COLLISION AVOIDANCE AND FORMATION CONTROL FOR MULTI-AGENT SYSTEMS

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

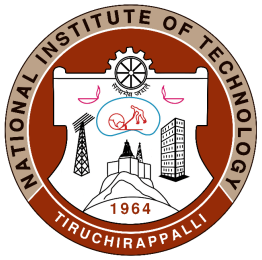
B.Tech

in

Instrumentation and Control Engineering

By
Rishab Balasubramanian - 110116070

Lalita Soundari - 110116046



Department of Instrumentation And Control Engineering
National Institute of Technology
Tiruchirapalli - 620015

June 2020

BONAFIDE CERTIFICATE

This is to certify that the project titled **COLLISION AVOIDANCE AND FORMATION CONTROL FOR MULTI-AGENT SYSTEMS** is a bonafide record of the work done by

Rishab Balasubramanian - 110116070
Lalita Soundari - 110116046

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Instrumentation And Control Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2019-2020.

Dr. M. Umapathy
Guide

Dr. G. Uma
Head of the Department

Project Viva-Voce held on _____

Internal Examiner

External Examiner

1. ABSTRACT

We present a formal approach to reciprocal n-body collision avoidance, where multiple mobile robots need to avoid collisions with each other while moving in a common workspace. In the formulation presented, each robot acts fully independently, and does not communicate with other robots. Based on the definition of velocity obstacles, we derive sufficient conditions for collision-free motion by reducing the problem to solving a low-dimensional linear program. The algorithm is based upon the Optimal Reciprocal Collision Avoidance (ORCA). We test our approach in simulation scenarios and compute collision-free actions for all of them in only a few milliseconds, using software-in-the-loop Gazebo ROS simulatior.

Keywords: Multi-agent, collision avoidance, robot, ORCA, reciprocal velocity, reciprocal colision avoidance, optimization

2. ACKNOWLEDGEMENTS

We wish to express our sincere thanks to our research guide, Dr. M. Umapathy, for providing us with all the expertise and knowledge for the implementation of our project work and for having faith in our aptitude through this entire period. We would also like to extend our gratitude to all the faculty and staff members of the department of Instrumentation and Control Engineering for their support and guidance.

Contents

3. LIST OF FIGURES

4. INTRODUCTION

4.1 Multi-Agent Systems

A multi-agent system (MAS or "self-organized system") is a computerized system composed of multiple interacting intelligent agents. Multi-agent systems can solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Multi-agent systems consist of agents and their environment. Typically multi-agent systems research refers to software agents. However, the agents in a multi-agent system could equally well be robots, humans or human teams. A multi-agent system may contain combined human-agent teams. Agents can be divided into types spanning simple to complex. Categories include:

1.Passive agents or "agent without goals" (such as obstacle, apple or key in any simple simulation)

2.Active agents with simple goals (like birds in flocking, or wolf–sheep in prey-predator model)

Agent environments are split in two: Discrete and Continuous. Agent environments can also be organized according to properties such as accessibility (whether it is possible to gather complete information about the environment), determinism (whether an action causes a definite effect), dynamics (how many entities influence the environment in the moment), discreteness (whether the number of possible actions in the environment is finite), episodicity (whether agent actions in certain time periods influence other periods), and dimensionality (whether spatial characteristics are important factors of the environment and the agent considers space in its decision making). Agent actions are typically mediated via an appropriate middleware. This middleware offers a first-class design abstraction for multi-agent systems, providing means to govern resource access and agent coordination.

4.2 Collision Avoidance & Formation Control

Multi-robot systems are designed to achieve tasks by collaboration. Multi-agent Navigation is a field that has recently gained increasing attention both in the robotics and the control communities, due to the need for autonomous control of more than one mobile robotic agents in the same workspace. A key requirement for their efficient operation is good coordination and reciprocal collision avoidance. Collision avoidance is a fundamental problem in robotics. Moving a vehicle on a collision-free path is a well-studied problem in robot navigation. The problem can generally be defined in the context of an autonomous mobile robot navigating in an environment with obstacles and/or other moving entities, where the robot employs a continuous cycle of sensing and acting. In each cycle, an action for the robot must be computed based on local observations of the environment, such that the robot stays free of collisions with the obstacles and the other moving entities, while making progress towards a goal. Note that the problem of (local) collision-avoidance differs from motion planning, where the global environment of the robot is considered to be known and a complete path towards a goal configuration is planned at once, and collision detection, which simply determines if two geometric objects intersect or not.

The problem of collision avoidance has been well studied for one robot avoiding static or moving obstacles. In this thesis, we address the more involved and less studied problem of reciprocal n-body collision avoidance, where collisions need to be avoided among multiple robots (or any decision-making entities). Basically, similar approaches as in the single robot cases can be applied in the context of collision avoidance for multiple robots. However, the increase in robot density and collaborative interaction needs methods that scale well with the number of robots. While most efforts in the past had focused on centralized planning, specific real-world applications have lead researchers throughout the globe to turn their attention to decentralized concepts. Among the various specifications that the control design aims to impose on the multi-agent team, formation convergence and achievement of flocking behavior are two objectives that have been pursued extensively in the last few years. Decentralized control helps lowering computational cost and introduces additional robustness and flexibility to the multi-robot system.

This problem has important applications in many areas in robotics, such as multi-robot navigation and coordination among swarms of robots. It is also an key component in crowd simulation for computer graphics and VR, modeling of non-player characters in AI, studying flocks of birds and fish in biology, and real-time (air) traffic

control. We develop a fast method based on the Optimal Reciprocal Collision Avoidance (ORCA) that simultaneously determines actions for many (possibly thousands of) robots that each may have different objectives. The actions are computed for each robot independently, without communication among the robots or central coordination. Yet, we show that the method guarantees collision-free motion for each of the robots.

We use a simplified robot model, where each robot is assumed to have a sim- ple shape (circular or convex polygon) moving in a two-dimensional workspace. Furthermore, we assume that the robot is holonomic, i.e. it can move in any direc- tion, such that the control input of each robot is simply given by a two-dimensional velocity vector. Also, we assume that each robot has perfect sensing, and is able to infer the exact shape, position and velocity of obstacles and other robots in the environment.

We use a rigorous approach for reciprocal n-body collision avoidance that provides a sufficient condition for each robot to be collision-free for at least a fixed amount of time into the future, only assuming that the other robots use the same collision-avoidance protocol. Our approach is velocity-based. That implies that each robot takes into account the observed velocity of other robots in order to avoid collisions with them, and also that the robot selects its own velocity from its velocity space in which certain regions are marked as 'forbidden' because of the presence of other robots. The formulation infers for each other robot a half-plane (in velocity-space) of velocities that are allowed to be selected in order to guarantee collision avoidance. The robot then selects its optimal velocity from the intersection of all permitted half-planes, which can be done efficiently using linear programming. Under certain conditions with densely packed robots, the resulting linear program may be infeasible, in which case we select the 'safest possible' velocity using a three-dimensional linear program. We experimented with our approach on several complex simulation scenarios containing thousands of robots. As each robot is independent, we can fully paral- lellize the computation of the actions for each robot and report very fast real-time running times. Furthermore, our experiments show that our approach achieves convincing motions that are smooth and collision-free.

The main feature of formation control is the cooperative nature of the equilibria of the system. Agents must converge to a desired configuration encoded by the inter-agent relative positions. On the other hand, flocking behavior involves convergence of the velocity vectors and orientations of the agents to a common value at steady state. The main feature of formation control is the cooperative nature of the equilibria of the system. Agents must converge to a desired configuration encoded by the inter-agent

relative positions.In most cases, formation convergence involves kinematic models of the agents' motion, while flocking behavior dynamic ones. Hence the problem of flocking motion has rarely been examined in the context of kinematic models of motion. Formation infeasibility is equivalent to the case when inter-agent objectives cannot occur simultaneously in the state space. By decoupling the two objectives (collision avoidance and formation convergence) it is shown that undercertain assumptions formation infeasibility forces the agents velocity vectors to a common value at steady state.

5. LITERATURE REVIEW

Collision avoidance is central to many robotics applications,such as multiagent coordination, autonomous navigation through human crowds, pedestrian motion prediction ,and computer crowd simulation. Yet, finding collision-free,time efficient paths around other agents remains challenging,because it may require predicting other agents' motion andanticipating interaction patterns, through a process that needsto be computationally tractable for real-time implementation. If there is a reliable communication network for agents to broadcast their intents (e.g. goals, planned paths), then collision avoidance can be enforced through a centralized planner. For instance, collision avoidance requirements can be formulated as separation constraints in an optimization framework for finding a set of jointly feasible and collision-free paths. However, centralized path planning methods can be computationally prohibitive for largeteams. To attain better scalability, researchers have also proposed distributed algorithms based on message-passing schemes which resolve local (e.g. pairwise) conflicts without needing to form a joint optimization problem betweenall members of the team. Agents would need to cooperate without necessarily having knowledge of the other agent's intents.

Many approaches assume the observed obstacles to be static (i.e. non-moving) [1, 2, 3, 4, 5, 6, 7], and compute an immediate action for the robot that would avert collisions with the obstacle, in many cases taking into account the robot's kinematics and dynamics. If the obstacles are also moving, such approaches typically repeatedly "replan" based on new readings of the positions of the obstacles. This may work well if the obstacles move slower than the robot, but among fast obstacles (such as crossing a highway), the velocity of the obstacles need to be specifically taken into account. This problem is generally referred to as "asteroid avoidance", and approaches typically extrapolate the observed velocities in order to estimate the future positions of obstacles [8, 9, 10, 11, 12, 13].

The problem of collision avoidance becomes harder when the obstacles are not simply moving without considering their environment, but are also intelligent decision-

making entities that try to avoid collisions as well. Simply considering them as moving obstacles may lead to oscillations if the other entity considers all other robots as moving obstacles as well [14, 15]. Therefore, the reactive nature of the other entities must be specifically taken into account in order to guarantee that collisions are avoided. Yet, the robot may not be able to communicate with other entities and may not know their intents.

Reciprocal Collision Avoidance (RVO) [3], a collaborative collision avoidance method based on velocity obstacles, was shown to be solved efficiently through a low-dimensional linear program, which results in completeness and a speed-up of the algorithm. Each robot makes a similar collision avoidance reasoning and collision-free motion is guaranteed all time, but holonomic robots are assumed and oscillations in the form of reciprocal dances can occur. Theew also exist extensions that combines both the concepts of basic velocity obstacles and RVO to reduce the amount of oscillations. In addition, robot kinematics and sensor uncertainty are included by enlarging the velocity cones, even though a formal proof of collision-free motion is not given. As the method requires extensive numeric computation and relies on probabilistic sampling, it may fail to find an existing feasible solution. There are also solutions for differential-drive robots by applying ORCA on the robot's virtual center. This is in contrast to our approach of extending the robot's radius, which allows to decrease its extension to zero in crowded scenarios. [13] also relies on the mapping between desired holonomic and non-holonomic velocities, but is different from ours in how it is derived; moreover it further constrains the motion of the robots. Another reactive collision avoidance method for unicycles based on velocity obstacles was presented in [9], where inputs are obtained by a weighted combination of the closest collision in normal and tangential directions.

Existing workon non-communicating collision avoidance can be broadly classified into two categories, reaction-based and trajectory-based. Reaction-based methods specify one-step interaction rules for the current geometric configuration. For example, reciprocal velocity obstacle (RVO) is a reaction-based method that adjusts each agent's velocity vector to ensure collision-free navigation. However, since reaction-based methods do not consider evolution of the neighboring agents' future states, they are short-sighted in time and have been found to create oscillatory and unnatural behaviors in certain situations.In contrast, trajectory-based methods explicitly account for evolution of the joint (agent and neighbors) future states by anticipating other agents' motion. A subclass of non-cooperative approaches propagates the other agents' dynamics forward in time, and then plans a collision-free path with respect

to the other agents' predicted paths. However, in crowded environments, the set of predicted paths often marks a large portion of the space untraversable/unsafe, which leads to the freezing robot problem. A key to resolving this issue is to account for interactions, such that each agent's motion can affect one another. Thereby, a subclass ofcooperative approaches has been proposed, which first infers the other agents' intents (e.g. goals), then plans a set of jointly feasible paths for all neighboring agents in the environment.

Velocity obstacles (VO) [5] have been a successful velocity-based approach to avoid collisions with moving obstacles; they provide a sufficient and necessary con- dition for a robot to select velocity that avoids collisions with an obstacle moving at a known velocity. This approach was extended for robot-robot collision avoidance with the definition of Reciprocal Velocity Obstacles (RVO), where both robots were assumed to select a velocity outside the RVO induced by the other robot. However, this formulation only guarantees collision-avoidance under specific conditions, and does not provide a sufficient condition for collision-avoidance in general. 2 In this paper, we present the principle of optimal reciprocal collision avoidance (ORCA) that overcomes this limitation; ORCA provides a sufficient condition for multiple robots to avoid collisions among one another, and thus can guarantee collision-free navigation. We note that it is possible to provide a sufficient and necessary condition for multiple (say n) robots to select collision-avoiding velocities, by defining a composite velocity obstacle in the 2n-dimensional space of velocities of all n robots [1]. However, this is not only computationally impractical, it also requires central coordination among robots. This is incompatible with the type of distributed multi-robot navigation we focus on in this paper, in which each robot independently and simultaneously selects its velocity from its own 2-D velocity space.

Cooperative trajectory-based methods often produce paths with better quality (e.g. shorter time forall agents to reach their goal) than that of reaction-basedmethods. However, planning paths for all other agentsis computationally expensive, and such cooperative approach typically requires more information than is readily available(e.g. other agent's intended goal). Moreover, due to modelar and measurement uncertainty, the other agents' actual paths might not conform to the planned/predicted paths, particularly beyond a few seconds into the future. Thus, trajectory-based methods also need to be run at a high (sensor update) rate, which exacerbates the computational problem.The major difficulty in multiagent collision avoidance isthat anticipating evolution of joint states (paths) is desirablebut computationally prohibitive.

6. PROBLEM DEFINITION

In this thesis we work to replicate the results as stated by van den Berg et.al. in their paper "Reciprocal n-Body Collision Avoidance". The problem discussed in the paper, and subsequently in this thesis is formally defined as follows. Let there be a set of n robots sharing an environment. For simplicity, we shall assume that they are restricted to move in the XY- plane of $\mathbb{R}^2$.Each robot A has a current position p_A (the center of its disc), a current velocity v_A and a radius r_A . These parameters are part of the robot's external state, i.e. we assume that they can be observed by other robots. In this thesis we assume that the radius of all robots are equal. Furthermore, each robot has a maximum speed v_A^{max}, and a preferred velocity v_A^{pref}. The maximum velocity of the robot is bound by its physical constraints, while we shall define the preferred velocity as the velocity the robot shall assume under the abscence of any obstacles. In this work, we assume the maximum velocities of all robots to be equal.

The task is for each robot A to independently (and simultaneously) select a new velocity v_{new}^A for itself such that all robots are guaranteed to be collision-free for at least a preset amount of time τ when they would continue to move at their new velocity. As a secondary objective, the robots should select their new velocity as close as possible to their preferred velocity. The robots are not allowed to communicate with each other, and can only use observations of the other robot's current position and velocity. However, each of the robots may assume that the other robots use the same strategy as itself to select a new velocity.

7. FORMULATION OF RECIPROCAL COLLISION AVOIDANCE

For two robots A and B, the velocity obstacle $VO^{\tau}_{A|B}$ (called the velocity obstacle of A induced by B in time window τ) is the set of all relative velocities of A with respect to B that will result in a collision between A and B at some moment before time τ. It is formally defined as follows. Let D(p, r) denote an open disc of radius r centered at p;

$$D(p, r) = \{\mathbf{q} | \|\mathbf{q} - p\| \leq r\} \tag{1}$$

then:

$$VO^{\tau}_{A|B} = \{\mathbf{v} | \exists t \in [0, \tau] : t\mathbf{v} \in D(p_A - p_A, r_A + r_B\} \tag{2}$$

where p_B, r_B, p_A, r_A are the positions and radii of A and B respectively.

Let v_A and v_B be the current velocities of robots A and B, respectively. The definition of the velocity obstacle implies that if $v_A - v_B \in VO^{\tau}_{A|B}$, or equivalently if $v_B - v_A \in VO^{\tau}_{B|A}$, then A and B will collide at some moment before time τ if they continue moving at their current velocity. Conversely, if $v_A - v_B \notin VO^{\tau}_{A|B}$, robot A and B are guaranteed to be collision-free for at least τ time.

More generally, let $X \oplus Y$ denote the Minkowski sum of sets X and Y such that

$$X \oplus Y = x + y | x \in X, y \in Y, \tag{4.1}$$

then for any set V_B , if $v_B \in V_B$ and $v_A \notin VO^{\tau}_{A|B} \oplus V_B$, then A and B are guaranteed to be collision-free at their current velocities for at least τ time. This leads to the definition of the set of collision-avoiding velocities for A given that B selects its velocity from V B (refer Fig. 1(c)):

$$CA^{\tau}_{A|B}(VB) = \mathbf{v} | \mathbf{v} \notin VO^{\tau}_{A|B} \oplus V_B \tag{3}$$

We thus call this set $CA^{\tau}_{A|B}$ as the collision avoidance set of velocities for A relative

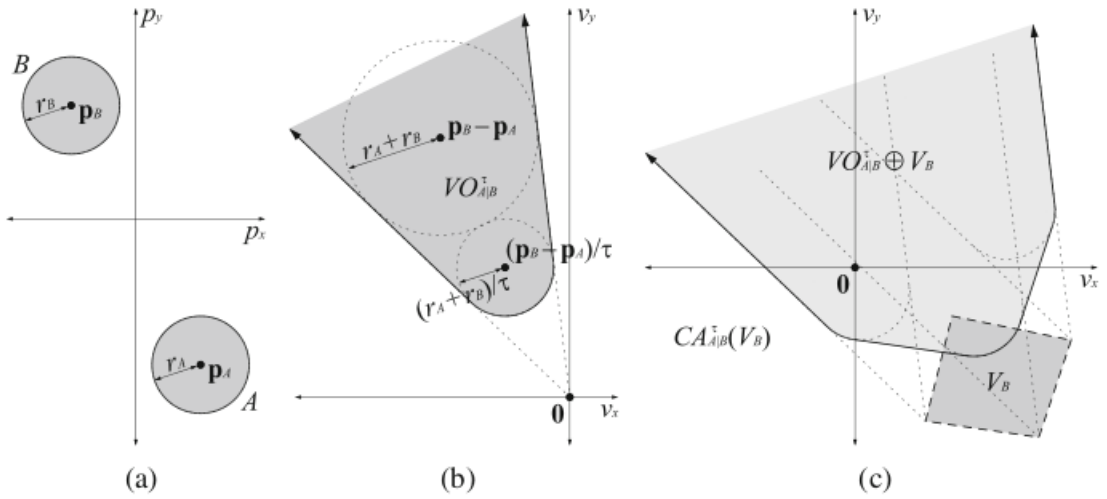


Fig. 1 **(a)** A configuration of two robots A and B. **(b)** The velocity obstacle $VO^{\tau}_{A|B}$ (gray) can geometrically be interpreted as a truncated cone with its apex at the origin (in velocity space) and its legs tangent to the disc of radius $r_A + r_B$ centered at $\mathbf{p}_B - \mathbf{p}_A$. The amount of truncation depends on the value of τ; the cone is truncated by an arc of a disc of radius $(r_A + r_B)/\tau$ centered at $(\mathbf{p}_B - \mathbf{p}_A)/\tau$. The velocity obstacle shown here is for $\tau = 2$. **(c)** The set of collision-avoiding velocities $CA^{\tau}_{A|B}(V_B)$ for robot A given that robot B selects its velocity from some set V_B (dark gray) is the complement of the Minkowski sum (light gray) of $VO^{\tau}_{A|B}$ and V_B.

to B. We call a pair of sets v_A and v_B of velocities for A and B reciprocally collision-avoiding if $v_A \in CA^{\tau}_{A|B}$ and $v_B \in CA^{\tau}_{B|A}$. If $v_A = CA^{\tau}_{A|B}(v_A)$ and $v_B = CA^{\tau}_{B|A}(v_A)$, we call v_A and v_B reciprocally maximal.

Given the definitions above, we would like to choose sets of permitted velocities v_A for A and v_B for B such that $CA^{\tau}_{A|B}(v_B) = v_A$ and $CA^{\tau}_{B|A}(v_A) = v_B$, i.e. they are reciprocally collision-avoiding and maximal and guarantee that A and B are collision-free for at least τ time. Also, because A and B are individual robots, they should be able to infer their set of permitted velocities without communication with the other robot. There are infinitely many pairs of sets v_A and v_B that obey these requirements, but among those we select the pair that maximizes the amount of permitted velocities "close" to optimization velocities V_A^{opt} for A and v_B^{opt} for B. We denote these sets $ORCA^{\tau}_{A|B}$ for A and $ORCA^{\tau}_{B|A}$ for B, and formally define them as follows. Let —V— denote the measure (i.e. area in R 2) of set V, then: $ORCA^{\tau}_{A|B}$ and $ORCA^{\tau}_{B|A}$ are defined such that they are reciprocally collision-avoiding and maximal, i.e. $CA^{\tau}_{A|B}$

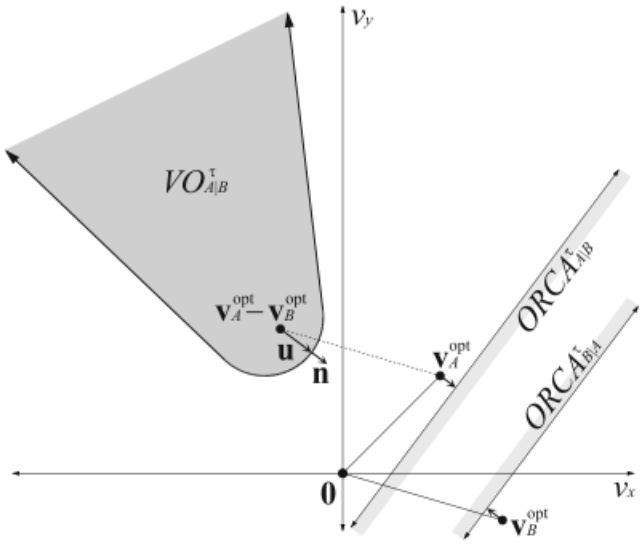


Fig. 2 The set $ORCA^{\tau}_{A|B}$ of permitted velocities for A for optimal reciprocal collision avoidance with B is a half-plane delimited by the line perpendicular to $\mathbf{u}$ through the point $\mathbf{v}_A^{\text{opt}} + \frac{1}{2}\mathbf{u}$, where $\mathbf{u}$ is the vector from $\mathbf{v}_A^{\text{opt}} - \mathbf{v}_B^{\text{opt}}$ to the closest point on the boundary of $VO^{\tau}_{A|B}$.

$(ORCA^{\tau}_{B|A}) = ORCA^{\tau}_{A|B}$ and $CA^{\tau}_{B|A}$ $(ORCA^{\tau}_{A|B}) = ORCA^{\tau}_{B|A}$, and such that for all other pairs of sets of reciprocally collision-avoiding velocities v_A and v_B (i.e. $v_A \in CA^{\tau}_{A|B}(v_B)$ and $v_B \in CA^{\tau}_{B|A}(v_A)$, and for all radii r ¿ 0,

$$|ORCA^{\tau}_{A|B} \cap D(v_A^{opt}, r)| = |ORCA^{\tau}_{B|A} \cap D(v_A^{opt}, r)| \tag{4}$$

This means that $ORCA^{\tau}_{A|B}$ and $ORCA^{\tau}_{B|A}$ contain more velocities close to v_A^{opt} and v_B^{opt}, respectively, than any other pair of sets of reciprocally collision-avoiding velocities. Also, the distribution of permitted velocities is "fair", as the amount of velocities close to the optimization velocity is equal for A and B. We can geometrically construct $ORCA^{\tau}_{A|B}$ and $ORCA^{\tau}_{B|A}$ as follows. Let us assume that A and B adopt velocities v_A^{opt} and v_B^{opt}, respectively, and let us assume that that causes A and B to be on collision course, i.e. $v_A^{opt} - v_B^{opt} \in VO_{A|B}$. Let u be the vector from $v_A^{opt} - v_B^{opt}$ to the closest point on the boundary of the velocity obstacle, defined as:

$$u = \underset{v \in \partial VO^{\tau}_{A|B}}{\arg\min} \left(\left\| v - (v_A^{opt} - v_B^{opt}) \right\| - (v_A^{opt} - v_B^{opt}) \right) \tag{4.2}$$

and let n be the outward normal of the boundary of $VO^{\tau}_{A|B}$ at point $(v_A - v_B) + u$. Then, u is the smallest change required to the relative velocity of A and B to avert collision within τ time. To "share the responsibility" of avoiding collisions among the robots in a fair way, robot A adapts its velocity by (at least) $\frac{1}{2}u$ and assumes that B takes care of the other half. Hence, the set $ORCA^{\tau}_{A|B}$ of permitted velocities for A is the half-plane pointing in the direction of n starting at the point $v_A + \frac{1}{2}u$. More formally:

$$ORCA^{\tau}_{A|B} = \{v|(v - (v^{opt}_A + \frac{1}{2}u)).n \geq 0\} \tag{5}$$

The set $ORCA^{\tau}_{B|A}$ for B is defined symmetrically (see Fig. 2). The above equations also apply if A and B are not on a collision course when adopting their optimization velocities, i.e. $v^{opt}_A - v^{opt}_B \in VO^{\tau}_{A|B}$. In this case, both robots each will take half of the responsibility to remain on a collision-free trajectory. This set gives us the hyperplane of collision avoiding velocities. Every velocity in this hyperplane ensures collision avoidance.

8. APPLYING ORCA TO MULTI-AGENT SYSTEMS

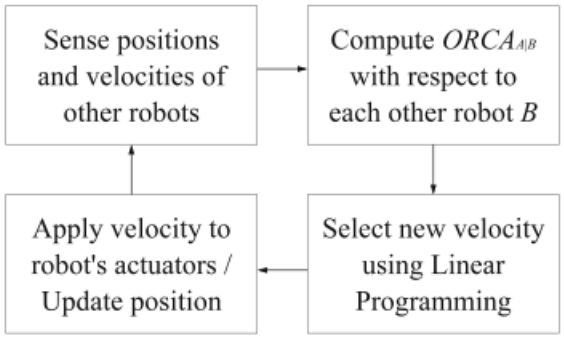


Fig. 3 A schematic overview of the continuous cycle of sensing and acting that is independently executed by each robot.

The overall approach is as follows. Each robot A performs a continuous cycle of sensing and acting with time step Δt. In each iteration, the robot acquires the radius, the current position and the current optimization velocity of the other robots (and of itself). Based on this information, the robot infers the permitted half-plane of velocities $ORCA^{\tau}_{A|B}$ with respect to each other robot B. The set of velocities that are permitted for A with respect to all robots is the intersection of the half-planes of permitted velocities induced by each other robot, and we denote this set $ORCA^{\tau}_{A}$.

$$ORCA^{\tau}_{A} = D(0, v^{A}_{max}) \cap \bigcap ORCA^{tau}_{A|B} \tag{6}$$

(see Fig. 4)

Next, the robot selects a new velocity v^{new}_{A} for itself that is closest to its optimal velocity velocity v^{pref}_{A} amongst all velocities inside the region of permitted velocities:

$$v_{next} = \underset{v \in ORCA^{\tau}_{A|B}}{\arg\min} \left\| v - v^{opt}_{A} \right\| \tag{7}$$

Algorithm 1: Multi body Optimal Reciprocal Collision Avoidance

initialize all robots;
for *each robot* **do**
  (i) Sense and estimate the position and velocities of other robots, which are in the surrounding of robot A and are a velocity obstacle. **if** *robot* $B \in VO^{\tau}_{A|B}$ **then**
    Compute the avoidance hyperspace ORCA for robot B with respect to A;
  else
    skip robot B;
    Use Linear programming to select the feasible and ideal velocity;
    New velocity v_{next} is selected as the closest velocity to our preferred velocity v_{pref} according to Eq. 4.7;
    Apply the new velocity to and update the new position and velocity ;
  end
end

Finally, the robot reaches its new position:

$$p_A^{new} = p_A + v_A^{new} \Delta t \tag{8}$$

and the cycle repeats as in Fig. 3.

The key step in the above procedure is to compute the new velocity v_A^{new} as defined. This can efficiently be done using linear programming, as $ORCA_A^{\tau}$ is a convex region bounded by linear constraints induced by the half-planes of permitted velocities with respect to each of the other robots. The optimization function is the distance to the preferred optimal velocity v_A^{opt} . Even though this is a quadratic optimization function, it does not invalidate the linear programming characteristics, as it has only one local minimum. The algorithm has an expected running time of $O(n)$, where n is the total number of constraints in the linear program (which equals the number of robots in our case). The fact that we include a circular constraint for the maximum speed does not significantly alter the algorithm, and does not affect the running time. The algorithm returns the velocity in $ORCA_A^{\tau}$ that is closest to v_A^{opt} , and reports failure if the linear program is infeasible.

To find the best feasible solution, we bount all the hyperplanes such that they lie within the circle (sphere in case of 3D situations) or radius v_A^{max}. Then we find the common region formed by the intersection of these planes, and the point closest to the desired velocity v_A^{opt} is our next velocity. We can observe that we shall always

have a solution to this optimization problem, simply by observing that by choosing 0 as the next velocity of all robots, we can prevent collision. Therefore, the system of equations is always solvable with a trivial solution in the worst case.

If the optimization velocities for the robots are chosen carefully $ORCA_A^\tau$ will never be empty, and hence there will always be a solution that guarantees that the robots are collision-free for at least τ time. We can increase the efficiency of selecting velocities by not including the constraints of all other robots, but only of those that are "close" by. In fact, robots B that are farther away from robot A than $(v_A^{max} + v_B^{max})\tau$ will never collide with robot A within τ time, so they can safely be left out of the linear program when computing the new velocity for robot A.

We apply the algorithm as given in Algorithm.1. To provide more insight, we shall expand upon the Linear Programming step to select optimal velocities.

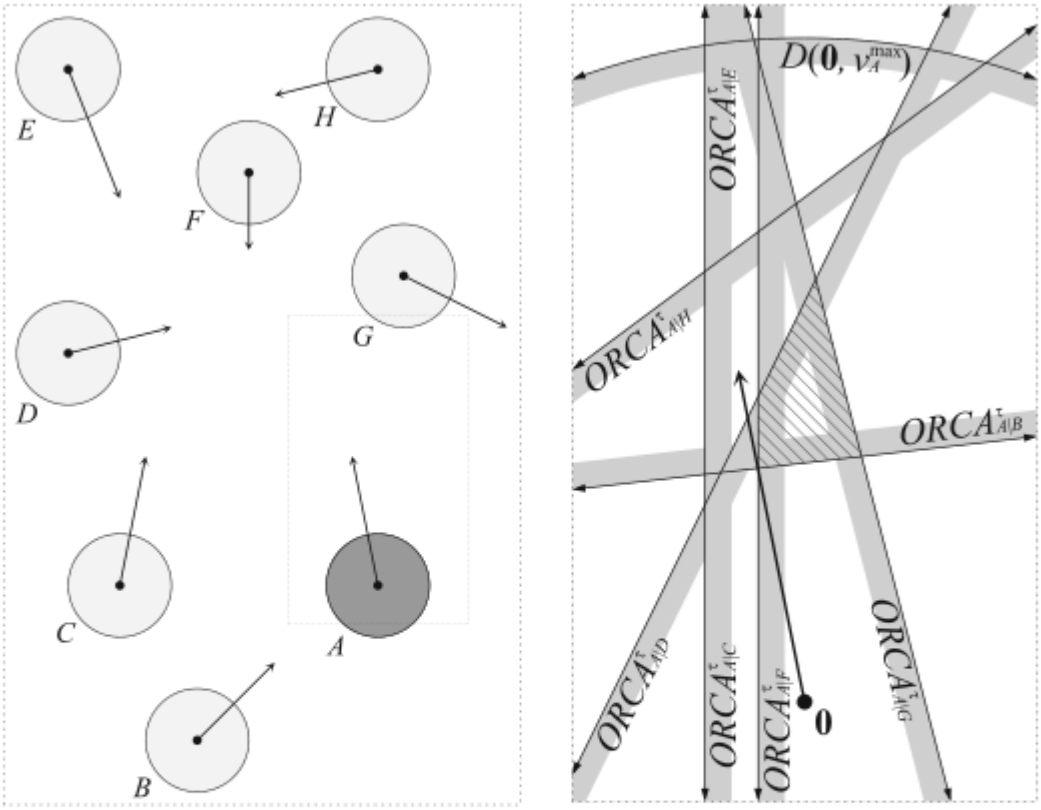


Fig. 4 **(a)** A configuration with eight robots. Their current velocities are shown using arrows. **(b)** The half-planes of permitted velocities for robot A induced by each of the other robots with $\tau = 2$ and with $\mathbf{v}_*^{\text{opt}} = \mathbf{v}_*$ for all robots (i.e. the optimization velocity equals the current velocity). The half-planes of E and C coincide. The dashed region is $ORCA_A^{\tau}$, and contains the velocities for A that are permitted with respect to all other robots. The arrow indicates the current velocity of A.

Let us consider a multi-robot situation as in Fig.4. Fig.4(a) shows the various agents and their current velocity. Let us say that all agents act as velocity obstacles to our controlled agent (robot A), and after solving for ORCA between every other robot and robot A, we get various hyperplanes as in Fig.4(b).

9. OBTAINING OPTIMAL VELOCITIES

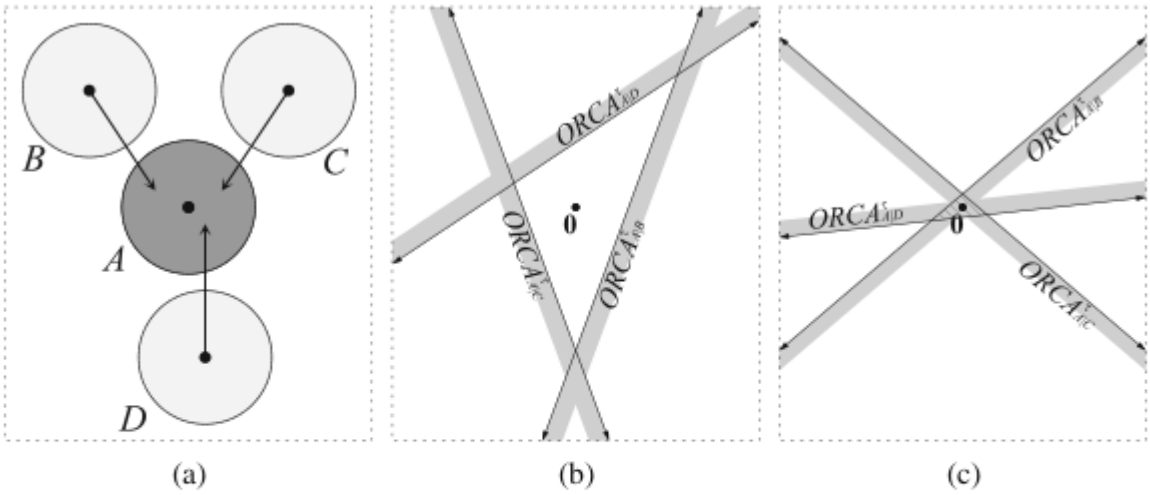


Fig. 5 **(a)** A dense configuration with three robots moving towards robot A. The current velocities of the robots are shown using arrows; robot A has zero velocity. **(b)** The half-planes of permitted velocities for robot A induced by each of the other robots with $\tau = 2$ and $\mathbf{v}_*^{\text{opt}} = \mathbf{v}_*$ for all robots. The region $ORCA_A^\tau$ is empty, so avoiding collisions within τ time cannot be guaranteed. **(c)** The half-planes of permitted velocities for robot A induced by each of the other robots with $\tau = 2$ and $\mathbf{v}_*^{\text{opt}} = \mathbf{0}$ for all robots. The dashed region is $ORCA_A^\tau$.

One issue that we have left open is how to choose the optimal velocity v_A^{opt} for each robot A. Here, we discuss some reasonable possibilities:

• $v_A^{opt} = 0$ for all robots A. If we set the optimization velocity to zero for all robots, it is guaranteed that $ORCA_A^\tau$ is non-empty for all robots A (see Fig. 5(c)). Hence, the linear programming algorithm as described above will find a velocity for all robots that guarantees them to be collision-free for at least τ time. This can be seen as follows. For any other robot B, the point 0 always lies outside the velocity obstacle $VO_{A|B}^\tau$ (for finite τ). Hence the half-plane $ORCA_{A|B}^\tau$ always includes at least velocity 0. In fact, the line delimiting $ORCA_{A|B}^\tau$ is perpendicular to the line connecting the

current positions of A and B. A drawback of setting the optimization velocity to zero is that the behavior of the robots is unconvincing, as they only take into account the current positions of the other robots, and not their current velocities. In densely packed conditions, this may also lead to a global deadlock, as the chosen velocities for the robots converge to zero when the robots are very close to one another.

• $v_A^{opt} = v_A^{pref}$ (i.e. the preferred velocity) for all robots A. The preferred velocity is part of the internal state of the robots, so it cannot be observed by the other robots. Let us, for the sake of the discussion, assume that it is somehow possible to infer the preferred velocity of the other robots, and that the optimization velocity is set to the preferred velocity for all robots. This would work well in low-density conditions, but, as the magnitude of the optimization velocity increases, it is increasingly more likely that the linear program is infeasible. As in most cases the preferred velocity has a constant (large) magnitude, regardless of the density conditions, this would lead to unsafe navigation in even medium density conditions.

• $v_A^{opt} = v_A^{cur}$ (i.e. the current velocity) for all robots A. Setting the optimization to the current velocity is the ideal trade-off between the above two choices, as the current velocity automatically adapts to the situation: it is (very) indicative of the preferred velocity in low-density cases, and is close to zero in dense scenarios. Also, the current velocity can be observed by the other robots. Nevertheless, the linear program may be infeasible in high-density conditions (see Fig. 5(b)).

In this work we used $v_A^{opt} = v_A^{pref}$. To obtain the preferred velocity for robot A v_A^{pref}, we apply a PID algorithm. The PID uses the current positions of the robot and its desired goal positions to generate the required velocity commands. To obtain the current robot positions and velocities, we use and IMU atop each agent. We assume that the data is deterministic (not noisy), and hence use this information directly

10. SIMULAION RESULTS

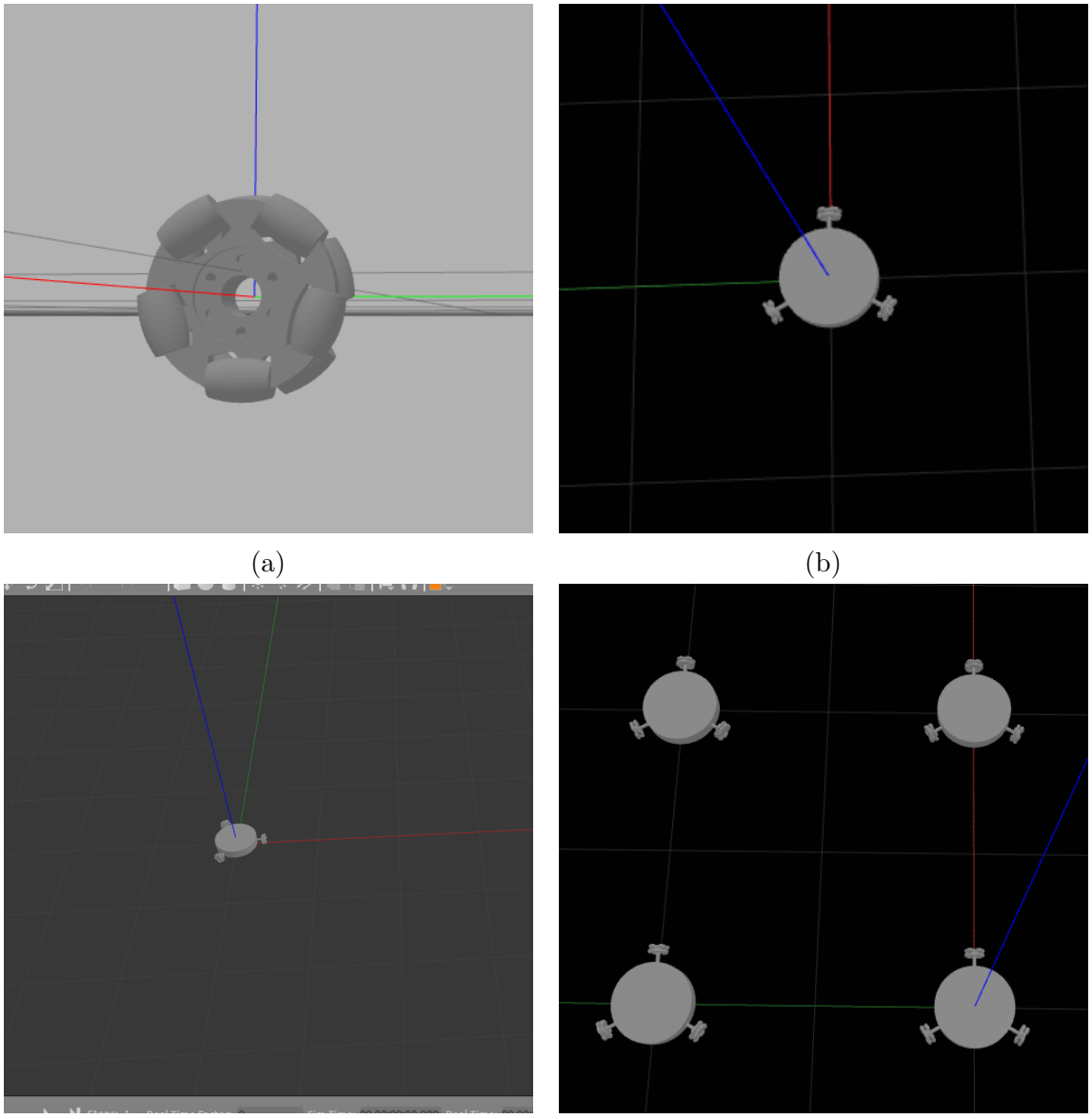
To start with the simulations, we first develop designs for the omni-wheel as in Fig.6a. Each wheel has 5 rollers and radius 0.065625, similar to comercially available 3D printed wheels. The body is then designed to hold three wheels, the radius of the body being 0.42m. We use FreeCAD for the designing phase.

Once designed the robot is assembeld as in Fig.6b and 6c. After this we create spawning scripts to produce multiple robots in the world frame. Each robot, has the same physics and characteristics, but must use different sensorsfor calculating its positions and orientations and publish them as seperate topics. Finally we finish the design process with creating such multiple independent robots that can be made to move independent of the other like in Fig6d

The next phase involves developing the algorithm. As a start, we use Python programming. Each robot publishes it's data to the Robotics Operating System (ROS), from where we collect the data individually. For the matrix multiplications and calculations we use numpy, and for solving the optimization problems we use sympy.

Finally, we show the simulation results of running the Optimal Reciprocal Collision Avoidance algorithm with multiple sets of robots. We run only small scale simulations of upto eight robots due to restrictions in computation capacity. We show three scenarios which highlight how robots smoothly avoid collisions with each other on the local level. In the first, shown in Fig.7, two robots exchange position. When the robots notice that a collision is imminent (i.e. it will happen within τ time), they change velocities to smoothly avoid it. The second scenario shows four robots whose goal is to move to the antipodal position. As Fig.8 shows, the robots smoothly spiral aroundeach other to avoid collisions. We also show the results with six robots in Fig.9

Because each agent makes independent decisions, we are able to efficiently parallelize the simulation by distributing the computations for agents across multiple terminals. All simulations were performed using the Robotics Operating System (ROS) for communication purposes and Gazebo for 3D physics simulations. We use Python programming as a bridge between these two and CAD software to develop the robots



(a) (b)

(c) (d)

Figure 6: The outputs of the designing phase from the 3D simulations using Gazebo

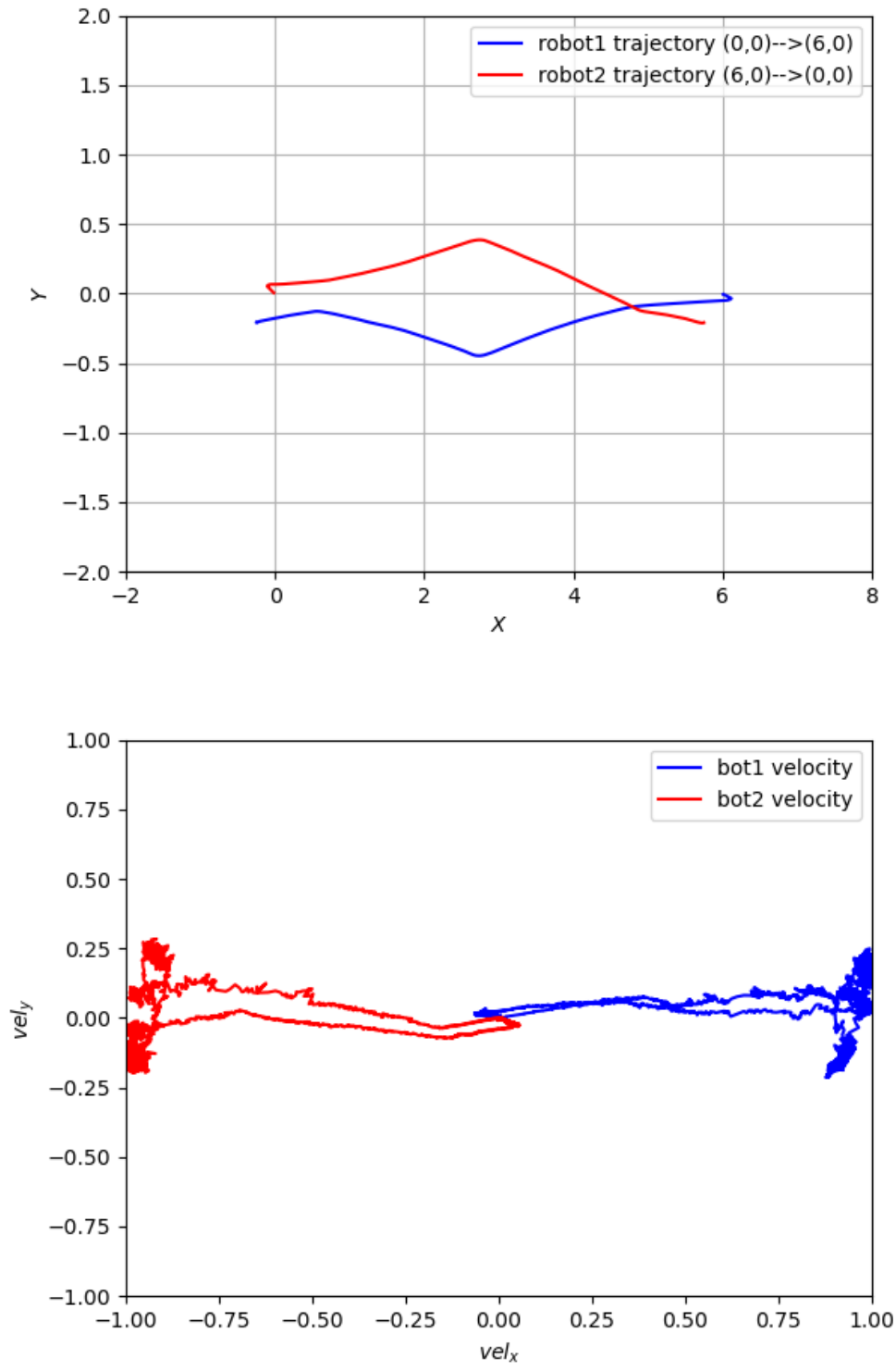


Figure 7: Trajectories and velocity profiles of two robots as they move to each others starting positions

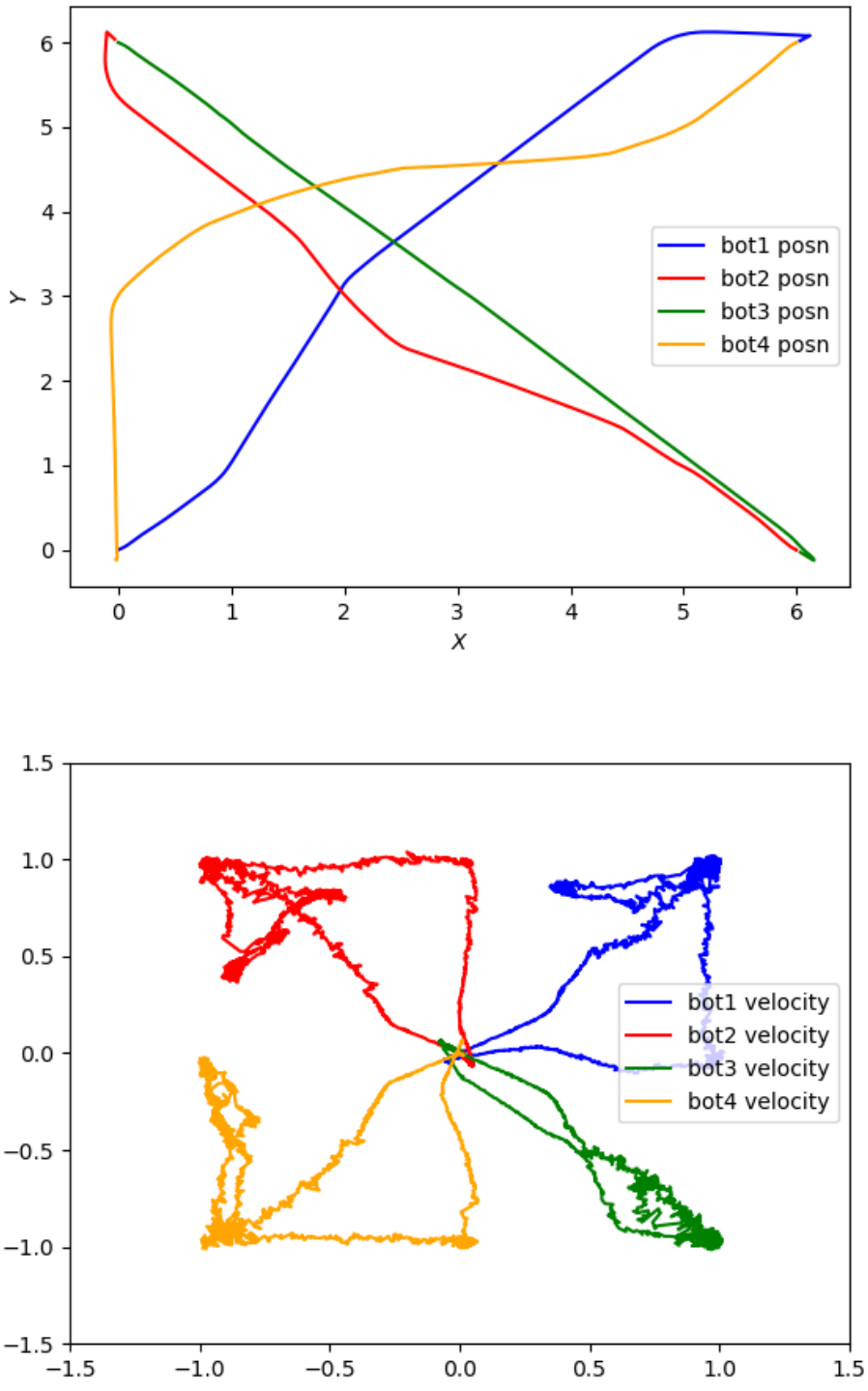


Figure 8: Trajectories and velocities of four robots as they move to opposite coordinates on a square of side length 6m

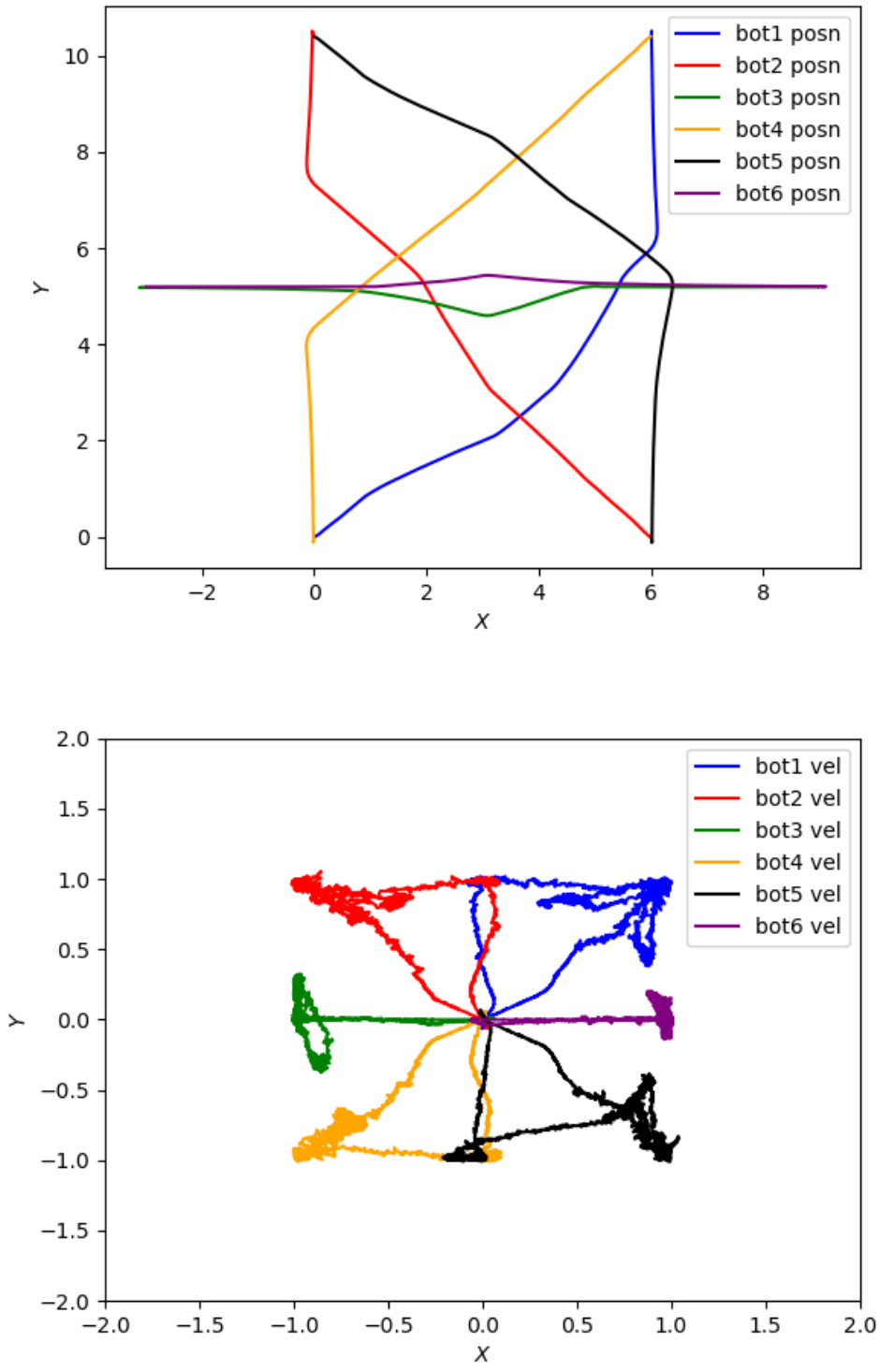


Figure 9: Trajectories and velocities of four robots as they move to opposite coordinates on a hexagon of side length 6m

11. CONCLUSION AND FUTURE WORK

We have discussed an efficient method that provides a sufficient condition for multiple robots to select an action that avoids collisions with other robots, though each acts independently without communication with others. The proposed approach to reciprocal n-body collision avoidance exhibits fast running times and smooth, convincing behavior in our experiments. We have used a simple robot model, in which kinematics and dynamics are ignored. An important extension for future work is to take such constraints into ac- count. We can either do this as a post-processing step, in which the computed new velocity is 'clamped' to what the kinematic and dynamic constraints allow. This would not strictly guarantee avoiding collisions anymore, but it may work well in practice. A more thorough solution would be to take these factors intrinsically into account in the derivation of the permitted velocities for the robots. Also, we have demonstrated results for only 2-D environments. However, all definitions and the algorithm can be extended to 3-D. This may be interesting for applications such as autonomous aerial vehicles, or flocking simulation of birds or fish. Another important direction for future work is to implement the presented framework on real robots and incorporate sensing uncertainty.

Bibliography

[1] Borenstein, J., Koren, Y.: The vector field histogram - fast obstacle avoidance for mobile robots. IEEE Journal of Robotics and Automation 7(3), 278–288 (1991)

[2] Faverjon, B., Tournassoud, P.: A local based approach for path planning of manipulators with a high number of degrees of freedom. In: IEEE Int. Conf. Robot. Autom., pp. 1152–1159 (1987)

[3] Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. 4, 23–33 (1997)

[4] Fraichard, T., Asama, H.: Inevitable collision states - a step towards safer robots? Advanced Robotics 18(10), 1001–1024 (2004)

[5] Kanehiro, F., Lamiraux, F., Kanoun, O., Yoshida, E., Laumond, J.-P.: A local collision avoidance method for non-strictly convex polyhedra. In: Robotics: Science and Systems (2008)

[6] Khatib, O.: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. Int. Journal of Robotics Research 5(1), 90–98 (1986)

[7] Simmons, R.: The curvature-velocity method for local obstacle avoidance. In: IEEE Int.Conf. on Robotics and Automation, pp. 3375–3382 (1996)

[8] Fulgenzi, C., Spalanzani, A., Laugier, C.: Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid. In: IEEE Int. Conf. Robot. Autom., pp.1610–1616 (2007)

[9] Gil de Lamadrid, J.: Avoidance of Obstacles With Unknown Trajectories: Locally Optimal Paths and Periodic Sensor Readings. Int. Journal of Robotics Research 13(6), 496–507 (1994)

[10] Hsu, D., Kindel, R., Latombe, J., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. Int. J. Robot. Res. 21(3), 233–255 (2002)

[11] Martinez-Gomez, L., Fraichard, T.: Collision avoidance in dynamic environments: an ICS-based solution and its comparative evaluation. In: IEEE Int. Conf. on Robotics and Automation (2009)

[12] Petti, S., Fraichard, T.: Safe motion planning in dynamic environments. In: IEEE RSJ Int. Conf. Intell. Robot. Syst., pp. 2210–2215 (2005)

[13] Zucker, M., Kuffner, J., Branicky, M.: Multipartite RRTs for rapid replanning in dynamic environments. In: IEEE Int. Conf. on Robotics and Automation, pp. 1603–1609 (2007)

[14] Kluge, B., Prassler, E.: Reflective navigation: Individual behaviors and group behaviors.In: IEEE Int. Conf. Robot. Autom., pp. 4172–4177 (2004)

[15] van den Berg, J., Lin, M., Manocha, D.: Reciprocal Velocity Obstacles for real-time multi-agent navigation. In: IEEE Int. Conf. on Robotics and Automation, pp. 1928–1935 (2008)

[16] Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using Velocity Obstacles. Int. Journal of Robotics Research 17(7), 760–772 (1998)