Decentralized Collision Avoidance and Motion Planning for Multi-Robot Manipulator Systems

by

Divyam Mehta

(Under the direction of Prashant Doshi)

Abstract

We present Fast-ORCA (FORCA), a decentralized collision-avoidance and motion-planning framework for multi-robot manipulators performing precision sorting tasks. FORCA adapts Optimal Reciprocal Collision Avoidance and translates velocity control to position control. At each cycle it selects a collision-free end-effector displacement using 3D velocity-obstacle geometry and accepts it only if it yields a valid inverse-kinematics solution. The pose is then connected by an RRTConnect path retimed by a waypoint speed-control module so that the executed motion matches the FORCA velocity command. We evaluate four UR3e arms in Gazebo Harmonic under Far and Near inspection layouts and onion densities from 30 to 50. Compared with a greedy Straight-to-Goal baseline, FORCA reduces inter-robot collisions from anywhere within 0–10 per run to zero while reducing total throughput by only 2–9% (e.g., 14.95 to 14.33 onions/min at 30-onion Near Inspection).

Index words: [Robotic Manipulators, Optimal Reciprocal Collision Avoidance, Decentralized Planning, Precision Sorting, Collaborative Robots]

Decentralized Collision Avoidance and Motion Planning for Multi-Robot Manipulator Systems

by

Divyam Mehta

B.Tech., Pandit Deendayal Energy University, India, 2021

A Thesis Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Master of Science

Athens, Georgia

2025

Decentralized Collision Avoidance and Motion Planning for Multi-Robot Manipulator Systems

by

Divyam Mehta

Major Professor: Prashant Doshi

Committee: Ramviyas Parasuraman

Frederick Maier

Electronic Version Approved:

Ron Walcott

Dean of the Graduate School

The University of Georgia

December 2025

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Multi-robot manipulator systems are increasingly in demand in modern industrial automation, especially where tasks are repetitive, time-critical or need to be performed with great precision such as in pick-and-place applications, sorting and assembly lines. Compared to a single robotic arm, having several manipulators combined in a cooperating manner is advantageous in many ways. Multi-robot configurations provide distinct advantages that make them especially well suited to these applications. Collaborative robotic systems are capable of executing tasks that either surpass the physical limitations of an individual manipulator or cannot be accomplished with comparable levels of efficiency.

The distribution of workload among multiple manipulators enables complex operations to be decomposed into smaller, manageable subtasks that may be executed concurrently. This parallelization approach not only maximizes efficiency but also facilitates scalability, where more manipulators can be incorporated within the system without radical modifications in the overall infrastructure. In addition, several robots improve the effective coverage of workspace without the need to develop and construct a larger physical manipulator, which often leads to higher structural complexity and more expense.

Another significant advantage of having several manipulators is the potential to enhance the system's robustness and fault tolerance. In single manipulator systems, hardware failure or software malfunction

normally leads to full operational shutdown. For multi-robot systems, though, there is continuity in operation since tasks are distributed among the remaining manipulators when one is no longer available. This inherent resilience is of great value in industries where uninterrupted production and minimal disruption are desired.

1.2 Motivation

Automation is increasingly migrating into environments that possess repetitive, time-sensitive and high-accuracy tasks. In these environments, several robots may have the same workspace, requiring robust and fault-tolerant coordination. Although much work has been accomplished in motion planning and navigation for mobile robots, multi-robot manipulator planning has used centralized techniques for task assignment and trajectory planning to a great extent. Centralization may facilitate coordination but provides a single point of failure and may have the difficulty of scaling nicely.

A decentralized approach is appealing because the system can remain functional even if one robot fails. For mobile robots, Optimal Reciprocal Collision Avoidance (van den Berg et al., 2011) is a leading decentralized method. However, ORCA-style reasoning has seen little adoption for articulated manipulators, where joint limits, kinematics, and complex workspace geometry add new challenges. This thesis investigates adapting decentralized planning to multi-robot manipulators operating in dynamic, shared environments typical of processing lines, sorting stations, and assembly cells where arms must safely and efficiently pick and place objects from one location to another.

1.3 Goal

The goal of this work is to develop a decentralized motion planning and collision-avoidance method for multi-robot manipulators, derived from ORCA. Our experiments handle onions on a conveyor with workcells containing four robot arms. The task is onion sorting. Specifically, we aim to: (a) correctly sort all onions by blemish severity; (b) prevent collisions among the manipulators; (c) generate efficient

trajectories for each manipulator; and (d) assign onion-picking tasks via a centralized allocator while execution and collision avoidance remain decentralized.

1.4 Challenges

Implementing ORCA in multi-manipulator settings has seveeral challenges: (a) **Joint-space coupling**: Position and velocity of the child joints are dependent on their parent joints. However, in ORCA, the state variables (instantaneous position and velocity) are independent. (b) **Geometry**: Manipulators comprises of multiple links and are complex in shape. ORCA needs the agents to be confined within a circle/sphere; and (c) **IK feasibility**: An ORCA-suggested end-effector velocity may have no valid inverse-kinematics solution (or may violate joint/velocity/acceleration limits), unlike a holonomic mobile robot.

1.5 Contribution

This section introduces a decentralized motion-planning and collision-avoidance framework for multi-robot manipulator systems operating in a tightly shared workspace, with a specific focus on precision sorting tasks using multiple robotic arms. The contributions below address the previously mentioned challenges with the manipulators in a unified way and are validated in simulation on a four-manipulator UR3e layout in Gazebo Harmonic using real task flows and onion-sorting configurations.

1.5.1 Fast-ORCA (FORCA): Adapting ORCA to Manipulators

The first and central contribution is the design of *Fast-ORCA (FORCA)*, a decentralized collision-avoidance method that retains the reciprocal, prediction-based nature of Optimal Reciprocal Collision Avoidance (ORCA) but reformulates it for articulated arms. Classical ORCA assumes agents can directly realize a 2D velocity. FORCA gives a position update instead of velocity update at each timestep, and hence replaces the velocity controller with a joint trajectory position-based controller. The collision-free velocity suggested by the ORCA half-plane is interpreted as an expected displacement over the next control

horizon, and this displacement is then converted into a feasible end-effector pose update which is finally tracked by the arm's joint-space controller. This preserves the decentralized property of ORCA that each arm reasons only about local neighbors while making it compatible with manipulator kinematics.

1.5.2 IK-Aware Velocity Filtering

A second contribution is the explicit *IK-feasibility filtering* inside the avoidance step. In dense shared workspaces, many collision-free displacements in Cartesian space do not admit a valid IK solution because of joint limits, camera-inspection poses or reachability constraints. FORCA handles this by generating a small set of candidate escape velocities (as in VO/ORCA), mapping each to a Cartesian displacement, and retaining only those displacements whose target end-effector pose yields a valid IK solution. In other words, the collision-avoidance stage is made robot-aware and not just geometry-aware. The planner never asks the robot to go somewhere it cannot actually reach. This closes a key gap between mobile-robot ORCA and manipulator applications.

1.5.3 Waypoint-Based Speed Control Consistent with FORCA

Since the ORCA step outputs an average admissible velocity for the next horizon, and because manipulator trajectories are executed as sequences of joint positions, this work introduces a *custom waypoint-based speed-control post-processor*. After a nominal joint-space path is produced via RRTConnect (with a high goal bias), this post-processor adjusts the timing between successive waypoints so that the executed end-effector motion respects the average velocity predicted by FORCA. Consequently, trajectory generation and collision avoidance become time-consistent, preventing temporal mismatches that could reintroduce collisions during execution.

1.5.4 Lightweight Neighbor Handling for Manipulators

A practical contribution is the *computationally efficient neighbor-selection strategy*. Instead of solving a linear program over many nearby manipulators simultaneously, FORCA evaluates one neighboring

agent at a time when computing the next admissible velocity for the primary arm. In the four-UR3e onion-sorting cell, this keeps the per-cycle computation low while maintaining collision-free operation.

1.5.5 End-to-End Validation in a Realistic Onion-Sorting Cell

To demonstrate that the above contributions hold in a realistic setting, the entire framework was implemented in Gazebo Harmonic with four UR3e manipulators performing onion-sorting tasks. Two inspection regimes were tested—*Far Inspection* (longer travel distance) and *Near Inspection* (shorter travel distance)—under multiple onion densities (30, 35, 40, 45, 50 for Far; 30, 40, 50 for Near), with three random spatial configurations per density. This comprehensive setup ensured that each planner was evaluated under diverse workspace and congestion conditions.

1.5.6 Quantitative Comparison with a Greedy STG Baseline

A further contribution is the quantitative comparison against a greedy *Straight-to-Goal (STG)* planner, which drives each robot directly to its goal without considering mutual avoidance. The comparison highlights the trade-off between throughput and safety:

- **Throughput:** In the Far Inspection setting, FORCA completed runs only **2–5% slower** than STG, despite taking detours to avoid other arms. In the Near Inspection setting, FORCA maintained throughput within **3–6%** of STG.
- **Path Safety:** While STG's collision counts increased with onion density (up to **8–10 collisions per run**), FORCA achieved **zero collisions across all tested configurations**.

These results confirm that FORCA achieves collision-free operation with minimal performance degradation compared to a non-avoiding baseline, proving its practicality for dense multi-arm systems.

1.6 Thesis Structure

The remainder of this thesis is organized as follows:

- **Chapter 2: Related Work.** This chapter reviews the existing literature on Velocity Obstacles (VO), Reciprocal Velocity Obstacles (RVO), and Optimal Reciprocal Collision Avoidance (ORCA).
- **Chapter 3: Fast-ORCA for Manipulators.** This chapter introduces the formal framework of the Fast-ORCA algorithm. It elaborates on the underlying principles including:
  - The mathematical formulation of the 3D collision cone.
  - The strategy employed to determine the optimal avoidance maneuver.
  - The extension and formal application of these principles to multi-manipulator robotic systems for precision sorting tasks.

  This chapter establishes the theoretical basis upon which the subsequent experimental analysis is constructed.
- **Chapter 4: Experiments and Results.** This chapter details the experimental setup, simulation environment and performance evaluation of the Fast-ORCA approach.
  - Section 4.2 introduces the performance baseline used for comparative analysis.
  - Section 4.3 presents the results and performance metrics such as overall throughput and path safety by comparing Fast-ORCA against the baseline method.

  The findings from this chapter quantitatively demonstrate the efficacy of our proposed approach.
- **Chapter 5: Conclusion.** The final chapter summarizes the contributions of this thesis, reflects on its limitations, and outlines potential directions for future extensions.

Chapter 2

Related Work

2.1 Velocity Obstacles (VO)

The Velocity Obstacle (VO) paradigm, introduced by (Chakravarthy & Ghose, 1998; Fiorini & Shiller, 1996, 1998) and consolidated by subsequent generalizations and surveys (Vesentini et al., 2024; Wilkie et al., 2009) has established a foundational geometric framework for decentralized collision avoidance in dynamic multi-agent environments. This approach models interactions between an autonomous agent and surrounding moving obstacles within a velocity space where feasible velocities correspond to collision-free motion. Each obstacle induces a velocity cone that defines the set of agent velocities leading to future collisions and allows the agent to select an admissible velocity from outside of these regions. This approach transformed the problem of motion planning under dynamic constraints into a velocity-space reasoning task and made it particularly suitable for holonomic robots with circular approximations. Over the years, VO-based methods have become central to real-time navigation systems, forming the basis for later refinements such as Reciprocal Velocity Obstacles (van den Berg et al., 2008), Optimal Reciprocal Collision Avoidance (van den Berg et al., 2011), Hybrid Reciprocal Velocity Obstacles (HRVO) (Snape et al., 2009, 2011), and other generalized formulations for large-scale crowd simulation and multi-robot navigation (Guy et al., 2009; Vesentini et al., 2024; Wilkie et al., 2009).

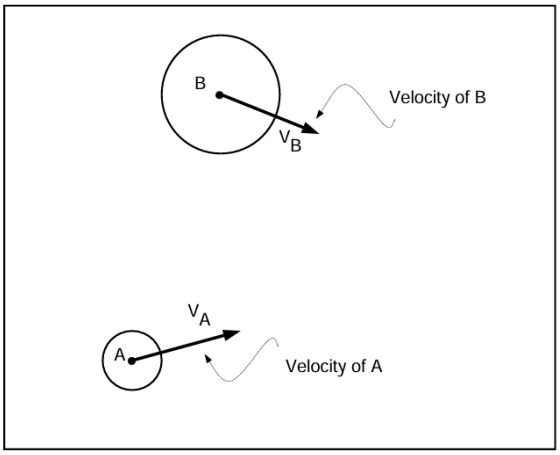

Figure 2.1: Agent A and a dynamic obstacle B in the plane (Fiorini & Shiller, 1998).

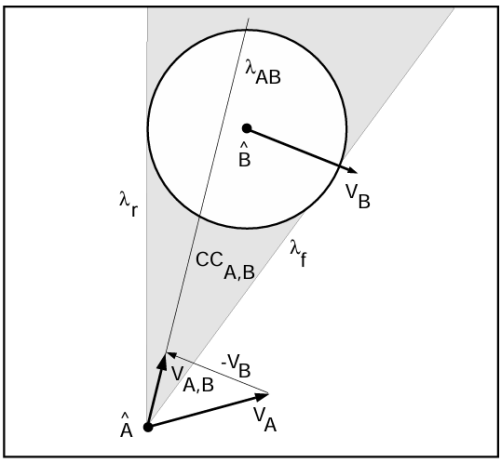

Figure 2.2: The relative velocity v_{AB} and the collision cone $CC_{A,B}$; see (2.1) (Fiorini & Shiller, 1998).

Consider a dynamic environment that consists of a holonomic disc–shaped agent A and moving disc–shaped obstacles B_i, $i = 1, \ldots, m$. The agent is described by its position $p_A \in \mathbb{R}^2$, radius $r_A > 0$, and velocity $\mathbf{v}_A \in \mathbb{R}^2$. Similarly, each obstacle B_i has $(p_{B_i}, r_{B_i}, \mathbf{v}_{B_i})$. The agent A is assumed to be capable of detecting the position, radius and velocity of every obstacle in its vicinity, i.e., $(p_{B_i}, r_{B_i}, \mathbf{v}_{B_i})$ for all $i = 1, \ldots, m$. The Velocity Obstacle (VO) framework represents motion on the tangent bundle $T\mathbb{R}^2 \simeq \mathbb{R}^2 \times \mathbb{R}^2$ where each state $(p(t), \mathbf{v}(t))$ describes the agent's configuration and its corresponding feasible velocities in the velocity space.

For each pair (A, B_i), define the combined radius $R = r_A + r_{B_i}$, and relative position $\mathbf{r}_{AB_i}$ and velocity $\mathbf{v}_{AB_i}$ as

$$\mathbf{r}_{AB_i} := p_{B_i} - p_A, \qquad \mathbf{v}_{AB_i} := \mathbf{v}_A - \mathbf{v}_{B_i}.$$

Let $\lambda_{A,B_i} = \{\, s\,\mathbf{r}_{AB_i} \mid s \geq 0 \,\}$ denote the ray from the origin along $\mathbf{r}_{AB_i}$. When $\|\mathbf{r}_{AB_i}\| > R$, the cone half–angle was given by (Chakravarthy & Ghose, 1998; Fiorini & Shiller, 1998) as

$$\theta = \arcsin\Big(\frac{R}{\|\mathbf{r}_{AB_i}\|}\Big) \in (0, \tfrac{\pi}{2}),$$

The transformation matrix for a 2D Planar rotation is given by

$$\mathrm{Rot}_{\varphi} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix},$$

Hence, the boundary or tangent rays are defined as

$$\lambda_f = \{\, s\,\mathrm{Rot}_{+\theta}\mathbf{r}_{AB_i} \mid s \geq 0 \,\}, \qquad \lambda_r = \{\, s\,\mathrm{Rot}_{-\theta}\mathbf{r}_{AB_i} \mid s \geq 0 \,\}.$$

Geometrically, the agent A is reduced to a point $\hat{A}$, while each obstacle B_i is expanded by the radius r_A of the agent to form an enlarged disc $\hat{B}_i$ of radius R as shown in 2.2. The *Relative Collision Cone* CC_{AB_i} is defined by the two tangents drawn from $\hat{A}$ to $\hat{B}_i$ and represents the set of forbidden relative velocities that corresponds to all velocities that would lead to a future collision between the agent and the obstacle (Fiorini & Shiller, 1996, 1998).

$$CC_{AB_i} = \Big\{ \mathbf{u} \in \mathbb{R}^2 \;\Big|\; \angle(\mathbf{u}, \lambda_{A,B_i}) \leq \theta,\ \ \mathbf{r}_{AB_i} \cdot \mathbf{u} \geq 0 \Big\}, \tag{2.1}$$

Translating the cone by $\mathbf{v}_{B_i}$ yields the absolute set of forbidden velocities for A also known as *Velocity Obstacle* $VO^A_{B_i}$ for A induced by B_i:

$$VO^A_{B_i} \;:=\; \mathbf{v}_{B_i} \,\oplus\, CC_{AB_i} \;=\; \big\{\, \mathbf{v}_A \in \mathbb{R}^2 \;\big|\; \mathbf{v}_{AB_i} \in CC_{AB_i} \big\}. \tag{2.2}$$

With multiple obstacles $B_1, \ldots, B_m$, the *Multiple Velocity Obstacle* for agent A is the union

$$VO^A \;=\; \bigcup_{j=1}^{m} VO^A_{B_j}. \tag{2.3}$$

Agent A must select a velocity vector $\mathbf{v}_A^{new}$ outside the VO^A (2.3) to avoid any collision with dynamic obstacles B_i.

Let $V \subset \mathbb{R}^2$ be the set of admissible velocities and $\mathbf{v}_A^{\text{pref}}$ be the preferred velocity (straight–line to goal, capped at $v_{\max}$). A basic VO update chooses the next velocity as

$$\mathbf{v}_A^{\text{new}} \;=\; \arg \min_{\mathbf{v} \in V \setminus VO^A} \left\| \mathbf{v} - \mathbf{v}_A^{\text{pref}} \right\|^2, \tag{2.4}$$

This needs to be recomputed at each sensing–actuation cycle.

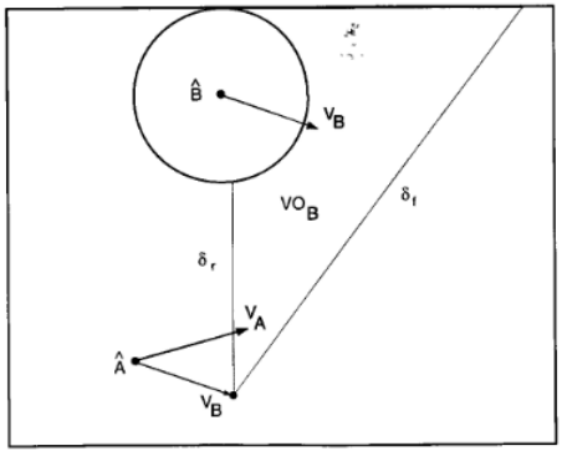

Figure 2.3: The velocity obstacle VO_B^A ; see (2.2) (Fiorini & Shiller, 1998).

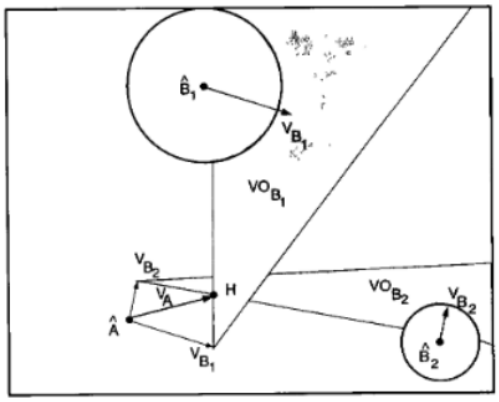

Figure 2.4: Multiple Velocity Obstacles for B_1 and B_2; see (2.3) (Fiorini & Shiller, 1998).

While the classical VO paradigm assumes an infinite time horizon for collision prediction, more recent refinements frequently incorporate a time-bounded formulation to constrain the predictive horizon during collision evaluation (Guy et al., 2009; Rufli et al., 2013; van den Berg et al., 2011). This extension gives rise to the *Finite-Time Velocity Obstacle* (FVO) denoted as $VO_{B_i}^{A,\tau}$, where $\tau \in \mathbb{R} > 0$ represents the truncation time (Guy et al., 2009). Within this framework, agent A ensures collision avoidance with obstacle B_i by selecting a velocity vector $\mathbf{v}_A^{\text{new}}$ such that $\mathbf{v}_A^{\text{new}} \notin VO_{B_i}^{A,\tau}$. Geometrically, the truncated cone that defines $\text{VO}_{B_i}^{A,\tau}$ encapsulates all relative velocities leading to a potential intersection within the time interval $[0, \tau]$. This temporal truncation approach is a significant refinement to the classical VO formulation by introducing a finite-horizon predictive mechanism that allows the avoidance strategy to focus on imminent collision risks rather than relying solely on asymptotic safety guarantees.

The Finite–Time Velocity Obstacle is formulated as

$$VO_{B_i}^{A,\tau} = \Big\{ \mathbf{v}_A \;\Big|\; \exists\, t \in [0,\tau] \;\; \|\mathbf{r}_{AB_i} + t(\mathbf{v}_A - \mathbf{v}_{B_i})\| \leq R \Big\}. \tag{2.5}$$

In velocity space, (2.5) is obtained by truncating the cone with the circle D_τ of radius $r_\tau = R/\tau$ centered at $c = \mathbf{r}_{AB_i}/\tau$; the original disc D has radius R and center $c = \mathbf{r}_{AB_i}$.

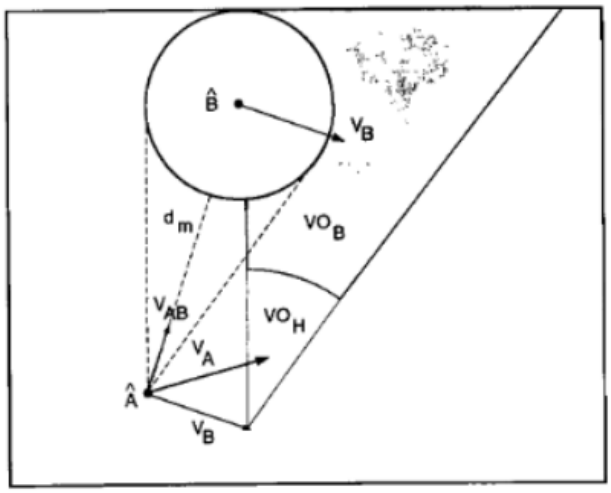

Figure 2.5: The velocity obstacle $\mathrm{VO}_B^{A,H}$ for a short time horizon; see (2.5) (Fiorini & Shiller, 1998).

Note that as $\tau \to \infty$, $VO_{B_i}^{A,\tau} \to VO_{B_i}^{A}$.

The concept of planar Velocity Obstacle can be easily generalized to $\mathbb{R}^3$ by replacing discs with spheres and rays with cones. Motion is described as $(p_A(t), \mathbf{v}_A(t)) \in T\mathbb{R}^3 \simeq \mathbb{R}^3 \times \mathbb{R}^3$. The relations (2.1)–(2.5) remain valid under this extension with the inner products and norms defined in $\mathbb{R}^3$.

2.2 From VO to reciprocity: RVO

The Reciprocal Velocity Obstacle (RVO) framework, introduced by (van den Berg et al., 2008). extends the classical VO paradigm to cooperative multi-agent environments. In this formulation, each agent assumes equal responsibility for avoiding collisions with its peers. Given two agents A and B moving with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$ respectively, each computes the velocity obstacle induced by the other (i.e., VO_B^A

and VO_A^B). In traditional VO setting, when a potential collision is detected, agent A selects a new velocity $\mathbf{v}_A^{\text{new}} = \mathbf{v}_A + \mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^2$ is a deviation vector such that $\mathbf{v}_A + \mathbf{w} \notin VO_B^A$. Similarly, agent B updates its velocity as $\mathbf{v}_B^{\text{new}} = \mathbf{v}_B + \mathbf{w}$ such that $\mathbf{v}_B + \mathbf{w} \notin VO_A^B$. This independent correction by both agents causes a relative displacement of $2\mathbf{w}$, leading to sub-optimal maneuvers since each agent moves further than necessary to avoid collision.

To ensure symmetry and prevent redundant motion, both agents must share the responsibility for collision avoidance. In the RVO formulation, this responsibility is equally distributed between the two agents by adjusting their individual velocities by half of the required deviation i.e., $\mathbf{v}_A^{\text{new}} = \mathbf{v}_A + \mathbf{w}/2$ and $\mathbf{v}_B^{\text{new}} = \mathbf{v}_B + \mathbf{w}/2$. This symmetric update guarantees reciprocal avoidance. The admissible set of velocities for A is then defined outside the RVO set given by

$$RVO_B^A = \left\{ \mathbf{v}_A^{\text{new}} \;\middle|\; 2\mathbf{v}_A^{\text{new}} - \mathbf{v}_A \in VO_B^A \right\}. \tag{2.6}$$

Geometrically, it corresponds to a cone translated by $(\mathbf{v}_A + \mathbf{v}_B)/2$. This cooperative sharing of responsibility prevents overly conservative maneuvers and forms the foundation for decentralized, real-time navigation among multiple autonomous agents.

2.3 Optimal Reciprocal Collision Avoidance (ORCA)

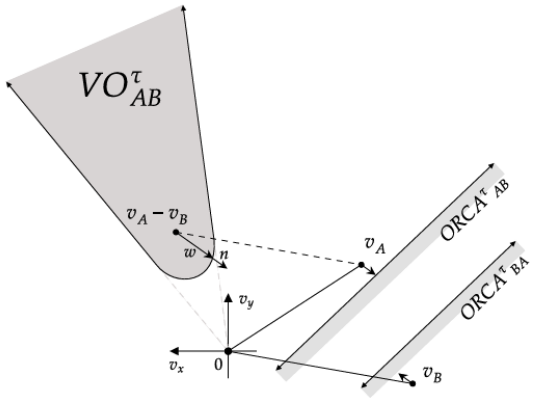

Figure 2.6: The $\text{ORCA}^{\tau}_{A|B}$ half-plane for A induced by B and its counterpart, $\text{ORCA}^{\tau}_{B|A}$, for B induced by A (van den Berg et al., 2011).

The Optimal Reciprocal Collision Avoidance (ORCA) framework proposed by (van den Berg et al., 2011). represents a significant improvement over the Velocity Obstacle (VO) and Reciprocal Velocity Obstacle (RVO) paradigms by formulating collision avoidance as a convex optimization problem. Instead of treating avoidance as a geometric exclusion in velocity space, ORCA constructs for each agent a half-plane of admissible velocities derived from the boundary of the finite-time VO cone. This makes reciprocity explicit and computationally light. For agents A and B with positions p_A, p_B, velocities v_A, v_B, radii r_A, r_B, and horizon $\tau > 0$, let $VO_B^{A,\tau}$ denote the time-horizon VO. Given the desired relative velocity

$$v_{\text{rel}}^{\text{opt}} = v_A^{\text{opt}} - v_B^{\text{opt}},$$

let u be the shortest vector that moves $v_{\text{rel}}^{\text{opt}}$ to the boundary/outside of $VO_B^{A,\tau}$, and n be the outward unit normal at that closest boundary point. This outward normal at the boundary point defines a linear constraint. Each agent adjustes its velocity by $\frac{1}{2}\mathbf{u}$ as they share the responsibility equally. This ensures fairness while minimizing deviation from desired motion.

$$\text{ORCA}_{A|B}^{\tau} = \Big\{ v \;\Big|\; \big(v - (v_A^{\text{opt}} + \tfrac{1}{2}u)\big) \cdot n \geq 0 \Big\}, \tag{2.7}$$

For agent A with neighbors $\mathcal{N}(A)$, the feasible set is the intersection of pairwise half-planes constrained by the maximum speed bound $v_{\max}$

$$\text{ORCA}_A^{\tau} = D(0, v_{\max}) \cap \bigcap_{B \in \mathcal{N}(A)} \text{ORCA}_{A|B}^{\tau}. \tag{2.8}$$

The intersection of all such half-planes forms a convex feasible region ORCA_A^{τ}. From this region, each agent selects its new velocity as

$$\mathbf{v}_A^{new} = \arg \min_{\mathbf{v} \in \text{ORCA}_A^{\tau}} \|\mathbf{v} - \mathbf{v}_A^{pref}\| \tag{2.9}$$

Since this region is convex and bounded by linear constraints, the optimal solution can be obtained efficiently using linear programming which produces smooth, non-oscillatory, and decentralized trajectories that guarantee safety for at least a finite horizon.

2.3.1 Failure modes and practical remedies

Although ORCA provides a sufficient condition for collision-free motion, it does not guarantee the existence of a feasible velocity solution in all configurations. In environments characterized by narrow passages, dense clusters or conflicting motion goals, the intersection of pairwise half-planes that defines the admissible velocity region may become empty. This situation is often referred to as the infeasibility problem and arises when no velocity satisfies all mutual avoidance constraints within the finite time horizon τ. To overcome this limitation in practical scenarios, implementations often rely on heuristic adjustments. A common strategy involves moderating the preferred velocity $\mathbf{v}_A^{\text{pref}}$ by either reducing its magnitude or biasing its direction toward locally unoccupied regions. Alternatively, reducing the prediction horizon τ or temporarily enlarging agent radii can encourage earlier deviation and thus restore feasible intersection. In extreme cases, a brief "stop-and-wait" behavior can ensure safety while surrounding agents clear the local bottleneck. These heuristic solutions have empirically shown to recover feasibility without compromising the theoretical safety guarantees of ORCA van den Berg et al., 2011.

2.4 VO-Based Planning for Robotic Manipulators

The *Velocity Obstacle* (VO) paradigm has emerged as a fundamental geometric framework for ensuring collision-free motion in dynamic multi-agent environments. By reasoning in the velocity space, each agent identifies regions corresponding to inadmissible velocities that would lead to potential collisions within a finite prediction horizon. This property has made VO-based methods particularly effective for mobile and aerial robots operating in densely populated settings. However, the adoption of VO reasoning within robotic manipulator domains remains relatively sparse. Traditional motion-planning schemes for

manipulators continue to rely on *Artificial Potential Fields* (APFs) and *Dynamic Movement Primitives* (DMPs) (Ijspeert et al., 2013; Khatib, 1986), which generate smooth trajectories by combining attractive and repulsive components in configuration space. While computationally efficient, these force-based methods often suffer from local minima and oscillatory behavior around obstacles (Koren & Borenstein, 1991) and fail to explicitly encode velocity constraints, limiting their performance in scenarios involving multiple dynamic obstacles.

To overcome these limitations, a VO-based trajectory planner was introduced by (Vesentini & Muradore, 2021) for a planar two-link (2R) manipulator, reformulating each link and joint as a *holonomic agent* within the velocity-space framework. In this formulation, the end-effector (EE) computes the velocity obstacle cones with respect to both the intermediate joint J_2 and an external moving obstacle, while the base joint J_1 remains fixed at the origin. The end-effector first determines its preferred velocity $\mathbf{v}_3^{\text{pref}}$ toward the goal position $\mathbf{P}_3^g$, and constructs the admissible velocity set V centered at its current position. This set is then refined to obtain the *Reduced Velocity Set* (RV) by removing directions that could induce self-collision between the manipulator links. The *Multiple Velocity Obstacle* (MVO_{EE}) is subsequently constructed as the union of collision cones with the elbow and the obstacle, i.e.,

$$MVO_{\text{EE}} = CC_{\text{EE},J_2} \cup CC_{\text{EE},O}. \tag{2.10}$$

The reduced admissible region $RMVO_{\text{EE}} = RV \setminus MVO_{\text{EE}}$ defines all feasible velocity vectors that guarantee instantaneous safety. The EE then solves the constrained optimization problem

$$\mathbf{v}_3^{\text{new}} = \arg \min_{\mathbf{v} \in RMVO_{\text{EE}}} \|\mathbf{v}_3^{\text{pref}} - \mathbf{v}\| \quad \text{s.t.} \quad \text{Link}_2(\mathbf{v}) \cap O = \emptyset, \tag{2.11}$$

This yields the next feasible velocity $\mathbf{v}_3^{\text{new}}$ that both minimizes deviation from the preferred motion and ensures no collision between Link 2 and the obstacle. The corresponding Cartesian update $\mathbf{P}_3^{\text{new}} = \mathbf{P}_3^{\text{curr}} + \mathbf{v}_3^{\text{new}} \Delta t$ is then propagated to the elbow configuration $\mathbf{P}_2^{\text{new}}$ via analytical inverse kinematics which ensures continuity of motion and non-singularity of the manipulator Jacobian.

Algorithm 1 VO 2R Planner

Input: Initial manipulator–obstacle–goal configuration
Set $d \leftarrow \|\mathbf{P}_3 - \mathbf{P}_3^g\|_2$
while $d > d_\epsilon$ **do**
  Acquire $\mathbf{P}_O$, $\mathbf{v}_O$, and $\mathbf{P}_3^g$
  Set $\mathbf{v}_3^{\text{pref}}$ for EE
  Create admissible velocities set V
  Adjust V creating RV according to $\mathbf{P}_1$
  Compute $RMVO_{\text{EE}}$
  for all $\mathbf{v} \in RMVO_{\text{EE}}$ **do**
    Compute $\mathbf{P}_3^{\text{new}} \leftarrow \mathbf{P}_3 + \mathbf{v}\Delta t$
    Compute $\mathbf{P}_2^{\text{new}}$ using IK equations
    Compute $\mathbf{P}_{\text{Link2}}^{\text{new}} \leftarrow \mathbf{P}_3^{\text{new}} - \mathbf{P}_2^{\text{new}}$
    Find $\mathbf{v}$ closest to $\mathbf{v}_3^{\text{pref}}$ such that $\text{Link}_2(\mathbf{v}) \cap O = \emptyset$
  end for
  Set $\mathbf{v}_3^{\text{new}} \leftarrow \mathbf{v}$
  Update $\mathbf{P}_3^{\text{new}} \leftarrow \mathbf{P}_3^{\text{curr}} + \mathbf{v}_3^{\text{new}}\Delta t$
  Compute $\mathbf{v}_2^{\text{new}}$
  Update $\mathbf{P}_2^{\text{new}} \leftarrow \mathbf{P}_2^{\text{curr}} + \mathbf{v}_2^{\text{new}}\Delta t$
  Update $d \leftarrow \|\mathbf{P}_3^{\text{new}} - \mathbf{P}_3^g\|_2$
end while
Set $\mathbf{v}_3^{\text{pref}} = 0$ ▷ Target reached

Algorithm 1 targets a planar 2R arm where the VO construction and IK checks are simpler. Most industrial manipulators, however, have 6–7 DoF (or more) which makes their motion more complex and tightly coupled. As a result, this 2R formulation does not transfer directly to high-DoF systems. In particular, extending VO to full-scale manipulators requires addressing issues such as configuration-dependent Jacobians and the need to ensure that any selected avoidance velocity corresponds to a valid and realizable end-effector motion in the joint space with a feasible IK solution.

In the next chapter, we will extend this idea to n-DoF manipulators by using the Fast-ORCA approach, which incorporates feasibility-aware IK checks, nearest-neighbor reasoning, and a position-based control scheme designed specifically for armed robots.

2.5 Extending Centralized DS Control with ORCA-Based Avoidance

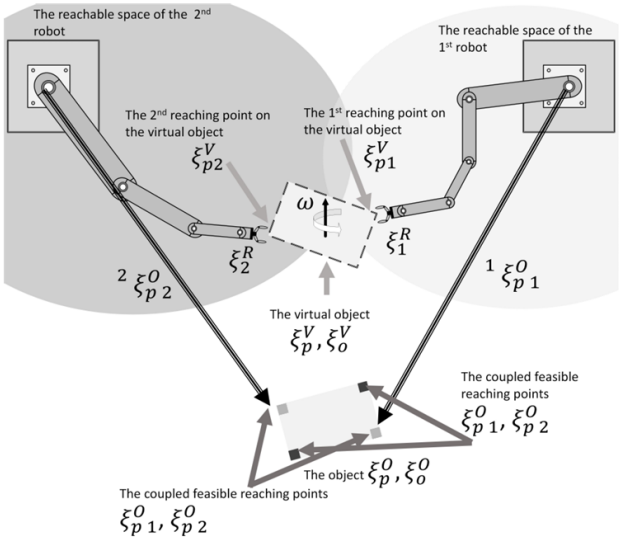

Figure 2.7: An illustration of the variables defined in the formulation is presented. The shaded reachable areas represent the feasible regions for grasping the object. Except for ${}^2\xi^O_{p_2}$ and ${}^1\xi^O_{p_1}$, all variables are expressed in the reference frame located at the desired intercept point, i.e., $\xi^O_i(T^*) = [0 \ \dots \ 0]^T \ \forall i \in \{p, o\}$. Note that $\boldsymbol{\omega}$ is the angular velocity of the virtual object, which is different from the numerical differentiation of the virtual object orientation; i.e., $\boldsymbol{\omega} \neq \dot{\xi}^V_o$ (Salehian et al., 2016).

Salehian, Figueroa, and Billard (Salehian et al., 2016, 2017, 2018) proposed a centralized scheme for coordinated motion of multiple manipulators that must simultaneously reach and grasp a moving object whose future trajectory is known only up to an online prediction. The key idea is to decouple object tracking and inter-arm synchronization by introducing a *virtual object* whose motion is governed by a stable dynamical system (DS). All arms are then commanded to track fixed grasp poses defined in the virtual-object frame, thereby ensuring implicit synchronization. Since a single DS produces the reference that every arm follows, this formulation is effectively centralized.

Let the real object have pose estimated from perception and prediction models.

$$\xi_O(t) = (p_O(t), R_O(t)) \in SE(3),$$

For each manipulator $i \in \{1, \ldots, K\}$, only a subset of the object's trajectory is reachable. Therefore, an interception time $\tau^\star$ is computed such that the predicted position $p_O(\tau^\star)$ lies within the intersection of the manipulators' workspaces:

$$p_O(\tau^\star) \in \bigcap_{i=1}^{K} \mathcal{W}_i.$$

A virtual pose is described as

$$\xi_V(t) = (p_V(t), R_V(t))$$

The motion of the virtual object is governed by a Linear Parameter-Varying (LPV) Dynamical System, which provides a flexible yet analytically stable representation of nonlinear motion. Instead of prescribing a simple first-order linear attraction toward the real object, the LPV formulation models both the translational and rotational evolution of the virtual object through sets of locally linear dynamics whose parameters vary smoothly with the system's state. In this framework, the virtual object's position and orientation evolve according to matrices whose values change with scheduling parameters computed from its current pose. The control inputs then guide the system so that the virtual object stays aligned with the predicted motion of the real object. Furthermore, because each local dynamic is linear and stable, the overall LPV-DS maintains global convergence while retaining the ability to approximate complex nonlinear motion patterns encountered during cooperative interception tasks (Ijspeert et al., 2013; Khansari-Zadeh & Billard, 2011; Khansari-Zadeh & Billard, 2012).

Each manipulator is assigned a fixed transformation that specifies its individual grasp configuration relative to the virtual object. This transformation defines the exact point and orientation at which the end-effector is intended to contact the virtual object. By expressing every manipulator's target pose in the coordinate frame of the virtual object, the system preserves a consistent grasp geometry across all arms. Consequently, any translational or rotational motion of the virtual object automatically produces corresponding updates to each manipulator's desired pose. This mechanism ensures that all end-effectors maintain their relative spatial arrangement.

Each manipulator uses a first-order tracking DS in task space:

$$\dot{x}_i(t) = \dot{x}_i^d(t) - A_i\big(x_i^d(t) - x_i(t)\big), \qquad A_i > 0, \tag{2.12}$$

where $x_i(t)$ is the pose of the ith end-effector and $x_i^d(t)$ is the pose of the corresponding virtual reaching point on the virtual object. Since $x_i^d(t)$ evolves with the virtual object state $\xi_V(t)$, this defines a tracking DS rather than a pure regulation problem.

To realize this behavior at the joint level, the desired task-space velocities of each manipulator are translated into joint-space commands through the robot's Jacobian matrix. A pseudoinverse of the Jacobian is used to obtain the minimum-norm joint velocity that achieves the required end-effector motion. Any remaining degrees of freedom that do not affect the primary end-effector task lie in the null space of the Jacobian. These are represented by an additional motion term that can be used for secondary objectives such as maintaining joint limits, optimizing manipulability or avoiding self-collisions.

Since each ξ_i^d is a direct function of the shared ξ_V, synchronization among manipulators happens naturally. The interception decision and reference generation are handled centrally through ξ_V. While the virtual-object DS guarantees stable and coordinated motion, it does not explicitly prevent collisions between manipulators. The formulation in (2.12) only ensures convergence of end-effectors to their grasp poses and leaves the null-space vector unspecified. As noted in the original work, null-space behaviors can encode secondary tasks such as joint-limit avoidance. However, inter-robot collision avoidance is not addressed. We propose one possible direction for extension by incorporating a repulsive velocity component derived from velocity-obstacle or ORCA constraints:

$$\dot{x}_i(t) = \dot{x}_i^d(t) - A_i\big(x_i^d(t) - x_i(t)\big) + \sum_{j \neq i} \phi_{ij}(x_i, x_j), \qquad A_i > 0, \tag{2.13}$$

where $\phi_{ij}(x_i, x_j)$ is a velocity term that enforces reciprocal collision avoidance between robots i and j. This augmentation would retain the DS stability properties while introducing decentralized collision

avoidance through either the task-space term ϕ_{ij} or the null-space vector that maximizes inter-link distances.

Chapter 3

Fast - ORCA for Manipulators

3.1 Overview

In environments where multiple manipulators must work together to perform precision sorting tasks, we propose an efficient strategy for collision avoidance and motion planning. This chapter introduces Fast-ORCA (FORCA), a decentralized approach that outputs the distance traveled instead of the agent's velocity. The method also incorporates feasibility checks by validating the IK solution of the goal position. To generate trajectories, we use the standard RRTConnect algorithm with a goal bias of 90%. The resulting trajectories are then post-processed so that the end-effector maintains the average velocity predicted by FORCA.

3.2 Fast-ORCA

We introduce a Fast-ORCA (FORCA) algorithm designed for collision avoidance and motion planning in multi-robot manipulator systems that perform precision sorting. The overall precision sort task is summarized in Algorithm 2. A key assumption of FORCA is that each agent plans its motion under the expectation that the other agents are using the same algorithm. The method is organized into two main components:

Determining the next position of the end effector. From the set of feasible, non-colliding velocities, the algorithm selects one that produces a collision-free position for the agent at a given time horizon t. The product of this velocity and time defines the agent's next preferred position. The end effector is required to reach this position within t while maintaining the velocity computed by ORCA. The agent selects a strategy that minimizes the deviation from its preferred position while ensuring that a valid IK solution exists for the chosen next pose.

Trajectory generation and refinement. Trajectory to the next position is generated using the standard RRTCONNECT planner which is configured with a goal bias of 90%. This trajectory is then refined at the joint level to ensure that the end-effector link maintains an average velocity consistent with the velocity determined in Step 1.

3.2.1 Fundamentals for FORCA

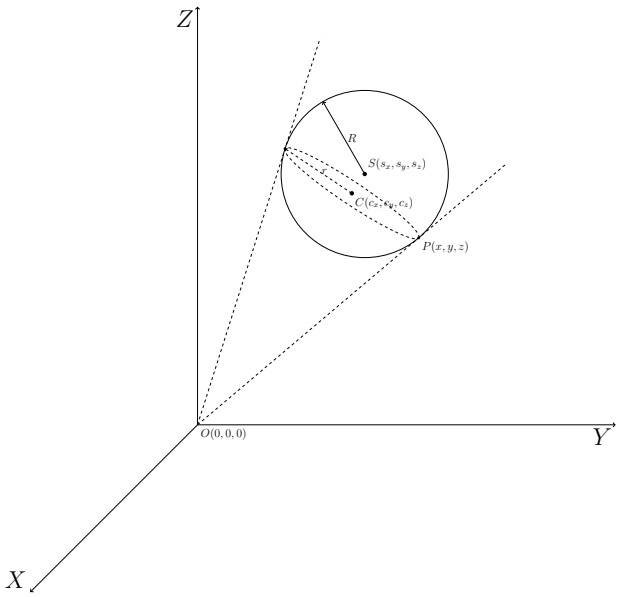

Figure 3.1: Geometric construction of the collision cone in $\mathbb{R}^3$. The origin $O(0, 0, 0)$ defines the relative reference frame of the observing agent, and the sphere of radius R centered at $S(s_x, s_y, s_z)$ represents the safety boundary around the other agent. Rays emanating from O that are tangent to this sphere define the boundary of the cone and their points of tangency $P(x, y, z)$ lie on a circle whose center is $C(c_x, c_y, c_z)$. Any relative velocity vector whose direction falls inside this cone will eventually intersect the sphere and thus leads to a potential collision, whereas directions outside the cone correspond to collision-free motion.

Consider a sphere of radius R with center $S = (s_x, s_y, s_z)$. The sphere is described by

$$(x - s_x)^2 + (y - s_y)^2 + (z - s_z)^2 = R^2 \tag{3.1}$$

Let $P(x, y, z)$ be a point of tangency seen from the origin $O(0, 0, 0)$. Since the radius SP is perpendicular to the tangent line OP, the orthogonality condition is

$$(x - s_x,\ y - s_y,\ z - s_z) \cdot (x,\ y,\ z) = 0 \tag{3.2}$$

Solving (3.1) and (3.2) to eliminate $x^2 + y^2 + z^2$ yields the plane that contains every such tangent point P:

$$s_x^2 + s_y^2 + s_z^2 - s_x\, x - s_y\, y - s_z\, z = R^2 \tag{3.3}$$

Intersecting this plane with the line OS locates the center of the circle of tangency points $C(c_x, c_y, c_z)$:

$$C(c_x, c_y, c_z) = \left(s_x - \frac{s_x R^2}{s_x^2 + s_y^2 + s_z^2},\ s_y - \frac{s_y R^2}{s_x^2 + s_y^2 + s_z^2},\ s_z - \frac{s_z R^2}{s_x^2 + s_y^2 + s_z^2} \right)$$

The radius r of this circle follows from the Pythagorean relation

$$CS^2 + CP^2 = SP^2,$$

which gives

$$r = R\sqrt{\frac{s_x^2 + s_y^2 + s_z^2 - R^2}{s_x^2 + s_y^2 + s_z^2}}$$

To parametrize the circle, take two orthogonal directions that lie in the plane (3.3). The equation for the plane normal is given by

$$\vec{n} = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix},$$

Now, choose the in-plane, mutually orthogonal basis

$$\vec{v}_1 = \begin{bmatrix} -s_y \\ s_x \\ 0 \end{bmatrix}, \qquad \vec{v}_2 = \vec{n} \times \vec{v}_1 = \begin{bmatrix} s_x s_z \\ s_y s_z \\ -s_x^2 - s_y^2 \end{bmatrix}$$

The parametric equation of the circle is therefore

$$\mathbf{P}(\theta) = \mathbf{C} + r\left(\frac{\vec{v}_1}{\|\vec{v}_1\|}\cos\theta + \frac{\vec{v}_2}{\|\vec{v}_2\|}\sin\theta\right), \qquad \theta \in [0, 2\pi). \tag{3.4}$$

The collision cone consists of all rays from O that pass through this circle.

After selecting a random angle $\theta \in \theta_0$ on the circle of tangency, let $\mathbf{V} = (v_x, v_y, v_z)$ be a random velocity in the velocity space. Define the unit vectors

$$\hat{\mathbf{p}} = \frac{\mathbf{P}(\theta_0)}{\|\mathbf{P}(\theta_0)\|}, \qquad \hat{\mathbf{v}} = \frac{\mathbf{V}}{\|\mathbf{V}\|}, \qquad \hat{\mathbf{s}} = \frac{\mathbf{S}}{\|\mathbf{S}\|}.$$

Then $\mathbf{V}$ lies inside the velocity obstacle (VO) iff

$$\hat{\mathbf{v}} \cdot \hat{\mathbf{s}} \geq \hat{\mathbf{p}} \cdot \hat{\mathbf{s}}. \tag{3.5}$$

This implies that the selected velocity $\mathbf{V}$ would result in a collision between the two agents (i.e., $\mathbf{V} \in \mathrm{VO}$).

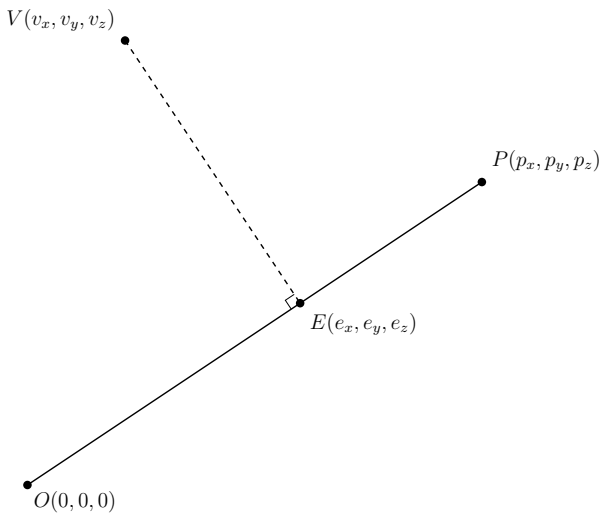

Figure 3.2: Strategy for selecting Optimal Escape Velocity

Once the collision cone has been characterized, FORCA must adjust any infeasible candidate velocity back into the admissible region. Instead of performing an expensive search over the entire velocity space, we use a simple geometric strategy where we project the current velocity onto the boundary ray of the cone that encompasses the candidate vector. This projection yields the smallest-norm change that moves the agent out of the collision cone. Since ORCA assumes reciprocal collision avoidance, each agent applies only half of this correction which leads to symmetric behavior without centralized coordination.

For a velocity vector $\mathbf{V}(v_x, v_y, v_z)$ lying inside the collision cone obtained by joining O and $P(\theta)$, the strategy we choose is to find the escape velocity vector that has the shortest distance from the collision cone. This is also the perpendicular distance from $\mathbf{V}$ to OP. Let us consider the point of intersection to be at $E(e_x, e_y, e_z)$.

$$E(e_x, e_y, e_z) = O + \lambda P$$

For the line segment OP, since $\overrightarrow{VE} \perp \overrightarrow{OP}$, we have

$$(\lambda P - V) \cdot P = 0$$

Solving for λ:

$$\lambda = \frac{V \cdot P}{\|P\|^2}$$

Thus,

$$E(e_x, e_y, e_z) = \frac{V \cdot P}{\|P\|^2} P$$

The escape velocity vector is then

$$\overrightarrow{VE} = E - V,$$

which represents the adjustment the agent must make in order to escape the collision.

Now, since both agents equally share the responsibility of escaping the collision cone, only 50% of the escape vector is applied by a single agent.

Hence, the next velocity for the agent is given by

$$\mathbf{v}_{\text{next}} \;=\; \mathbf{V} \;+\; \tfrac{1}{2}\,\overrightarrow{VE}. \tag{3.6}$$

Algorithm 2 FORCA-Enabled Precision Sort

Require: Set of manipulator agents $\mathcal{A}$, onion objects $\mathcal{O}$, error ε, threshold τ

```
// Set goal position for each manipulator agent
for all i ∈ A do
    goal_i ← CentralizedTaskAllocator(A, O, i)
end for
for all i ∈ A do
    // Continue until EE state is within goal limits
    while ||state_i − goal_i|| > ε do
        // Plan for the subgoal trajectory
        subgoal_traj_i ← FORCA(state_i, goal_i, τ)
        // Execute the trajectory asynchronously
        AsynchronousExecute(subgoal_traj_i)
        // Update the agent's current state
        state_i ← Update(state_i)
    end while
end for
```

Algorithm 3 FORCA($state_i, goal_i, \tau$)

for all $j \in \mathcal{A} \setminus \{i\}$ **do**
  $d_j \leftarrow distance(state_i, state_j)$
end for
$other\_agent \leftarrow \arg\min_{j \in \mathcal{A}\setminus\{i\}} d_j$
$agent\_distance \leftarrow d_{other\_agent}$
$v_{\text{pref}} \leftarrow \dfrac{(goal_i - state_i)}{\|goal_i - state_i\|} \times v_{\max}$
if $agent\_distance > \tau$ **then**
  `next_pos` $\leftarrow v_{\text{pref}} \cdot t$
  while $\neg$ hasValidIK(`next_pos`) **do**
    select $p \sim \mathcal{N}(0, \sigma)$
    $v \leftarrow v + p$
    `next_pos` $\leftarrow v \cdot t$
  end while
else
  $v_A \leftarrow \text{velocity}(state_i)$; $p_A \leftarrow \text{position}(state_i)$
  $v_B \leftarrow \text{velocity}(state_{\text{other\_agent}})$; $p_B \leftarrow \text{position}(state_{\text{other\_agent}})$
  Create set of admissible V such that $V \notin VO$ via equation (3.5)
  select v^* such that:
    $v \in V$
    $v^* \leftarrow \arg\min_v \ \|v \cdot t - v_{pref} \cdot t\|$
  `next_pos` $\leftarrow v^* \cdot t$
  while $\neg$ hasValidIK(`next_pos`) **do**
    $v \leftarrow \text{Neighbour}(v)$
    `next_pos` $\leftarrow v \cdot t$
  end while
end if
`// Generate a planned trajectory`
`Trajectory` $\leftarrow$ Planner(RRTConnect, `next_pos`)
`// Refine trajectory based on obtained velocity`
$planned\_traj \leftarrow \text{WaypointSpeedControl}(Trajectory, v)$

Although the sampling based planner produces collision-free joint-space paths to the selected subgoal pose, the resulting trajectories do not respect the end-effector speed specified by FORCA. In particular, the temporal spacing of waypoints is determined by the planner and may lead to highly non-uniform end-effector motion even if the geometric path is reasonable. To enforce consistency between the kinematic planner and the velocity-based collision avoidance layer, we introduce a WaypointSpeedControl post-processing step. This procedure retimes the joint-space waypoints so that the induced end-effector

motion follows the average speed requested by FORCA, while also computing joint velocities and accelerations that remain compatible with the controller's limits. In effect, FORCA chooses "how fast" the end effector should move, and WaypointSpeedControl retimes the initial planner trajectory so that the manipulator executes that motion in a dynamically meaningful way.

Algorithm 4 WaypointSpeedControl($trajectory, speed$)

Extract joint names and waypoints from trajectory plan
Initialize robot state using first waypoint
Compute initial end-effector transform T_{curr}
`// Adjust timestamps based on EE motion speed`
for each consecutive pair of waypoints (w_i, w_{i+1}) **do**
    Compute next EE transform T_{next} via forward kinematics
    Compute distance $d = \|T_{\text{next}} - T_{\text{curr}}\|$
    Update $w_{i+1}.t = w_i.t + d/speed$
    $T_{\text{curr}} \leftarrow T_{\text{next}}$
end for
`// Compute joint velocities and accelerations`
for each waypoint w_i **do**
    Compute $\Delta t_1, \Delta t_2$ from neighboring timestamps
    for each joint j **do**
        Estimate v_1, v_2 using finite differences
        Compute acceleration $a = 2(v_2 - v_1)/(\Delta t_1 + \Delta t_2)$
        Assign average velocity and computed acceleration to w_i
    end for
end for
return updated trajectory with adjusted timing and velocities

Chapter 4

Experiments and Results

4.1 Experiment Setup

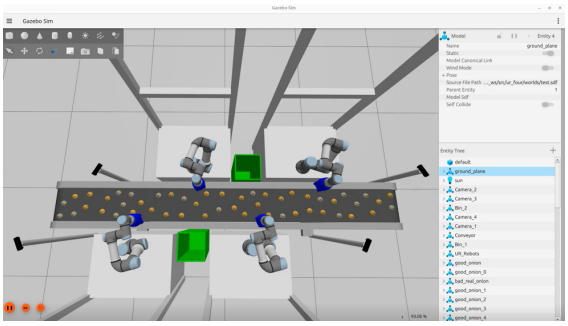

Figure 4.1: Far Inspection

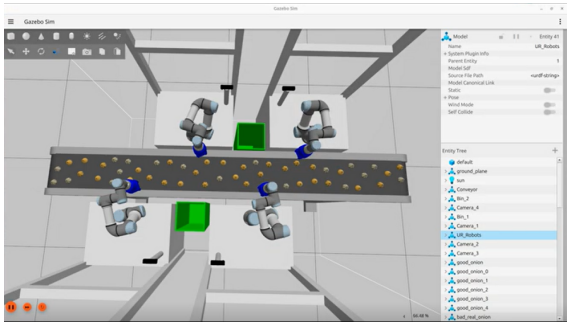

Figure 4.2: Near Inspection

The experimental setup involves a simulated workspace containing four UR3e robotic manipulators. All trials were conducted in Gazebo Harmonic, which was used to simulate both the physical interactions and the kinematic motion of the system. The workspace has two robots placed on each side of a central conveyor belt. Each pair shares a common bin. Four RGB cameras are strategically placed into the simulation with one assigned to each manipulator to monitor individual workspaces and provide high-resolution visual feedback during inspection. To evaluate the impact of camera placement on robot motion and task efficiency, two distinct sets of experiments were conducted — Far Inspection and Near Inspection. In the Far Inspection configuration, the robots were required to travel a greater distance toward the camera

position to perform the inspection, whereas in the Near Inspection setup, the inspection was carried out at a comparatively shorter distance from the camera.

The experimental task focuses on precision onion sorting. Task allocation is governed by a centralized allocator that assigns each onion to an appropriate robot based on a cost metric combining Euclidean proximity and inverse-kinematic feasibility. Following assignment, each manipulator performs a structured pick–inspect–place sequence: it retrieves the onion from the conveyor, transitions to its designated inspection waypoint under the associated camera for quality evaluation, and deposits the onion into the shared bin. Motion planning and collision avoidance for all manipulators are managed by the proposed Fast-ORCA framework which facilitates decentralized coordination and ensures safe and dynamically feasible motion within the multi-robot workspace.

System performance was evaluated by progressively increasing the task density on the conveyor. Five density levels were tested (30, 35, 40, 45 and 50 onions) for Far Inspection, and three density levels were tested (30, 40 and 50 onions) for Near Inspection case to examine how well the planner scales with workload. For each density, the onions were arranged in three different spatial configurations to account for randomness in task allocation. Each configuration was repeated to reduce uncertainty and the results presented later in this chapter correspond to the mean data collected across all trials.

4.2 Performance Baseline

To evaluate the comparative effectiveness of the proposed Fast-ORCA algorithm for the precision sorting task, we need to select a suitable baseline method. Since Fast-ORCA performs trajectory-level planning without incorporating other robots into the MoveIt planning scene, it is most appropriate to benchmark it against an approach that operates under similar assumptions. For this reason, a Greedy Straight-to-Goal (STG) method is selected as the baseline.

In the STG approach, each manipulator generates a direct, straight-line trajectory from its current pose to the assigned goal while passing through a predefined sequence of sub-goal waypoints similar to those used by Fast-ORCA. The planner does not consider the presence of neighboring manipulators

when computing trajectories which ensures a fair comparison of the algorithms under equivalent planning conditions. Unlike Fast-ORCA, however, the STG method lacks collision-awareness at the trajectory level and thus does not actively resolve potential inter-robot conflicts.

During experimentation, the STG baseline was executed without halting the simulation in the event of gripper or link-level collisions. This choice was intentional in order to allow the complete execution of all trials so that performance metrics mentioned in the subsequent section could be objectively compared. By maintaining identical starting conditions and motion goals across both approaches, the STG method provides a consistent reference for evaluating the gains achieved by the Fast-ORCA framework in terms of computation time, coordination and overall sorting performance.

4.3 Performance Evaluation

This section provides a performance analysis of the proposed Fast-ORCA framework relative to the baseline Greedy Straight-to-Goal (STG) technique. Our evaluation aims to measure the effectiveness of Fast-ORCA in enhancing cooperative motion and task efficiency inside a multi-robot sorting environment. To accomplish this, both methodologies were evaluated under uniform simulation settings, and their results were measured using two principal performance metrics: overall throughput and the frequency of inter-robot collisions recorded during execution.

The overall throughput parameter quantifies the total number of onions accurately sorted per unit of time and indicates the speed and efficiency of the multi-robot coordination process. The collision count metric quantifies the safety and stability of the generated trajectories by monitoring physical encounters between the robots' manipulators or end-effectors during the runtime.

These two parameters collectively provide a comprehensive assessment of the system's performance by encompassing both productivity and safety dimensions. The results associated with each indicator are examined and elaborated upon in the next section.

4.3.1 System Throughput

The comparative analysis of throughput across onion densities under the Far Inspection configuration reveals that STG marginally outperforms the Fast-ORCA (FORCA) algorithm in terms of raw sorting rate. As shown in Table 4.1, FORCA's throughput values are consistently lower by approximately 2 to 5 percent across all onion configurations. Specifically, the maximum deviation of −4.87 % occurs at the 45-onion configuration while the minimum deviation of −2.12 % is observed at the 30-onion configuration.

This decrease in throughput for the FORCA algorithm is due to integration of collision-avoidance maneuvers. Each robot dynamically modifies its route to ensure safe distance from adjacent manipulators which results in slight detours (and sometimes oscillations) and temporal delays. These avoidance behaviors inherently prolong the overall sorting time per robot in comparison to the STG approach which lacks such maneuvers.

Another consistent tendency noted across configurations is the reduction in total system throughput as onion density increases. As the onion density in the workspace increases, a greater fraction is situated further from the manipulators' initial reach zones. As a result, the robots must execute longer distances to finish each sorting cycle which extends motion length and lowers overall throughput.

Onion Density	**FORCA (onions/min)**	**STG (onions/min)**
30 onions	7.584	7.748
35 onions	7.132	7.452
40 onions	7.152	7.360
45 onions	6.564	6.900
50 onions	6.348	6.588

Table 4.1: Mean Total Throughput (4 Robots Combined) for FORCA and STG across Onion Configurations under the Far Inspection setup. Values represent the combined sorting rate of all four UR3e manipulators expressed in onions per minute and illustrates the overall system efficiency under varying workspace densities.

The throughput analysis for the Near Inspection configuration also demonstrates a notable performance improvement for both algorithms compared to the Far Inspection setup. As shown in Table 4.2, the proximity of the inspection area to the manipulators significantly reduces travel time and results in higher overall sorting rates across all onion densities.

Onion Density	FORCA Mean	FORCA Std Dev	STG Mean	STG Std Dev
30 onions	14.33	0.389	14.95	0.375
40 onions	12.21	0.296	13.92	0.523
50 onions	11.07	0.262	12.14	0.774

Table 4.2: Total Throughput (4 Robots Combined) for FORCA and STG under the Near Inspection configuration. Values represent the combined mean sorting rate of all four UR3e manipulators (expressed in onions per minute) along with standard deviations across three independent onion configurations.

Similar to the Far Inspection, STG continues to achieve slightly higher throughput values across the onion densities. FORCA's mean total throughput remains approximately 3–6 % lower than STG's across the evaluated onion densities. Specifically, at the 30-onion configuration, FORCA achieves 14.33 onions/min compared to STG's 14.95 onions/min (a deviation of −4.16 %) while at the 50-onion configuration, the deviation increases to −8.81 %.

The reduced throughput of FORCA can be again attributed to its dynamic collision avoidance maneuvers which introduces velocity adjustments to maintain safe inter-manipulator distances. These safety-oriented behaviors slightly increase execution time but ensure smoother coordination.

Another consistent observation is that the total system throughput decreases as the onion density increases. This occurs because onions positioned farther from the manipulators' initial reach zones requires longer end-effector motions and increases the time per every succesive sorting cycle. However, compared to the Far Inspection setup, the overall throughput remains substantially higher in the Near Inspection configuration due to the reduced travel distance between the pickup-inspection and inspection-bin locations.

4.3.2 Path Safety

The primary criterion for assessing the safety of the planned trajectories is the number of physical contacts or collisions that are registered between the robotic manipulators during the sorting process. Each collision event serves as a direct indicator of hazardous trajectory generation by failing to maintain a sufficient separation distance between the robots' end-effectors or links.

Tables 4.3 and 4.4 summarizes the collision counts obtained across several simulation trials for the Straight-to-Goal (STG) method for different onion densities under the Far Inspection and Near Inspection configuration respectively. Although the initial joint configurations of the robots were kept identical in all experiments, noticeable variation in the number of collisions was observed across runs. This can be attributed to the random spatial placement of onions on the conveyor which affects task allocation and occasionally leads to overlapping motion paths.

To quantify the variability in safety performance, collision counts across the three trials were examined for each onion density configuration. The mean number of collisions for the STG approach was observed to increase with onion density which indicates that denser workspaces inherently heighten the likelihood of inter-robot interference. For instance, in Far Inspection setup, configurations with 30 and 35 onions exhibited an average of approximately 2.0–2.7 collisions per run, whereas configurations with 40 and 45 onions showed significantly higher averages exceeding 5 collisions per run. The 50-onion configuration while being slightly lower in average collision count still demonstrated occasional contact events.

In contrast, the FORCA exhibited no collisions in any of the tested configurations. This consistent outcome demonstrates the strength of FORCA's decentralized, velocity-based avoidance mechanism which allows each manipulator to adjust its trajectory in real time to prevent contact and without relying on explicit inter-robot communication. Overall, these results indicate that FORCA offers a clear improvement in path safety and operational stability within multi-robot sorting environments.

Onion Density	Config-1	Config-2	Config-3
30 onions	3	3	0
35 onions	4	1	3
40 onions	7	4	8
45 onions	10	2	3
50 onions	4	0	5

Table 4.3: Collision counts for the Greedy Straight-to-Goal (STG) over onion densities under the Far Inspection configuration. Each onion configuration represents an independent simulation run and the reported values correspond to the total number of inter-robot collisions detected during execution.

Onion Density	Config-1	Config-2	Config-3
30 onions	0	2	2
40 onions	4	6	2
50 onions	7	3	8

Table 4.4: Collision counts for the Greedy Straight-to-Goal (STG) over onion densities under the Near Inspection configuration. Each onion configuration represents an independent simulation run and the reported values correspond to the total number of inter-robot collisions detected during execution.

Chapter 5

Conclusion

Decentralized motion planning continues to attract growing interest due to its broad applicability in both mobile robotics and robotic manipulation. In this work, we extended the principles of velocity-based planning that were originally developed for mobile robots to the domain of multi-robot manipulators. Building upon this foundation, we introduced the Fast-ORCA framework, a modified velocity-obstacle-based method tailored specifically for manipulator systems operating in shared workspaces.

Unlike conventional ORCA implementations that output the next velocity vector at each sense–act cycle, Fast-ORCA directly computes the next feasible position. This design choice addresses one of the key challenges in manipulator control where the actuator velocities of different joints are interdependent. By reasoning in position space, this method avoids the propagation of parent joint effects on child joints, leading to more stable and predictable motion in joint space.

Once the next position is selected, a full trajectory is generated using the existing motion planner available in MoveIt. This trajectory is then post-processed at the joint level so that the resulting end-effector velocity on average matches the target velocity derived from Fast-ORCA. To ensure feasibility of the solution, intermediate inverse-kinematics (IK) checks are incorporated at every iteration. These checks validate whether the computed next position is kinematically feasible; if not, the algorithm automatically explores neighboring tangent directions and selects an alternative velocity vector that yields a valid IK solution for the next pose.

The current collision model is limited to the gripper links of the manipulators and does not account for the remaining robot links. While this assumption simplifies computation since a large portion of MoveIt's runtime is spent on checking complex mesh-based self-collisions, it also narrows the scope of avoidance awareness. Although we have not yet compared this simplification against full-link collision models, future studies could benchmark Fast-ORCA against such modern baselines to quantify the trade-off between computational speed and safety coverage.

One key limitation of the current approach lies in the use of position-based joint trajectory controllers. These controllers require the joints to come to rest at every subgoal and results in noticeable discontinuities or jerkiness in motion. Future extensions could adopt a predictive execution strategy where the next subgoal is computed before the robot reaches its current one. This would allow non-zero terminal velocities and smoother transitions between subgoals that ultimately improves motion fluidity.

Additionally, recent developments in the velocity-obstacle family such as Hybrid Velocity Obstacles (HVO) can offer potential directions for improvement. Unlike standard ORCA which divides avoidance responsibility equally between agents, HVO allows variable responsibility assignments based on dynamic factors such as speed or maneuverability. Integrating such adaptive weighting into Fast-ORCA could make the framework more efficient in heterogeneous multi-robot settings. A learning-based policy could further refine this process that assignes responsibility dynamically according to the cost of altering each robot's state variables.

Occasionally, oscillatory behavior was observed when two manipulators shared a common subgoal position or operated in close proximity. Addressing this issue would require better heuristics for velocity adjustment or damping mechanisms to prevent repetitive motion.

Overall, Fast-ORCA demonstrates that velocity-based decentralized motion planning can be effectively extended to robotic manipulators. While the framework presents several areas for future refinement, it establishes a solid foundation for efficient, cooperative and scalable motion planning in multi-robot manipulation systems.

Bibliography

Chakravarthy, A., & Ghose, D. (1998). Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *28*(5), 562–574. https://doi.org/10.1109/3468.709600

Fiorini, P., & Shiller, Z. (1996). Robot motion planning in dynamic environments. In G. Giralt & G. Hirzinger (Eds.), *Robotics research* (pp. 237–248). Springer London. https://doi.org/10.1007/978-1-4471-1021-7_25

Fiorini, P., & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, *17*(7), 760–772. https://doi.org/10.1177/027836499801700706

Guy, S. J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., & Dubey, P. (2009). Clearpath: Highly parallel collision avoidance for multi-agent simulation. *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 177–187. https://doi.org/10.1145/1599470.1599494

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural computation*, *25*(2), 328–373. https://doi.org/10.1162/NECO_a_00393

Khansari-Zadeh, S. M., & Billard, A. (2011). Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, *27*(5), 943–957. https://doi.org/10.1109/TRO.2011.2159412

Khansari-Zadeh, S. M., & Billard, A. (2012). A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, *32*(4), 433–454. https://doi.org/10.1007/s10514-012-9287-y

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, *5*(1), 90–98. https://doi.org/10.1177/027836498600500106

Koren, Y., & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1398–1404 vol.2. https://doi.org/10.1109/ROBOT.1991.131810

Rufli, M., Alonso-Mora, J., & Siegwart, R. (2013). Reciprocal collision avoidance with motion continuity constraints. *IEEE Transactions on Robotics*, *29*(4), 899–912. https://doi.org/10.1109/TRO.2013.2258733

Salehian, S. S. M., Figueroa, N., & Billard, A. (2016). Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty. *Robotics: Science and Systems*.

Salehian, S. S. M., Figueroa, N., & Billard, A. (2017). Dynamical system-based motion planning for multi-arm systems: Reaching for moving objects. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 4914–4918. https://doi.org/10.24963/ijcai.2017/693

Salehian, S. S. M., Figueroa, N., & Billard, A. (2018). A unified framework for coordinated multi-arm motion planning. *The International Journal of Robotics Research*, *37*(10), 1205–1232. https://doi.org/10.1177/0278364918765952

Snape, J., Berg, J. v. d., Guy, S. J., & Manocha, D. (2011). The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, *27*(4), 696–706. https://doi.org/10.1109/TRO.2011.2120810

Snape, J., van den Berg, J., Guy, S. J., & Manocha, D. (2009). Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5917–5922. https://doi.org/10.1109/IROS.2009.5354821

van den Berg, J., Guy, S. J., Lin, M., & Manocha, D. (2011). Reciprocal n-body collision avoidance. In C. Pradalier, R. Siegwart, & G. Hirzinger (Eds.), *Robotics research* (pp. 3–19). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19457-3_1

van den Berg, J., Lin, M., & Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. *2008 IEEE International Conference on Robotics and Automation*, 1928–1935. https://doi.org/10.1109/ROBOT.2008.4543489

Vesentini, F., & Muradore, R. (2021). Velocity obstacle-based trajectory planner for two-link planar manipulators. *2021 European Control Conference (ECC)*, 690–695. https://doi.org/10.23919/ECC54610.2021.9655184

Vesentini, F., Muradore, R., & Fiorini, P. (2024). A survey on velocity obstacle paradigm. *Robotics and Autonomous Systems*, *174*, 104645. https://doi.org/10.1016/j.robot.2024.104645

Wilkie, D., van den Berg, J., & Manocha, D. (2009). Generalized velocity obstacles. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5573–5578. https://doi.org/10.1109/IROS.2009.5354175

Appendix A

Simulation Parameters and Additional Resources

This appendix summarizes the parameters and supporting resources used in the experimental evaluations presented in Chapter 4.

Simulation Configuration

- **Type of robot:** UR3e (Universal Robots)
- **Number of robots:** 4
- **Number of cameras:** 4
- **Number of bins:** 2
- **Onion densities tested:** 30, 35, 40, 45, and 50 onions per configuration under Far Inspection; and 30, 40 and 50 onions per configuration under Near Inspection.
- **Total configurations per density:** 3

Supplementary Material

Simulation videos illustrating both the Fast-ORCA (FORCA) and the Greedy STG approaches are available online at the following link:

Google Drive: Simulation Videos for Fast-ORCA and STG Approaches