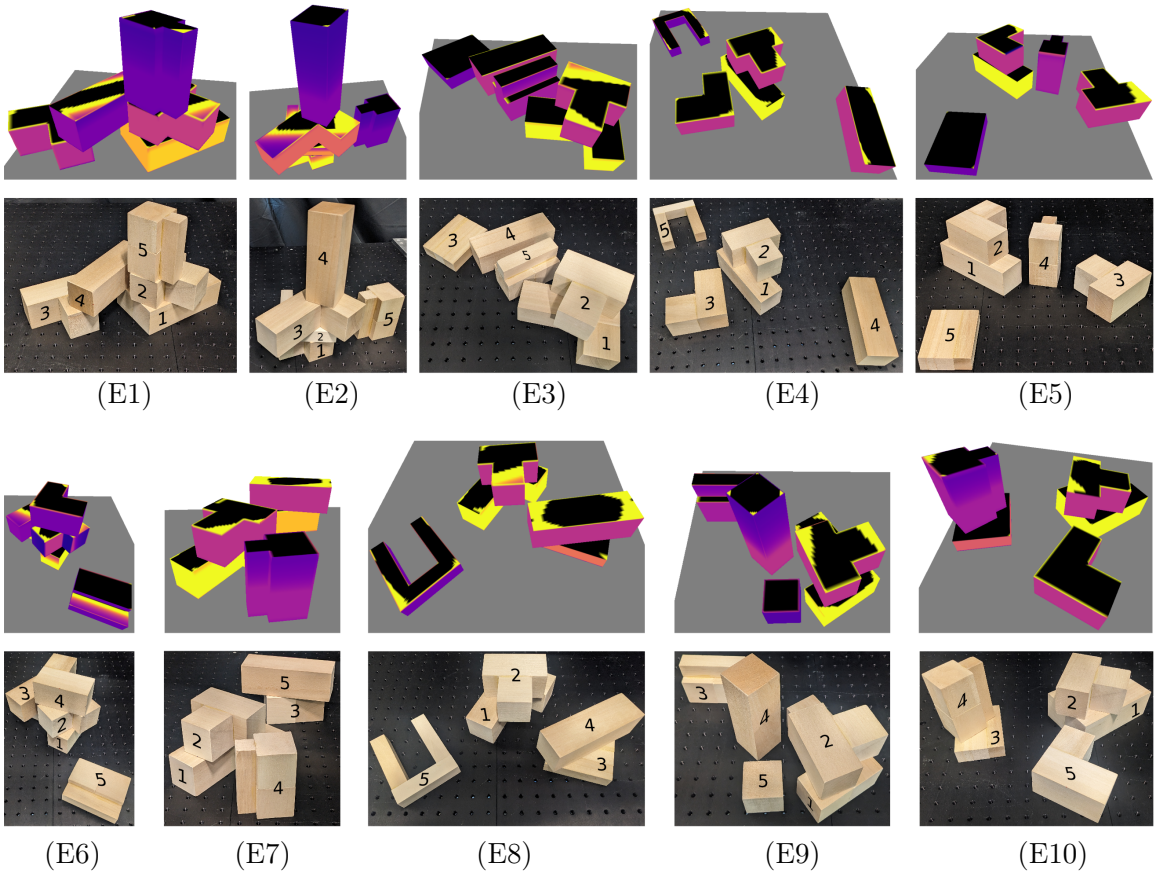


Figure 5.10: The ten scenes from the real robot experiments with the SR map in the top row and the final assembly in the bottom row. The order in which objects are placed is indicated by the numbers on the objects.

With the placement pose defined, a feasible grasp as far as possible to the contact points is determined. For each placement, the RRTConnect (Kuffner and LaValle, 2000) motion planner was run twice, each time for at most 10 seconds, to find collision-free trajectories to pick up and place the object.

The success of any placement depends on five cascading steps: (i) finding a feasible placement pose, (ii) grasping the object, (iii) planning a collision-free motion trajectory, (iv) executing the placement without slippage or collisions, and (v) keeping the assembly stable. During our experiments, human observation was used to determine the outcome of each step and any issue that arose (e.g. slip, contact with the environment, toppled assembly) was noted. The human observer manually carried out any uncompleted placement by approximately placing the object in its goal pose such that the following placements could be subsequently performed by the robot. The success rate of each step relative to the number of placements that reached the step is reported in Table 5.2.

Step	Success Rate
Placement Planning	50/50 = 100%
Grasping	41/50 = 82%
Motion Planning	37/41 = 90.2%
Placement Execution	33/37 = 89.2%
Stable Assembly	33/33 = 100%

Table 5.2: Success rate of each step in the real robot experiments with 50 placements, computed relative to the number of successful attempts from the previous step.

5.8 Discussion

The performance of the proposed algorithm in comparison with other algorithms is analyzed in Section 5.8.1. Observations made during the real robot experiments are outlined in Section 5.8.2 and the computational complexity of the proposed algorithm is analyzed in Section 5.8.3.

5.8.1 Performance Analysis

In our experiments, *Chance* never found a valid placement in the 3,000 attempts made, indicating that the placement planning problem is not trivial.

According to Table 5.1, the proposed algorithm using static robustness was about 20 times faster to find a stable placement when compared to the same algorithm without static robustness and about eight times faster than the *HM* algorithm. The simulator also took much longer than the proposed algorithm to find a valid solution, on average taking about 13.6 times longer with *Sim-Above* and 30 times longer with *Sim-Random*. This poor performance is expected, given that the simulator needlessly solves a much more complex problem (i.e., coupled kinematics and dynamics) (Kaufman, 2009).

Compared to other algorithms, *HM* can find placements that are more robust to external wrenches. However, *HM* found valid placements in only 66.7% of the experiments, while the other algorithms (except *Chance*) found valid placements in 93% to 100% of the experiments. Even though *HM* optimizes for packing density while *Ours-SR* does not, both algorithms produce assemblies with similar volumes. In contrast, the other algorithms can produce better packed assemblies by having the object be supported by the floor (e.g. under the table).

Algorithms that sample uniformly struggle in large scenes that afford only few placements, as seen in the *Stack* scene where the time required by *Sim-Random* and *Ours-Uniform* is significantly higher than for the other algorithms. In the *Table* scene, the robustness to external wrench is significantly lower for *HM* than for the other algorithms since *HM* selects the first stable pose found, which is not the one in the centre of the table. However, in

Pyramids, *Ours-SR* finds placements with relatively low robustness compared to *HM* since the former does not center the cube on the three pyramids. By virtue of the three pyramids being identical, *HM* produces better centered placements, resulting in a more robust assembly. In the *Sawteeth* and *Canyon* scenes, *HM* is not able to find any valid placements since the algorithm iterates over the N highest scoring placements, which are all unstable in these scenes. This issue could be alleviated by performing stability checks at the candidate pose generation stage, at the cost of a prohibitive increase in computation time due to *HM*'s iteration over all discretized positions and orientations of the object.

5.8.2 Observations In Real Robot Experiments

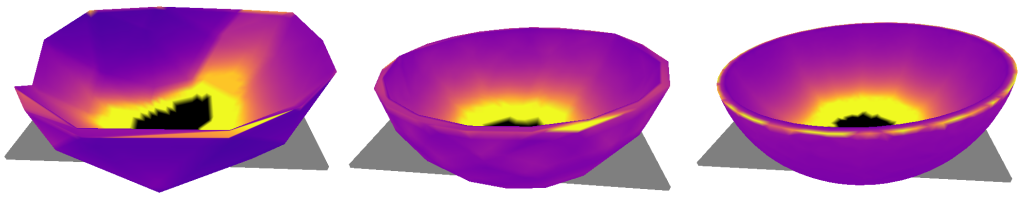
Four objects in the set (B, C, D, and I) are relatively long but thin, making them quite weak when placed upright and quite robust when placed lying down. Two of these objects (D and I) were frequently selected by the planner in the experiments (i.e., in E2, E3, E4, E5, E8, E10), and a lying down pose was always selected for them, which is expected from a SR point of view. However, the objects being thin, the gripper could not grasp them when they were lying down, ultimately resulting in a grasp planning failure. Similarly, although the pose selected for object H in E3 and object J in E9 are collision-free and make sense from a SR point of view, placing them would have caused the gripper to collide with other objects in the scene. This issue suggests that the planning of the grasping and release phases should be better integrated with placement planning. A rejection sampling step as done in (Chen et al., 2021; Wang and Hauser, 2021) could be performed to ensure that the object can be grasped and placed in the desired pose. Also, scene and object points that generated a grasp planning failure could have their likelihood of being selected again reduced in subsequent iterations with a slight modification to (5.10). In two occasions, the motion planner could not find a path to the placement pose in the allocated time, highlighting that planning for placements that can be easily executed is an avenue for future work. While a complete system might include a reachability component to validate that candidate placements can be carried out by the robot, this work focuses on the placement planning problem, free of constraints imposed by a specific robot platform.

A supplemental video, from which Figure 5.1 is taken, is provided[2] to illustrate the real robot experiments. The accompanying video highlights (i) a case where the robot successfully places objects in a scene and (ii) another where the robot topples the assembly when placing the last object. In the first case, when the robot placed the last object whose static robustness is deemed to be very low, a slight pose error almost caused the neighbouring object to topple, effectively validating the computed static robustness. In the second case, the robot created a contact with a neighbouring object when placing the last object due to an erroneous motion trajectory. This contact exerted force in a direction in which an infinite amount of force could be sustained according to the computed static robustness. As

[2] At the following URL: http://tiny.cc/stableplacement

	Compute Time (ms)	Proportion
QR-based Reaction Forces	9.8	6.8%
QP-based Reaction Forces	82.3	57.3%
Static Robustness	15.0	10.4%
Pose Definition	0.6	0.4%
Contact Resolution	35.9	25.0%
Total	144	100%

Table 5.3: Average compute time per iteration for each step.



(a) 50 vertices (b) 150 vertices (c) 1500 vertices

Figure 5.11: The bowl, with various vertex number and overlaid SR map, used to study the computational complexity of the proposed algorithm with curved objects approximated by triangle meshes.

the robot applied force, the whole structure collapsed, outlining the importance of using static robustness not only to define the placement pose, but also the approach direction of the robot when placing an object.

5.8.3 Computational Complexity

As expected, in terms of compute time, the contact resolution and constrained optimization steps are the most demanding, taking together more than 80% of the total compute time according to Table 5.3. The QR-based reaction force solver being significantly faster than the QP-based solver confirms that the latter should be used only to verify placements that are likely to be successful. Per iteration, the overhead of computing the static robustness heuristic accounts for about 10% of the compute time, which is largely outweighed by the time saved with the reduction in the number of iterations required to find a solution.

By design, the computation of the SR map grows linearly with the number of objects in the scene. This is due to each object being considered in isolation in the computation of the SR map. Hence, in a scene comprising a large number of objects, the proposed heuristic should enable our placement planner to focus on subsets of the scene that are more likely to offer a strong support.

Objects with curved shapes can be challenging for the proposed algorithm, which relies on planar interfaces. The shape of a curved object, like the bowl shown in Figure 5.11, can be approximated by a triangle mesh, with a lower number of vertices producing a coarser

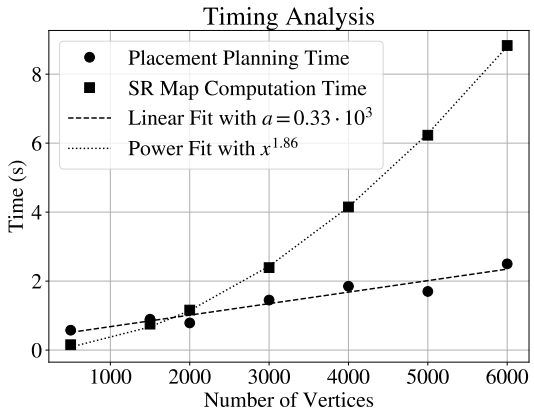


Figure 5.12: The time required to compute the SR map grows less than quadratically with the number of vertices in the object mesh, and the time required to find a stable placement grows linearly with the number of vertices.

approximation that might result in erroneous placement planning. While a higher number of vertices usually reduces the modeling error, it also increases the computational cost of the algorithm and ultimately slows down the planning process.

Placement planning experiments on a hemispherical shell with flattened end (i.e., a bowl), were performed to study the computational complexity of the proposed algorithm. Triangle meshes with various numbers of vertices were obtained through quadric decimation (Garland and Heckbert, 1997) and used by the proposed algorithm to find stable placements for a cube. Ten experiments were performed for each mesh, and the average time required to compute the SR map and find a stable placement was recorded. Results from these experiments are shown in Figure 5.12, where the time required to compute the SR map grows approximately following

$$t_{\text{SR}} = 8.53 \cdot 10^{-7} \cdot v^{1.856} \ , \tag{5.64}$$

and the time required to find a stable placement grows approximately following

$$t_{\text{plan}} = 0.346 + 0.333 \cdot 10^{3} v \ , \tag{5.65}$$

where v is the number of vertices in the object mesh. Hence, the time required to compute the SR map grows less than quadratically with the number of vertices in the object mesh, and the time required to find a stable placement grows linearly with the number of vertices. However, as can be noticed in Figure 5.11, there is a diminishing return to increasing the resolution of the SR map beyond a certain point with only a subtle difference between Figure 5.11b and Figure 5.11c, with the latter computed at a 10× greater resolution. Consequently, an adequate mesh resolution should be selected to balance the trade-off between modeling error and computational complexity. Although the execution time of the proposed algorithm does

not grow prohibitively with the number of vertices, the processing time can be limited by using a coarser resolution for SR computations and an interpolation scheme to produce a finer map.

5.9 Conclusion

This chapter defines an inertia-aware object placement planner that can scale to scenes with many objects with no assumption made on object convexity, homogeneous density, or shape. Our algorithm uses object inertial parameters at every step of the process to increase the odds of sampling a stable placement pose and to preserve the assembly's robustness to external perturbations. We show that the use of our *static robustness* heuristic greatly increases the odds of sampling a stable placement pose, with a light computational overhead, making the proposed algorithm much faster than other methods. Future work will focus on improving the efficiency of the algorithm by performing a local optimization after having defined a placement pose such that an almost stable pose can be adjusted to be stable. Also, a learning-based approach to static robustness inference will be investigated to improve compute time while keeping the generalizability of the method.

Chapter 6

Conclusion

> If machines produce everything we need, the outcome will depend on how things are distributed.
>
> Stephen Hawking.

Recognizing that the operations of mobile collaborative robots (cobots) are currently limited to well-defined tasks in structured environments, this thesis contributes algorithms to improve cobots autonomy in object handling tasks. In particular, algorithms presented in this thesis enable cobots to better model the impact of their actions on their environment and on the objects they manipulate. This improved situational awareness can be leveraged by proposed algorithms to safely transport sets of objects and stably place them amongst other objects, two key components of effective object handling.

6.1 Summary of Contributions

In Chapter 3, we introduced two novel methods for the inertial parameter identification of manipulated objects with cobots, which are constrained to move within safe velocity limits. The first method reduces the influence of noisy signals on the identification process by considering signal-to-noise ratios in the measurement data and transitioning between approximate and complete dynamics models depending on a measure of motion dynamism. The second method combines visual and force-torque measurements such that safe "stop-and-go" motions are sufficient, under mild assumptions, to identify the complete set of inertial parameters. In Chapter 4, we introduced an algorithm that can determine the capacity of multi-object assemblies to withstand external forces without becoming unstable. We demonstrate the practical use of our algorithm in safely transporting a set of objects in frictional contact, without resorting to computationally expensive dynamics simulations. Our novel *robustness* evaluation algorithm is also the basis of the placement planner proposed in Chapter 5, in which we presented a method to efficiently plan stable placements of objects

in multi-object assemblies. Our planning algorithm offers substantial speed improvements over existing methods by searching for valid placement pose in a much smaller space, and by leveraging our assembly robustness model to increase the odds of sampling stable placements.

In summary, this thesis contributes algorithms enhancing the versatility and adaptability of cobots in object handling tasks, setting the stage for a broader utilization of these machines across operations and environments. Specifically, the main contributions of this thesis are:

1. an analysis of the dynamics of safe cobot operation and of human manipulation during activities of daily living;

2. a method to quickly identify inertial parameters of manipulated objects that is specifically targeted at the typical cobot operating regime;

3. an inertial parameter identification formulation incorporating the homogeneous part density assumption, resulting in a simplified problem;

4. a part-segmentation method combining bottom-up point-cloud segmentation with hierarchical tetrahedra clustering;

5. a general, physically-grounded approach to assess the robustness of multi-object assemblies to external forces;

6. a method making use of the robustness evaluation to determine the maximal acceleration that a mobile robot can safely generate when transporting a set of objects; and

7. an algorithm that efficiently generates stable placement poses for assemblies of rigid objects in frictional contact.

All contributions are supported by extensive simulation studies and real robot experiments that demonstrate the effectiveness of the proposed methods. In addition to the conceptual contributions listed above, this thesis also contributes various tools, mostly in the form of publicly-available software, that can be used by the community as a basis for further research.

6.2 Future Research Directions

The research contributions presented in this thesis collectively support the more general object handling task. To this end, we have introduced algorithms that improve the capacity of cobots to efficiently build object models and use them to complete manipulation tasks. However, there exist several promising avenues of future work. The methods introduced in Chapter 3 for the inertial parameters identification of manipulated objects require a motion trajectory to be performed while data is collected. To achieve greater efficiency, it would

be desirable to blend identification trajectories with the robot motions performed as part of normal operations. With such a scheme, cobots could continuously and autonomously refine object models while performing value-adding tasks.

Our assembly robustness assessment algorithm, introduced in Chapter 4, produces an accurate evaluation of the robustness of multi-object assemblies to external forces, but can be too computationally expensive for large assemblies. In contrast, Chapter 5 introduced an approximation algorithm running in polynomial time, where objects making up the assembly are treated in isolation. A promising future research direction would consider a middle ground between these two approaches, where a subset of the assembly would be considered when evaluating the robustness of a given object. An anytime algorithm could progressively increase the size of the assembly subset considered, as more time is spent on the computation, effectively refining the robustness evaluation.

The stable placement planner introduced in Chapter 5 samples points in the scene to generate placement candidates based on the robustness assessment of the assembly. Although our approach greatly improves the efficiency of the planning process, further speed gains could achieved by considering previous placement candidates when generating new ones. A probabilistic approach could be used to avoid selecting placement candidates in regions of the scene where previous ones have failed the validation step.

The use of neural networks for generating stable placements is another promising research direction. Since candidate placements have to pass a stability validation step, the capacity of generative models to propose a diversity of solutions for a given input would make them well suited for this task. Finally, very large neural networks have shown remarkable capabilities for abstract long-horizon task planning involving multiple intermediate operations. In contrast, our stable object placement planner is focused on physically-grounded short-horizon planning, where a single step is considered but the dynamics of the objects are taken into account. A promising avenue of future research would combine the strengths of both approaches to perform long-horizon tasks involving intricate dynamics.

Image Credits

Introduction

In Figure 1.1, the images are credited as follows:

- The robot unloading the pallet of mixed items is from RIOS Intelligent Machines.
- The worker at the workstation is a rendering from Geolean USA.
- The shelf full of various objects is from Qingdao ABC Tools MFG Corp.
- The robot docked at a charging station is from Fetch Robotics.

Preliminaries

Robot Systems

In Figure 2.1, the images are credited as follows:

- The automatic temple door-opening mechanism is from Bennet Woodcroft's 1851 translation of Heron's "Pneumatics" book.
- The artwork is from Engelberger's US patent 3,661,051.

In Figure 2.2, the images are credited as follows:

- The photograph from R.U.R is in the public domain.
- The cover of the first edition of "I, Robot" is done by Edd Cartier for Gnome Press.
- The picture of Shakey is from SRI International.
- The photograph of the Unimate is under a Creative Commons license and provided by The Henry Ford Museum.
- Image adapted from Arnold Reinhold's photograph under CC-BY-SA 4.0.
- The rendering of the UR5 robot is from the data sheet provided by Universal Robots.

In Figure 2.3, the images are credited as follows:

- The 6-axis robot arm is a Kinova Link6 and the image is from the manufacturer.
- The SCARA robot is a Mecademic Meca500 and the image is from the manufacturer.
- the Delta robot is a ABB FlexPicker and the image is from the manufacturer.

Bibliography

Agboh, W. C., Sharma, S., Srinivas, K., Parulekar, M., Datta, G., Qiu, T., Ichnowski, J., Solowjow, E., Dogar, M., and Goldberg, K. (2023). Learning to efficiently plan robust frictional multi-object grasps. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 10660–10667.

Alterovitz, R., Koenig, S., and Likhachev, M. (2016). Robot planning in the real world: Research challenges and opportunities. *AI Magazine*, 37(2):76–84.

Andersen, E. D. and Andersen, K. D. (2000). The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In *High performance optimization*, pages 197–232.

Atkeson, C. G., An, C. H., and Hollerbach, J. M. (1986). Estimation of inertial parameters of manipulator loads and links. *The Int. J. Robotics Research*, 5(3):101–119.

Attene, M., Mortara, M., Spagnuolo, M., and Falcidieno, B. (2008). Hierarchical convex approximation of 3D shapes for fast region selection. In *Computer Graphics Forum*, volume 27, pages 1323–1332.

Ayusawa, K. and Nakamura, Y. (2010). Identification of standard inertial parameters for large-DoF robots considering physical consistency. In *Proc. 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 6194–6201. IEEE.

Baraff, D. (1991). Coping with friction for non-penetrating rigid body simulation. *ACM SIGGRAPH computer graphics*, 25(4):31–41.

Baraff, D. (1993). Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10(2):292–352.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483.

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., and Taubin, G. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359.

Bhushan, B. (1998). Contact mechanics of rough surfaces in tribology: multiple asperity contact. *Tribology letters*, 4:1–35.

Bhushan, B. (2013). *Introduction to Tribology, Second Edition*. John Wiley & Sons.

Bicchi, A. (1995). On the Closure Properties of Robotic Grasping. *The Int. Journal of Robot. Research*, 14(4):319–334.

Bicchi, A. and Kumar, V. (2000). Robotic grasping and contact: A review. In *IEEE Int. Conf. on Robot. and Autom.*, volume 1, pages 348–353.

Billard, A. and Kragic, D. (2019). Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414.

Boneschanscher, N., van der Drift, H., Buckley, S. J., and Taylor, R. H. (1988). Subassembly stability. In *AAAI Conf. on Artificial Intelligence*, volume 88, pages 780–785.

Borst, C., Fischer, M., and Hirzinger, G. (1999). A fast and robust grasp planner for arbitrary 3D objects. In *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, volume 3, pages 1890–1896.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2015). Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set. *IEEE Robotics & Automation Magazine*, 22(3):36–52.

Chen, H., Wan, W., Koyama, K., and Harada, K. (2021). Planning to build block structures with unstable intermediate states using two manipulators. *IEEE Trans. on Autom. Science and Engineering*, 19(4):3777–3793.

Chen, X., Golovinskiy, A., and Funkhouser, T. (2009). A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics*, 28(3):1–12.

Christensen, H. I. (2024). A roadmap for us robotics: Robotics for a better tomorrow. pages 1–70.

Coulomb, C. (1821). *Théorie des machines simples: en ayant égard au frottement de leurs parties et à la roideur des cordages*. Bachelier.

Coumans, E. and Bai, Y. (2016–2021). PyBullet, a Python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`.

Dogar, M. R., Hsiao, K., Ciocarlie, M. T., and Srinivasa, S. S. (2012). Physics-based grasp planning through clutter. In *Robotics: Science and systems*, volume 8, pages 57–64.

Donald, B., Xavier, P., Canny, J., and Reif, J. (1993). Kinodynamic motion planning. *Journal of the ACM*, 40(5):1048–1066.

Dou, Z., Lin, C., Xu, R., Yang, L., Xin, S., Komura, T., and Wang, W. (2022). Coverage axis: Inner point selection for 3d shape skeletonization. 41(2):419–432.

Ericson, C. (2004). *Real-Time Collision Detection*. CRC Press.

Farsoni, S., Landi, C. T., Ferraguti, F., Secchi, C., and Bonfe, M. (2017). Compensation of load dynamics for admittance controlled interactive industrial robots using a quaternion-based kalman filter. *IEEE Robotics and Automation Letters*, 2(2):672–679.

Farsoni, S., Landi, C. T., Ferraguti, F., Secchi, C., and Bonfè, M. (2018). Real-time identification of robot payload using a multirate quaternion-based Kalman filter and recursive total least-squares. In *Proc. 2018 IEEE Int. Conf. on Robotics and Automation*, pages 2103–2109.

Ferrari, C., Canny, J. F., and others (1992). Planning optimal grasps. In *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, volume 3, pages 1–6.

Flores, F. G. and Kecskeméthy, A. (2013). Time-optimal path planning for the general waiter motion problem. In *Advances in Mechanisms, Robot. and Design Education and Research*, pages 189–203. Springer.

Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Conf. on Computer graphics and interactive techniques*, pages 209–216.

Gautier, M. and Khalil, W. (1992). Exciting trajectories for the identification of base inertial parameters of robots. *The International Journal of Robotics Research*, 11(4):362–375.

Goldberg, K., Mirtich, B. V., Zhuang, Y., Craig, J., Carlisle, B. R., and Canny, J. (1999). Part pose statistics: Estimators and experiments. *IEEE Trans. on Robot. and Autom.*, 15(5):849–857.

Goldstein, H., Poole, C., and Safko, J. (2002). *Classical Mechanics, 3rd ed.* Addison Wesley.

Golluccio, G., Gillini, G., Marino, A., and Antonelli, G. (2020). Robot dynamics identification: A reproducible comparison with experiments on the kinova Jaco2. *IEEE Robotics & Automation Magazine*.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. JHU press.

Hamrick, J. B., Battaglia, P. W., Griffiths, T. L., and Tenenbaum, J. B. (2016). Inferring mass in complex scenes by mental simulation. *Cognition*, 157:61–76.

Haustein, J. A., Hang, K., Stork, J., and Kragic, D. (2019). Object placement planning and optimization for robot manipulators. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 7417–7424.

Heins, A. and Schoellig, A. P. (2023). Keep it upright: Model predictive control for non-prehensile object transportation with obstacle avoidance on a mobile manipulator. *IEEE Robot. and Autom. Letters*.

Henderson, H. V. and Searle, S. (1979). Vec and vech operators for matrices, with some uses in Jacobians and multivariate statistics. *Canadian Journal of Statistics*, 7(1):65–81.

Hillier, F. S. and Lieberman, G. J. (2015). *Introduction to operations research*. McGraw-Hill.

Hoffman, D. D. and Richards, W. A. (1984). Parts of recognition. *Cognition*, 18(1-3):65–96.

Huang, Y. and Sun, Y. (2019). A dataset of daily interactive manipulation. *The Int. J. Robotics Research*, 38(8):879–886.

International Federation of Robotics (2024). World robotics report 2024.

ISO (2011). Robots and robotic devices - Safety requirements for industrial robots - Part 1: Robots. Technical Report ISO 10218-1:2011, ISO, Geneva (2011).

ISO (2016). Robots and robotic devices - Collaborative robots. Technical Report ISO/TS 15066:2016, ISO, Geneva (2016).

Ji, X. and Xiao, J. (2001). Planning motions compliant to complex contact states. *The Int. Journal of Robot. Research*, 20(6):446–465.

Jiang, Y., Lim, M., Zheng, C., and Saxena, A. (2012). Learning to place new objects in a scene. *The Int. Journal of Robot. Research*, 31(9):1021–1043.

Johns, R. L., Wermelinger, M., Mascaro, R., Jud, D., Hurkxkens, I., Vasey, L., Chli, M., Gramazio, F., Kohler, M., and Hutter, M. (2023). A framework for robotic excavation and dry stone construction using on-site materials. *Science Robotics*, 8(84):eabp9758.

Kaick, O. V., Fish, N., Kleiman, Y., Asafi, S., and Cohen-Or, D. (2014). Shape segmentation by approximate convexity analysis. *ACM Transactions on Graphics*, 34(1):1–11.

Kao, G. T.-C., Iannuzzo, A., Thomaszewski, B., Coros, S., Van Mele, T., and Block, P. (2022). Coupled rigid-block analysis: stability-aware design of complex discrete-element assemblies. *Computer-Aided Design*, 146:103216.

Kartmann, R., Paus, F., Grotz, M., and Asfour, T. (2018). Extraction of physically plausible support relations to predict and validate manipulation action effects. *IEEE Robot. and Autom. Letters*, 3(4):3991–3998.

Kaufman, D. M. (2009). *Coupled principles for computational frictional contact mechanics*. PhD Thesis, Rutgers University Graduate School.

Kavraki, L., Latombe, J.-C., and Wilson, R. H. (1993). On the complexity of assembly partitioning. *Information Processing Letters*, 48(5):229–235.

Kroger, T., Kubus, D., and Wahl, F. M. (2008). 12D force and acceleration sensing: A helpful experience report on sensor characteristics. In *Proc. 2008 IEEE Int. Conf. on Robotics and Automation*, pages 3455–3462.

Kubus, D., Kroger, T., and Wahl, F. M. (2008). On-line estimation of inertial parameters using a recursive total least-squares approach. In *Proc. 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3845–3852.

Kuffner, J. J. and LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, volume 2, pages 995–1001.

Kunz, T. and Stilman, M. (2014). Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3713–3719.

Lang, S. (1999). *Fundamentals of Differential Geometry*, volume 1. Springer Science and Media.

Lau, B., Sprunk, C., and Burgard, W. (2009). Kinodynamic motion planning for mobile robots using splines. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2427–2433. IEEE.

LaValle, S. M. and Kuffner Jr, J. J. (2001). Randomized kinodynamic planning. *The Int. Journal of Robot. Research*, 20(5):378–400.

Lee, S. (1994). Subassembly identification and evaluation for assembly planning. *IEEE Trans. on Systems, Man, and Cybernetics*, 24(3):493–503.

Lee, S. and Moradi, H. (1999). Disassembly sequencing and assembly sequence verification using force flow networks. In *IEEE Int. Conf. on Robot. and Autom.*, volume 4, pages 2762–2767.

Lee, S. and Yi, C. (1994). Force-based reasoning in assembly planning. In *Int. Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 12877, pages 165–172.

Lee, T., Wensing, P. M., and Park, F. C. (2019). Geometric robot dynamic identification: A convex programming approach. *IEEE Trans. Robotics*, 36(2):348–365.

Lee, Y., Thomason, W., Kingston, Z., and Kavraki, L. E. (2023). Object reconfiguration with simulation-derived feasible actions. In *IEEE Int. Conf. on Robot. and Autom.*, pages 8104–8111.

Levinshtein, A., Stere, A., Kutulakos, K. N., Fleet, D. J., Dickinson, S. J., and Siddiqi, K. (2009). Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297.

Li, P., Wang, B., Sun, F., Guo, X., Zhang, C., and Wang, W. (2015). Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics*, 35(1):1–16.

Lin, C., Li, C., Liu, Y., Chen, N., Choi, Y.-K., and Wang, W. (2021). Point2Skeleton: Learning Skeletal Representations from Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4277–4286.

Lin, C., Liu, L., Li, C., Kobbelt, L., Wang, B., Xin, S., and Wang, W. (2020). Seg-mat: 3d shape segmentation using medial axis transform. *IEEE Transactions on Visualization and Computer Graphics*.

Lin, Y., Wang, C., Zhai, D., Li, W., and Li, J. (2018). Toward better boundary preserved supervoxel segmentation for 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 143:39–47.

Liu, C. K. and Negrut, D. (2021). The role of physics-based simulators in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):35–58.

Lukos, J., Ansuini, C., and Santello, M. (2007). Choice of contact points during multidigit grasping: Effect of predictability of object center of mass location. *J. Neuroscience*, 27(14):3894–3903.

Lynch, K. M. and Park, F. C. (2017). *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press.

Maeda, Y., Goto, Y., and Makita, S. (2009). A new formulation for indeterminate contact forces in rigid-body statics. In *2009 IEEE Int. Symposium on Assembly and Manufacturing*, pages 298–303.

Maeda, Y., Oda, K., and Makita, S. (2007). Analysis of indeterminate contact forces in robotic grasping and contact tasks. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1570–1575.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *2001 IEEE International Conference on Computer Vision*, volume 2, pages 416–423.

Mason, M. T. (1986). Mechanics and planning of manipulator pushing operations. *The Int. Journal of Robot. Research*, 5(3):53–71.

Mason, M. T. (2001). *Mechanics of robotic manipulation*. MIT press.

Mason, M. T. (2018). Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:1–28.

Mason, M. T. and Lynch, K. M. (1993). Dynamic manipulation. In *Proc. 1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 152–159.

Mattikalli, R., Baraff, D., and Khosla, P. (1996). Finding all stable orientations of assemblies with friction. *IEEE Trans. on Robot. and Autom.*, 12(2):290–301.

Mattikalli, R., Khosla, P. K., Repetto, B., and Baraff, D. (1993). Stability of assemblies. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 1, pages 652–661.

Mavrakis, N. and Stolkin, R. (2020). Estimation and exploitation of objects' inertial parameters in robotic grasping and manipulation: A survey. *Robotics and Autonomous Systems*, 124:103374.

Mavrakis, N., Stolkin, R., Baronti, L., Kopicki, M., Castellani, M., and others (2016). Analysis of the inertia and dynamics of grasped objects, for choosing optimal grasps to enable torque-efficient post-grasp manipulations. In *Proc. 2016 IEEE/RAS 16th Int. Conf. on Humanoid Robots*, pages 171–178.

Mojtahedzadeh, R., Bouguerra, A., Schaffernicht, E., and Lilienthal, A. J. (2015). Support relation analysis and decision making for safe robotic manipulation tasks. *Robot. and Autonomous Systems*, 71:99–117.

Motoda, T., Petit, D., Nishi, T., Nagata, K., Wan, W., and Harada, K. (2022). Shelf replenishment based on object arrangement detection and collapse prediction for bimanual manipulation. *Robotics*, 11(5):104.

Nadeau, P. (2024). A Standard Rigid Transformation Notation Convention for Robotics Research. *arXiv preprint arXiv:2405.07351*.

Nadeau, P., Giamou, M., and Kelly, J. (2022). Fast Object Inertial Parameter Identification for Collaborative Robots. In *IEEE International Conference on Robotics and Automation*.

Nadeau, P. and Kelly, J. (2025). Stable Object Placement Planning From Contact Point Robustness. *IEEE Transactions on Robotics*, 41:3669–3683.

Newton, I. (1962). *Sir Isaac Newton's Mathematical Principles of Natural Philosophy and His System of the World (A. Motte, F. Cajori, Trans.)*. University of California Press. (Original work published 1846).

Omata, T. and Nagata, K. (2000). Rigid body analysis of the indeterminate grasp force in power grasps. *IEEE Trans. on Robot. and Autom.*, 16(1):46–54.

Otsuki, M. and Matsukawa, H. (2013). Systematic breakdown of Amontons' law of friction for an elastic object locally obeying Amontons' law. *Scientific reports*, 3(1):1586.

Pagano, C. C. and Turvey, M. (1992). Eigenvectors of the inertia tensor and perceiving the orientation of a hand-held object by dynamic touch. *Perception & Psychophysics*, 52(6):617–624.

Pang, J.-S. and Trinkle, J. C. (2000). Stability characterizations of fixtured rigid bodies with coulomb friction. In *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, volume 1, pages 361–368.

Rodrigues, R. S., Morgado, J. F., and Gomes, A. J. (2018). Part-based mesh segmentation: a survey. In *Computer Graphics Forum*, volume 37, pages 235–274.

Saudabayev, A., Rysbek, Z., Khassenova, R., and Varol, H. A. (2018). Human grasping database for activities of daily living with depth, color and kinematic data streams. *Scientific data*, 5(1):1–13.

Saxena, D. M. and Likhachev, M. (2023). Planning for Complex Non-prehensile Manipulation Among Movable Objects by Interleaving Multi-Agent Pathfinding and Physics-Based Simulation. In *IEEE Int. Conf. on Robot. and Autom.*, pages 8141–8147.

Shamir, A. (2008). A survey on mesh segmentation techniques. In *Computer Graphics Forum*, volume 27, pages 1539–1556.

Shigley, J. E. (1972). *Mechanical engineering design*. McGraw-Hill Companies.

Shin, H. V., Porst, C. F., Vouga, E., Ochsendorf, J., and Durand, F. (2016). Reconciling elastic and equilibrium methods for static analysis. *ACM Trans. on Graphics (TOG)*, 35(2):1–16.

Si, H. (2015). TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software*, 41(2):1–36.

Siciliano, B., Khatib, O., and Kröger, T. (2008). *Springer handbook of robotics*, volume 200. Springer.

Sinha, P. R. and Abel, J. M. (1990). A contact stress model for multifingered grasps of rough objects. In *IEEE Int. Conf. on Robotics and Autom.*, pages 1040–1045.

Song, C. and Boularias, A. (2020). A probabilistic model for planar sliding of objects with unknown material properties. In *Proc. 2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

Sousa, C. D. and Cortesao, R. (2014). Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach. *The International Journal of Robotics Research*, 33(6):931–944.

Srinivas, K., Ganti, S., Parikh, R., Ahmad, A., Agboh, W., Dogar, M., and Goldberg, K. (2023). The Busboy Problem: Efficient Tableware Decluttering Using Consolidation and Multi-Object Grasps. In *IEEE Int. Conf. on Autom. Science and Engineering (CASE)*, pages 1–6.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2020). OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672.

Stewart, D. and Trinkle, J. C. (2000). An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *IEEE Int. Conf. on Robotics and Autom.*, volume 1, pages 162–169.

Stewart, D. E. (2000). Rigid-body dynamics with friction and impact. *SIAM review*, 42(1):3–39.

Sugar, T. G. and Kumar, V. (2000). Metrics for analysis and optimization of grasps and fixtures. In *IEEE Int. Conf. on Robot. and Autom.*, volume 4, pages 3561–3566. IEEE.

Sun, L., Yoneda, T., Wheeler, S. W., Jiang, T., and Walter, M. R. (2024). StackGen: Generating Stable Structures from Silhouettes via Diffusion.

Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N., and Telea, A. (2016). 3d skeletons: A state-of-the-art report. In *Computer Graphics Forum*, volume 35, pages 573–597.

Toussaint, M. (2015). Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Int. Joint Conf. on Artificial Intelligence*.

Traversaro, S., Brossette, S., Escande, A., and Nori, F. (2016). Identification of fully physical consistent inertial parameters using optimization on manifolds. In *Proc. 2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 5446–5451.

Wang, F. and Hauser, K. (2019). Stable bin packing of non-convex 3D objects with a robot manipulator. In *2019 Int. Conf. on Robot. and Autom.*, pages 8698–8704.

Wang, F. and Hauser, K. (2021). Dense robotic packing of irregular and novel 3D objects. *IEEE Trans. on Robot.*, 38(2):1160–1173.

Wang, L., Guo, S., Chen, S., Zhu, W., and Lim, A. (2010). Two natural heuristics for 3D packing with practical loading constraints. In *Pacific Rim Int. Conf. on Artificial Intelligence*, pages 256–267. Springer.

Wang, Z., Song, P., and Pauly, M. (2021). State of the art on computational design of assemblies with rigid parts. *Computer Graphics Forum*, 40(2):633–657.

Webb, D. J. and Van Den Berg, J. (2013). Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics . In *IEEE Int. Conf. on Robot. and Autom.*, pages 5054–5061.

Wensing, P. M., Kim, S., and Slotine, J.-J. E. (2017). Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution. *IEEE Robotics and Automation Letters*, 3(1):60–67.

Wermelinger, M., Johns, R., Gramazio, F., Kohler, M., and Hutter, M. (2021). Grasping and object reorientation for autonomous construction of stone structures. *IEEE Robot. and Autom. Letters*, 6(3):5105–5112.

Whiting, E., Shin, H., Wang, R., Ochsendorf, J., and Durand, F. (2012). Structural optimization of 3D masonry buildings. *ACM Trans. on Graphics (TOG)*, 31(6):1–11.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57.

Xiao, J. (1993). Automatic determination of topological contacts in the presence of sensing uncertainties. In *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, pages 65–70.

Xu, D., Martín-Martín, R., Huang, D.-A., Zhu, Y., Savarese, S., and Fei-Fei, L. F. (2019). Regression Planning Networks. *Advances in Neural Information Processing Systems*, 32:1319–1329.

Yao, J., Kaufman, D. M., Gingold, Y., and Agrawala, M. (2017). Interactive design and stability analysis of decorative joinery for furniture. *ACM Trans. on Graphics*, 36(2):1–16.

Yoneda, T., Jiang, T., Shakhnarovich, G., and Walter, M. R. (2023). 6-DoF stability field via diffusion models. *arXiv preprint arXiv:2310.17649*.

Yu, K.-T., Bauza, M., Fazeli, N., and Rodriguez, A. (2016). More than a million ways to be pushed. A high-fidelity experimental dataset of planar pushing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 30–37. IEEE.

Zeng, A., Song, S., Lee, J., Rodriguez, A., and Funkhouser, T. (2019). Tossingbot: Learning to throw arbitrary objects with residual physics. *Proc. Robotics: Science and Systems (RSS)*.

Zhu, Y., Tremblay, J., Birchfield, S., and Zhu, Y. (2021). Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, pages 6541–6548.