Congestion control algorithms for robotic swarms with a common target based on the throughput of the target area
Author version

Yuri Tavares dos Passos[*]
yuri.passos@ufrb.edu.br

Xavier Duquesne
duquesne.xavier.13@gmail.com

Leandro Soriano Marcolino
l.marcolino@lancaster.ac.uk

October 7, 2022

Abstract

When a large number of robots try to reach a common area, congestions happen, causing severe delays. To minimise congestion in a robotic swarm system, traffic control algorithms must be employed in a decentralised manner. Based on strategies aimed to maximise the throughput of the common target area, we developed two novel algorithms for robots using artificial potential fields for obstacle avoidance and navigation. One algorithm is inspired by creating a queue to get to the target area (Single Queue Former – SQF), while the other makes the robots touch the boundary of the circular area by using vector fields (Touch and Run Vector Fields – TRVF). We performed simulation experiments to show that the proposed algorithms are bounded by the throughput of their inspired theoretical strategies and compare the two novel algorithms with state-of-art algorithms for the same problem (PCC, EE and PCC-EE). The SQF algorithm significantly outperforms all other algorithms for a large number of robots or when the circular target region radius is small. TRVF, on the other hand, is better than SQF only for a limited number of robots and outperforms only PCC for numerous robots. However, it allows us to analyse the potential impacts on the throughput when transferring an idea from a theoretical strategy to a concrete algorithm that considers changing linear speeds and distances between robots.

1 Introduction

To be more cost-effective, an increasing number of problems are tackled using many simple robots rather than a few complex robots (Dudek et al., 1993). Systems formed by a large number of robots are called robotic swarms. They are composed of robots that can only interact with direct neighbours and follow simple rules (Beni and Wang, 1993). Counter-intuitively, some combinations of simple rules can create complex global behaviours (Mataric, 1995; Panait, 2003). Hence, some of the robotic swarm research is based on the creation of complex behaviours based on simple rules (Navarro and Matía, 2013; Garnier et al., 2007). Therefore, careful design can enable swarms of simple robots to perform complex tasks with robustness instead of using complex and expensive robots.

However, robotic swarms are often slowed down by conflicts created by the trajectories of the robots. For example, traffic congestion appears when several robots try to reach the same area simultaneously (Treuille et al., 2006; Graciano Santos and Chaimowicz, 2011; Yan et al., 2013; Grossman, 1988). Congestion happens when several robots need to reach a common area, such as in waypoint navigation (Marcolino and Chaimowicz, 2008; Duarte et al., 2016) and in foraging (Ducatelle et al., 2011; Fujisawa et al., 2019). In the current state of the art, congestion problems in robotic swarms are mainly solved by collision avoidance in a decentralised manner because it enables better scalability of the algorithms Batra et al. (2022); Majcherczyk et al. (2018); Borrmann et al. (2015). Despite that, just avoiding collisions does not necessarily lead to a good performance in the common target problem. For example, Marcolino et al. (2017) showed that using only the decentralised collision avoidance algorithm ORCA (Optimal Reciprocal Collision Avoidance) (van den Berg et al., 2011) creates local minima around the common target region. They present experimental solutions for the congestion problem that arises when the swarm shares the same goal, but we found that these algorithms may completely fail in small target regions, with an area measuring less than five times the area of the robots. Moreover, evolution-based algorithms may adapt to congestions after thousands of iterations Fujisawa et al. (2019), but we focus on algorithms directly designed to handle congestion without requiring long iterations for training or adaptation.

Additionally, as congestion in robotic swarms grows with the number of individuals, any analysis of the efficiency of algorithms to reduce congestion needs to incorporate the increase in the number of robots. In

[*]Corresponding author

this regard, we introduced in our concurrent work (Passos et al., 2022) a viable metric for analysing large-scale swarm robotics systems: the throughput of the common target area. We defined it as the inverse of the average time between arrivals at the target area. Furthermore, in that work, we have developed theoretical strategies to maximise the throughput in scenarios where the robots have constant maximum linear speed and fixed minimum distance between each other. Although these scenarios are useful for ease of mathematical analysis, they are not practical in real applications as inter-robot distances and speeds vary, but the presented strategies help design new algorithms.

Therefore, the main contribution of this paper is that, based on these theoretical strategies, we present two novel algorithms for handling congestion in more realistic settings: Single Queue Former (SQF) and Touch and Run Vector Fields (TRVF). We evaluate our algorithms by realistic Stage (Gerkey et al., 2003) simulations with holonomic and non-holonomic robots. From these simulations, we conclude that SQF outperforms the state-of-the-art and can handle a small target region. In that situation, previous approaches completely failed in our simulations. Moreover, we learn that TRVF helps us to understand the potential impacts of the variation in the linear speed and distance between the robots when translating an idea from theoretical strategies to concrete algorithms.

This paper is organised as follows. In the next section, we discuss some related work. We describe the two algorithms in Section 3. Section 4 presents the experiments and compares the two new algorithms with the state-of-art ones. Finally, we summarise our results and make final remarks in Section 5.

2 Related Work

We found related work for traffic control for multiple robot systems, but they must follow lanes, predetermined paths or leaders. Also, there are papers on multi-agent systems for autonomous cars, but with global planning, without time analysis per number of robots or having spatial limitations. Until now, surveys in robotic swarms do not treat our problem. In the literature on multiple robot systems, we could find that collision avoidance is closely related to our problem, but works on this topic do not measure the efficiency concerning a common target. Also, heed that collision avoidance alone is not enough to solve our problem. Finally, we treated this problem mathematically in a previous work whose described strategies inspired the algorithms presented here. These are discussed in detail as follows.

Traffic control for multiple robots systems Robotic traffic control has been studied for a long time (Grossman, 1988; Kato et al., 1992; Caloud et al., 1990), but assuming robots navigating in delimited lanes and coordination is necessary only at the intersections. Other more recent works still focus on alleviating congestions in delimited lanes and circuits (Masud et al., 2020; Hoshino and Seki, 2013; Hoshino, 2011; Viswanath and Madhava Krishna, 2009). For instance, an algorithm based on Petri nets has been introduced to avoid deadlocks at intersections (Zhou et al., 2017), where each robot follows a pre-determined closed path, but it works only for robots on lanes and does not show performance in terms of simulation time nor throughput. Saska et al. (2020) developed a control algorithm for motion planning of formations for unmanned aerial vehicles in environments with narrow passages. The robots in this system follow a leader and maintain proximity while avoiding obstacles in a constrained space. Yoshimoto et al. (2018) describe a decentralised algorithm to maintain proximity between robots. Although their work is designed for a robotic swarm, it is also leader-based.

Multi-agent systems for autonomous cars Similarly, there are many relevant works in the multi-agent systems literature, but assuming autonomous cars that navigate following lanes or roads, and coordination is needed at the junctions (Carlino et al., 2013; Sharon and Stone, 2017). Other lines of work try to optimise trajectories across edges of complex traffic networks, for example, by global planning and using incentives or tolls for self-interested agents (Sharon et al., 2017, 2018). Cenedese et al. (2016) use the Kuramoto model to coordinate multiple vehicles towards a target while avoiding collisions and keeping them next to each other, but they do not analyse the time to get to the target and exit from it as the number of vehicles grows. Ma et al. (2020) describe a reinforcement learning algorithm for multi-agent planning for a swarm of vehicles to go to objectives distributed over space, but it does not perform well when the targets cannot be well divided into regions over the environment.

No similar study in robotic swarm survey The problem of alleviating congestion when a swarm of robots has a common target has not yet been well studied. Thorough surveys about swarm robotics (Sahin, 2005; Sahin et al., 2008; Barca and Sekercioglu, 2013; Brambilla et al., 2013; Bayındır, 2016; Chung et al., 2018) do not discuss these situations. Additionally, a recent survey on collision avoidance (Hoy et al., 2015) provides insights into multiple vehicle navigation, but they do not discuss this issue in multi-robot navigation.

Works in collision avoidance Collision avoidance is also an important related topic of study in robotics. We can find in the literature algorithms that let robots move efficiently in environments with a lot of obstacles, but they do not measure the efficiency of the algorithm in the common target problem (Hewawasam et al., 2022). As an example, Ferrera et al. (2017) have proposed a decentralised algorithm where a set of local reactive rules are followed to avoid collisions. We also note that their work shows an extensive list of benchmarks and metrics for robotics problems, but performance in common target situations is not included. Evolution-based algorithms may learn traffic rules for collision avoidance after thousands of iterations Fujisawa et al. (2019), but they were not yet explored for the common target problem. Additionally, in this paper, we focus on algorithms directly designed to handle congestion based on our theoretical work Passos et al. (2022) without requiring long iterations of training or optimisation.

Inadequacy of employing only collision avoidance However, only avoiding collisions does not necessarily lead to a good performance in the common target problem. For instance, Marcolino et al. (2017) showed that the ORCA algorithm (van den Berg et al., 2011) reaches an equilibrium where the robots cannot go to the target. They proposed three algorithms using artificial potential fields and offered experimental solutions for the target congestion problem. However, we found that their proposed algorithms face issues as the target area gets smaller.

Insight from a theoretical work Inspired by theoretical developments in our concurrent work (Passos et al., 2022), we propose new algorithms that outperform these previous approaches. That is, we proposed to analyse the arrival rate in the target area as the time tends to infinity as an alternative approach to analysing congestion in swarms with a common target area. This measure the advantages of being finite for any closed target region of any shape as the number of robots grows, as opposed to the number of robots per time of arrival at the target area. Assuming a circular target area and robots with constant maximum linear speed and fixed minimum distance from each other, we developed theoretical strategies for entering that area and calculated their theoretical throughput for a given time and their asymptotic throughput. The presented theoretical strategies were based on forming a corridor towards the target area or making multiple curved trajectories towards the boundary of the target area. However, they assumed constant speeds and fixed minimum distances between robots, thus not yet directly applicable in practice. Hence, based on the corridor strategy, we propose here the Single Queue Former algorithm, and, inspired by the multiple curved trajectories, we introduce the Touch and Run Vector Fields algorithm.

3 Proposed algorithms

In this section, we consider the scenario where a large number of robots must reach a common target. After reaching the target, each robot moves towards another destination which may or may not be common among the robots. We assume the target is defined by a circular area of radius s. A robot reaches the target if its centre of mass is at a distance below or equal to the radius s from the centre of the target. We assume that there is no minimum amount of time to stay at the target.

We present two algorithms based on the results in (Passos et al., 2022). In it, the *throughput* of the target area is used to measure performance, which is defined as the inverse of the average time between arrivals at the target. Differently to that work, which intended to find the maximum throughput for the strategies by using constant maximum linear speed and fixed minimum distance between the robots, in real applications, robots move at variable speeds and do not always keep the same distance between others, although they may try to maintain some distance to avoid bumping. In addition to that, the space around the target region is finite and may have other obstacles.

Having this in mind, we devised two algorithms: the Single Queue Former and the Touch and Run Vector Fields. These algorithms are only applied inside a circle of radius $D > s$ around the target. Also, we assume that if the robots have two or more targets, they are apart from each other at least $2D$.

We hereafter use geometric notation as follows. $\overrightarrow{AB}$ and $\overline{AB}$ represents a ray starting at A and passing through B and a segment from A to B, respectively. $|\overline{AB}|$ is the size of $\overline{AB}$. If a two-dimensional point is represented by a vector P_1, its x- and y-coordinates are denoted by $P_{1,x}$ and $P_{1,y}$, respectively. $\triangle ABC$ expresses the triangle formed by the points A, B and C. $\widehat{AOB}$ means an angle with vertex O, one ray passing through point A and another through B. All angles are measured in radians in this paper.

3.1 Single Queue Former

Based on the corridor strategy from (Passos et al., 2022), we propose an algorithm, named Single Queue Former (SQF), to improve throughput in case of target congestion whose aim is to form a single queue that goes towards

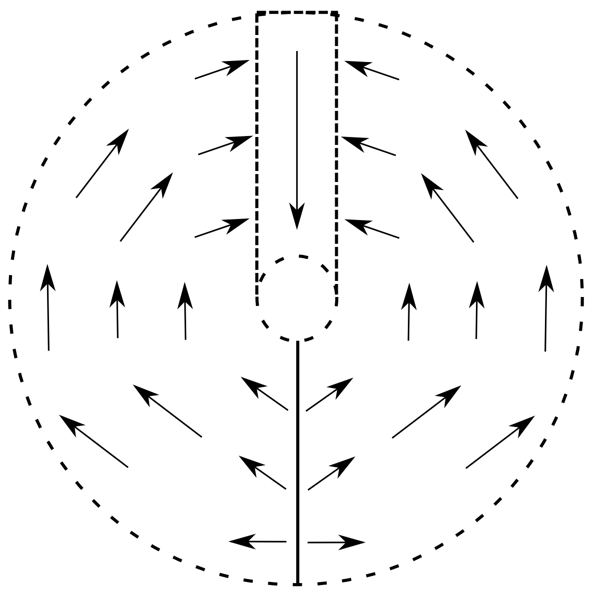


Figure 1: Rotational force field to reach the target with Single Queue Former algorithm.

the target. The queue is formed inside a rectangular corridor of width equal to the circular target diameter and a given fixed length. The robots should only enter the target region by this queue.

Specifically, this algorithm is based on the compact lanes and hexagonal packing strategies findings from (Passos et al., 2022). However, we do not enforce the hexagonal formation here to avoid robots losing time forming it, although the hexagon packing is the configuration of the robots that maximises the throughput in an ideal scenario where the robots maintain the same distance and keep constant linear speed (Passos et al., 2022). As it will be shown in Section 4, the throughput for this algorithm is bounded by the hexagonal packing throughput for the mean distance and mean linear speed achieved by the robots.

We use artificial potential fields (Siegwart and Nourbakhsh, 2004) to apply forces on the robots to form the queue and exit the target area efficiently. Let s be the radius of the current target. The corridor queue has a width $2s$ and starts from the current target centre, denoted by o. Let the length of the corridor be the same value used for the radius of the circular working area of the algorithms around the target, denoted D. Without any loss of generality, the corridor is located between the point (o_x, o_y) and the point $t = (o_x, o_y + D)$. Robots that are going to enter the target area are submitted to a rotational field whose centre is the position of the target centre. When they reach the corridor, they are submitted to an attractive force towards the target. An illustration is shown in Figure 1.

Therefore, robots outside the corridor are submitted to a force according to the following equations. For the right-hand side of the circle, an anti-clockwise rotational field:

$$F_x = -K_{SQF}\frac{p_y - o_y}{\|p - o\|}, \quad F_y = +K_{SQF}\frac{p_x - o_x}{\|p - o\|}. \tag{1}$$

For the left-hand side of the circle, a clockwise rotational field:

$$F_x = +K_{SQF}\frac{p_y - o_y}{\|p - o\|}, \quad F_y = -K_{SQF}\frac{p_x - o_x}{\|p - o\|}, \tag{2}$$

where F_x and F_y are the two components of the force applied, $p = (p_x, p_y)$ is the position of the robot, $o = (o_x, o_y)$ is the target centre, and K_{SQF} is a constant for setting the force magnitude.

Additionally, we noticed in previous algorithms (Marcolino et al., 2017) that the robots reaching the target tend to stop on the target and orient themselves towards their next goal. This increases the time during which the target is occupied and thus decreases the throughput. To solve this problem, robots exiting nearby the target are constrained by another rotational field. This field also is only applied inside a circle with radius D around the target (Figure 2). The aim of this field is to be aligned with the corridor orientation at the target and progressively allow robots to change directions. This field also depends on the new target position, not only on the robot's location, because if some robots reach the target area on a side but have to go to the opposite side when they leave the circle, it will cause congestion next to the exit area. For a robot with a new target located on the right-hand side of the previous target centre, we apply an anti-clockwise rotational field:

$$F_x = -K_{SQF}\frac{p_y - Q_y}{\|p - Q\|}, \quad F_y = +K_{SQF}\frac{p_x - Q_x}{\|p - Q\|}. \tag{3}$$

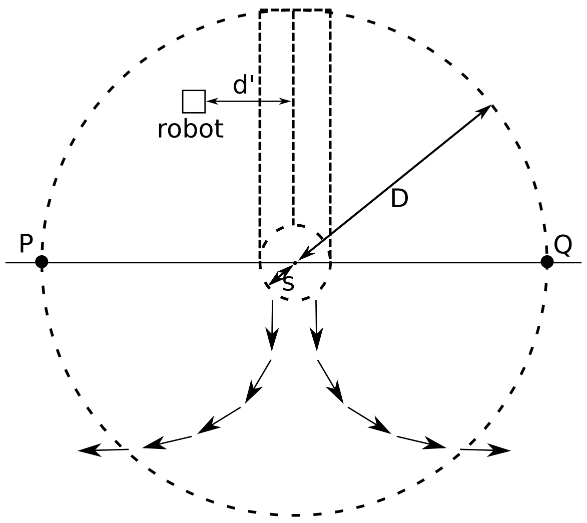


Figure 2: Rotational force field to leave the target in the SQF algorithm, and the distance d' from the central line of the corridor to the robot for varying its influence radius.

For a robot with a new target located on the left-hand side of the previous target centre, a clockwise rotational field is applied:

$$F_x = +K_{SQF}\frac{p_y - P_y}{\|p - P\|}, \quad F_y = -K_{SQF}\frac{p_x - P_x}{\|p - P\|}, \tag{4}$$

where $Q = (Q_x, Q_y) = (o_x + D, o_y)$ is a point on the right of the target, and $P = (P_x, P_y) = (o_x - D, o_y)$ is a point on the left of the target.

This outgoing rotational field enables robots to stay on their course after they reach the target and gradually rotate towards their next goal. The orientation of the field depends on the position of the robot relative to the corridor. If the robot is on the left of the corridor, the robot follows the rotational field going to the left with centre P. Otherwise, if it is on the right of the corridor, it follows the rotational field going to the right with centre Q.

Finally, let the influence radius be the maximum distance a robot considers anything sensed as an obstacle to avoid from its mass centre. To enable the robots to deal with small target sizes, we use two different constants for the influence radius between the robots when calculating repulsive forces, I_d and I_{min}, with $I_{min} < I_d$: I_d is the default and maximum radius for influence, and I_{min} is the minimum allowed.

For robots inside the corridor or exiting the target region, the influence radius is set to I_{min}. Now, consider a robot is outside the corridor, but above the $\overline{PQ}$ line in Figure 2. Let d' be the distance between the robot and the central line of the corridor (that is, from the position of the robot to the closest point in the vertical axis of Figure 2, starting from the target centre). Its influence radius I varies in relation to d' and is set to $I = I_{min} + d'$ only for $0 \leq d' \leq I_d - I_{min}$. This range for d' guarantees that $I_{min} \leq I \leq I_d$. For the rest of the robots the influence radius is I_d.

Algorithm 1 presents the SQF algorithm. Robots have the states *going_to_target*, *leaving_target* and *going-_to_corridor*, which respectively means the robot is going straightly to the target region, it is leaving it, and it is going to the corridor. The robots begin at *going_to_target* state. Figure 3 shows the state machine and its transitions, assuming that the robot may have two or more targets located at $o_j = (o_{j,x}, o_{j,y})$, the circular target region has radius s_j, and j is its current target index. Note that this algorithm is only executed when the robot is at a distance at most D from the target centre.

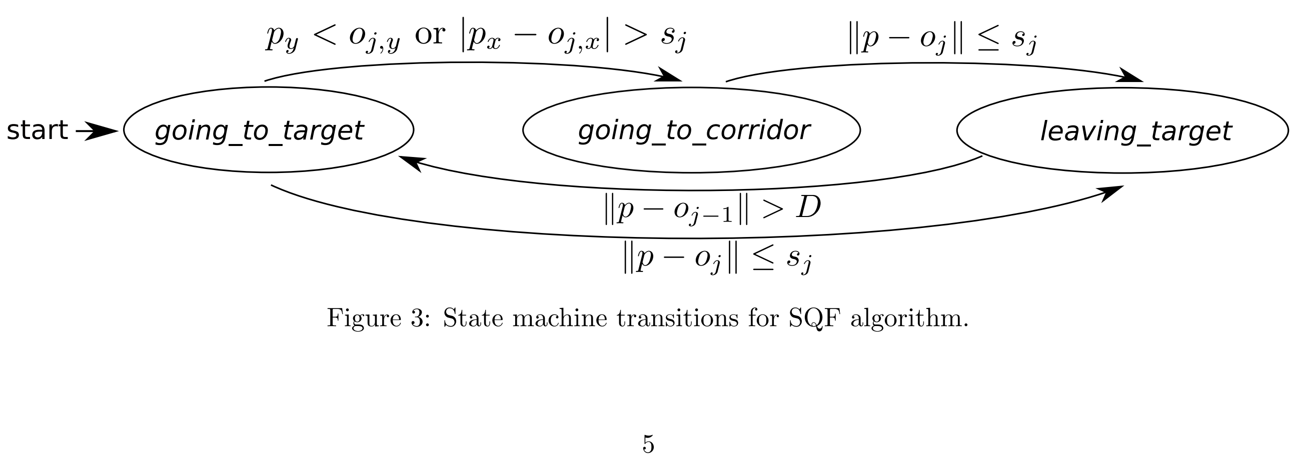


Figure 3: State machine transitions for SQF algorithm.

```
Input  : $K_{SQF}$ : force magnitude;
         $D$ : the length of the corridor;
         $o_1 = (o_{1,x}, o_{1,y}), \ldots, o_n = (o_{n,x}, o_{n,y})$ : a list of $n \geq 2$ circular target region centres;
         $s_1, \ldots, s_n$ a list of $n$ circular target region radii;
         $j$ : the current target index;
         $I_{min}, I_d$ : minimum and default influence radius of the robot.
Output: $F = (F_x, F_y)$ : force vector;
         $I$ : influence of the robot for repulsive force calculation.
1  Get position of the robot $p = (p_x, p_y)$, and let $o = o_j = (o_x, o_y)$ and $s = s_j$;
2  if $\|p - o\| \leq s$ then
3      state ← leaving_target;
4      Increment $j$, then let $o = o_j = (o_x, o_y)$ and $s = s_j$;
5  if state ≠ leaving_target then
6      if $\|p - o\| \leq D$ and ($p_y < o_y$ or $|p_x - o_x| > s$) then
7          state ← going_to_corridor;
8          Set $F$ according to (1) and (2);
9      else
10         state ← going_to_target;
11         F ← $K_{SQF} \frac{o-p}{\|o-p\|}$;
12 else
13     if $\|p - o_{j-1}\| \leq D$ then
14         Set $F$ according to (3) and (4);
15     else
16         state ← going_to_target;
17         F ← $K_{SQF} \frac{o-p}{\|o-p\|}$;
18 if state = going_to_target or state = leaving_target then
19     $I \leftarrow I_{min}$;
20 else if state = going_to_corridor and $p_y > o_y$ and $|p_x - o_x| < I_d - I_{min}$ then
21     $I \leftarrow I_{min} + |p_x - o_x|$;
22 else
23     $I \leftarrow I_d$;
24 return $(F, I)$;
```

Algorithm 1: SQF algorithm, executed at every update in the position of the robot.

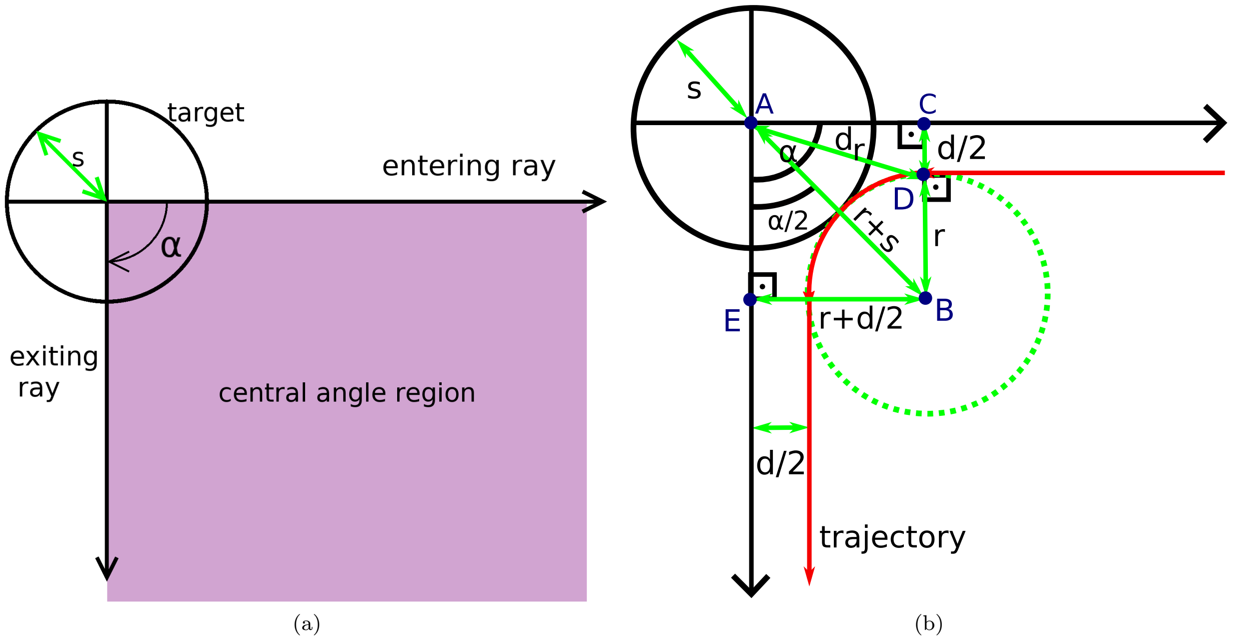


Figure 4: Illustration of the touch and run strategy (from (Passos et al., 2022)). (a) Central angle region and its exiting and entering rays, defined by the angle α. (b) The trajectory of a robot next to the target, in red. Here we have the relationship between the target area radius (s), the minimum safety distance between the robots (d), the turning radius (r), the central region angle (α) and the distance from the target centre for a robot to begin turning (d_r). The green dashed circle represents the whole turning circle.

3.2 Touch and Run Vector Fields

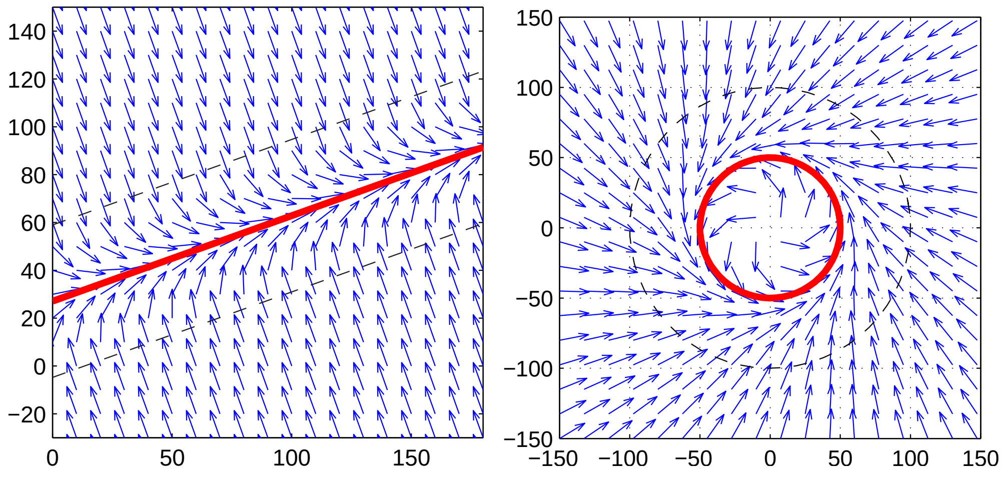
Since a robot should spend as little time as possible near the target, in (Passos et al., 2022), we imagined a simple scenario where robots travel in predefined curved lanes and tangent to the target area, where they spend minimum time on the target. To avoid collisions with other robots, the trajectory of a robot nearby the target is circular, and the distance between each robot must be at least d at any part of the trajectory. Hence, no lane crosses another, and each lane occupies a region defined by an angle in the target area – the central region angle – that we denote by α, shown in Figure 4 (a). Figure 4 (b) shows the trajectory of a robot towards the target region following that *touch and run* strategy. The robot first follows the boundary of the central angle region – that is, the entering ray – at a distance of $d/2$. Then, it arrives at a distance of s of the target centre using a circular trajectory with a turning radius r. As the trajectory is tangent to the target shape, it is close enough to consider that the robot reached the target region. Finally, the robot leaves the target by following the second boundary of the central angle region – that is, the exiting ray – at a distance of $d/2$.

Based on this strategy, we devise an algorithm to mimic the curved trajectories around the target region using vector fields, named Touch and Run Vector Fields (TRVF). The vector fields are only used inside a circle with radius D around the target centre and apply forces on the robot to make it follow a circular trajectory similar to the one described in the touch and run strategy.

As a way to do that, we adapted and combined the vector fields proposed by Nelson et al. (2006) for straight line and orbit following trajectories. We use their straight line following vector fields whenever the robot has to follow a straight trajectory and an anti-clockwise orbit following vector field for curved trajectories. We will first explain our modified functions for these trajectories, and afterwards, we will introduce the TRVF algorithm.

Our adaptation of their work concerns reducing the orbit following to a circular sector, instead of the full circle, and adding an attractive force towards a fixed point when the robots are performing the orbit following next to the target region. The reduction to a circular sector is due to the curved trajectory next to the target having $\alpha < 2\pi$. We add attractive forces towards a fixed point to the orbit following vector field intending to avoid the repulsive forces caused by nearby robots to push a robot far away from that fixed point.

Figure 5 shows the straight line and orbit following vector fields proposed by Nelson et al. (2006). Algorithm 2 presents our adapted function that computes the straight line following vector field, its parameters and values returned. It assumes a fixed segment to follow with initial and final waypoints w_i and w_f, respectively. Based on the current position and orientation of the robot, p and ξ, respectively, a force vector with magnitude K_{TRVF} is computed. The orientation of the force vector changes gradually as the robot approaches the segment. These changes are calculated using the maximum linear speed of the robot v, and its rate of change depends on the constants k_s and K_r.



(a) (b)

Figure 5: Straight line and orbit following vector fields (from (Nelson et al., 2006)).

```
1  Function straightPathFollowing ($K_{TRVF}, p, \xi, w_i, w_f, I_d, k_s, K_r, v$)
   Input  : $K_{TRVF}$ : force magnitude;
            $p = (p_x, p_y), \xi$ : current position and orientation of the robot;
            $w_i = (w_{i,x}, w_{i,y}), w_f = (w_{f,x}, w_{f,y})$ : initial and final waypoints for the segment to follow;
            $I_d$ : influence radius of the robot;
            $k_s > 1$: constant for exponentiation in the vector field calculation;
            $K_r$ : constant for proportional angular speed controller;
            $v$: maximum linear speed.
   Output: $F = (F_x, F_y)$ : force vector;
            $t$ : indicates the position along the path (0 when the position of the robot is on the line
            perpendicular to $\overline{w_i w_f}$ on the initial waypoint and 1 when that happens for the final
            waypoint).
2  $w_{fi} = w_f - w_i$;
3  $t \leftarrow \frac{(p-w_i)\cdot w_{fi}}{\|w_{fi}\|^2}$;
4  if $t \geq 1$ then return $((0,0), t)$;
5  $\xi_f \leftarrow \text{atan2}(w_{fi,y}, w_{fi,x})$;
6  $\epsilon \leftarrow \|p - w_i - t w_{fi}\|$;
7  $\rho \leftarrow sign(w_{fi} \times (p - w_i))$ ;
8  $\tau \leftarrow I_d/5$;
9  $\xi_e \leftarrow \pi/2$;
10 if $\epsilon > \tau$ then
11     $\xi_c \leftarrow \xi_f - \rho\xi_e$;
12 else
13     $\epsilon \leftarrow \rho\epsilon$;
14     $P_1 \leftarrow \left(\frac{\epsilon}{\tau}\right)^{k_s}$; if $P_1$ is NaN then $P_1 \leftarrow 0$;
15     $P_2 \leftarrow \epsilon^{k_s - 1}$; if $P_2$ is NaN then $P_2 \leftarrow 0$;
16     $\xi_c \leftarrow \xi_f - \xi_e P_1 - \left(\frac{k_s \xi_e v}{K_r \tau^{k_s}}\right) P_2 \sin(\xi)$;
17 $F \leftarrow K_{TRVF}(\cos(\xi_c), \sin(\xi_c))$;
18 return $(F, t)$;
```

Algorithm 2: Straight-line vector field algorithm adapted from Nelson et al. (2006).

In the original algorithm, ξ_e is the entry heading angle, and τ is the distance perpendicular to the line from which the vector field makes the heading angle begin to change (in Figure 5 (a), it is the distance from the

```
1  Function orbitPathFollowing ($K_{TRVF}, c, R, p, \xi, w_f, k_o, K_r, v$)
   Input  : $K_{TRVF}$ : force magnitude;
            $c, R$ : centre and radius for orbit following;
            $p = (p_x, p_y), \xi$ : current position and orientation of the robot;
            $w_f = (w_{f,x}, w_{f,y})$ : final waypoint;
            $k_o > 1$: constant for exponentiation in the vector field calculation;
            $K_r$ : constant for proportional angular speed controller;
            $v$: maximum linear speed.
   Output: $F = (F_x, F_y)$ : force vector;
            $t$ : indicates the position along the path (positive when the robot did not cross $\overrightarrow{cw_f}$).
2  $q \leftarrow p - c$;
3  $t \leftarrow q \times (w_f - c)$;
4  if $t \leq 0$ then return $((0,0), t)$;
5  $\gamma \leftarrow \text{atan2}(q_x, q_y)$;
6  if $\|q\| > 2R$ then
7      $\xi_c \leftarrow \gamma - \frac{5\pi}{6} + \frac{v}{\|q\|}\sin(\xi - \gamma)$;
8  else
9      $P_1 \leftarrow \left(\frac{\|q\|-R}{R}\right)^{k_o}$; if $P_1$ is NaN then $P_1 \leftarrow 0$;
10     $P_2 \leftarrow (\|q\| - R)^{k_o - 1}$; if $P_2$ is NaN then $P_2 \leftarrow 0$;
11     $\xi_c \leftarrow \gamma - \frac{\pi}{2} - \frac{\pi}{3}P_1 - \frac{v}{K_r\|q\|}\sin(\xi - \gamma) - \frac{k_o v \pi}{3R^{k_o}K_r}P_2\cos(\xi - \gamma)$;
12 $F \leftarrow K_{TRVF}(\cos(\pi/2 - \xi_c), \sin(\pi/2 - \xi_c))$;
13 return $(F, t)$;
```

Algorithm 3: Anti-clockwise orbit following vector field algorithm adapted from Nelson et al. (2006).

dashed line to the solid line). In our adaptation, we set $\tau = I_d/5$ and $\xi_e = \pi/2$, for a given default influence radius of the robot I_d. Additionally, we only compute the force vector if the robot did not reach the point w_f or if the robot is in a position on the plane perpendicular to $\overline{w_i w_f}$ limited by its endpoints, otherwise, we return force vector zero (line 4). At line 7, *sign* returns -1 for negative numbers and 1, otherwise. Here $\times$ stands for the cross-product of two vectors. We also add a verification for not a number (NaN) in the exponentiations and replace the result for zero as it is a common source of numerical errors (lines 14 and 15). We are assuming a proportional angular speed controller such that if the robot has orientation ξ and needs to turn to orientation given by ξ_c, then it uses angular speed $K_r(\xi_c - \xi)$ for a chosen $K_r > 0$.

Algorithm 3 shows our modified function that computes the orbit following vector field, its parameters and values returned. Let the orbit to follow be centred at a point $c = (c_x, c_y)$ and have radius R. As in the straight line following algorithm, given the current position and orientation of the robot, p and ξ, respectively, a force vector with magnitude K_{TRVF} is calculated. In this work, we intend to use the orbit following algorithm to perform curves, so a waypoint w_f must be given to indicate where the orbit following must stop. As in the previous algorithm, the orientation of the force vector changes gradually as the robot approaches the orbit, and these changes are computed using the maximum linear speed of the robot v, and its rate of change depends on the constants k_o and K_r.

This algorithm only generates a force vector different from zero if the robot did not cross $\overrightarrow{cw_f}$ for a given waypoint w_f. We verify if it crossed by the positive sign of the cross product between the vector $q = p - c$ (that is, the vector from the centre of the orbit to the position p of the robot) and the vector $w_f - c$. As the orbit is anti-clockwise, the orbit following vector field makes the cross product $q \times (w_f - c)$ decrease with time, being zero when q has the same orientation as $w_f - c$ and negative when the orientation of q is greater than the orientation of $w_f - c$. We return force vector zero when the robot is in a position beyond $\overrightarrow{cw_f}$ or intersects this ray (line 4).

At line 5, γ is the heading from the centre of the orbit to the position of the robot. Nelson et al. (2006) use clockwise orientation and they assume null orientation when the robot is at the y-axis, thus we are using $\gamma = \text{atan2}(q_x, q_y)$ instead of $\text{atan2}(q_y, q_x)$ as in the original algorithm. Due to this, at line 12 we use $\pi/2 - \xi_c$ for computing the result using the usual orientation, that is, anti-clockwise with null orientation at the x-axis, as justified by Figure 6.

As presented in the touch and run strategy of Passos et al. (2022), the TRVF algorithm uses K lanes. Each robot calculates four waypoints to follow in the lane located on the right side of the robot, assuming it is facing

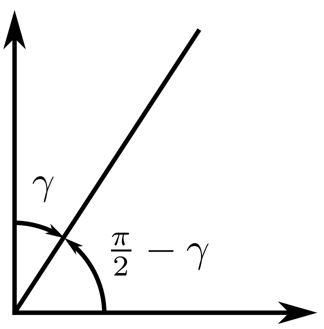


Figure 6: If a line has angle γ with the y-axis assuming clockwise orientation, then this same line has angle $\pi/2 - \gamma$ with the x-axis using anti-clockwise orientation, as the y-axis makes an angle of $\pi/2$ with the x-axis.

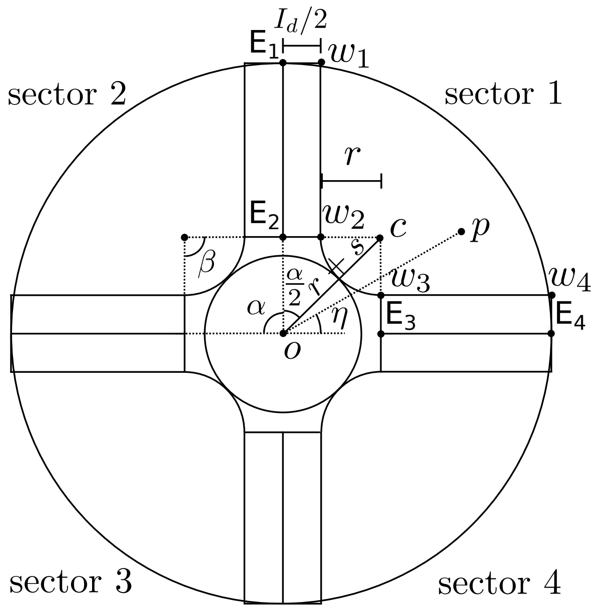


Figure 7: The robots follow the waypoints w_1, w_2, w_3 and w_4 depending on the sector where they are located. The centre c for the curved trajectory between waypoints w_2 and w_3 is distant by $r + s$ from the target centre o. In this example, we have $K = 4$ and a robot at position p and at sector 1.

the target located at point o. Figure 7 shows these four waypoints, which are based on the position of the robot p, and must be followed in order. The four waypoints are calculated depending on the sector where they are. There are K sectors, one per lane, numbered from 1 to K. Let η be the angle of the vector $p - o$. Thus, a sector i of a robot is calculated by $i = \lfloor \frac{\eta}{\alpha} \rfloor + 1$.

Let E_1 be the point in the entering ray on the boundary of the circle centred at o with radius D, E_2 be the point in the entering ray perpendicular to the point where the robot starts to turn in the touch and run strategy (as presented in Figure 4 (b)), E_3 be the point in the exiting ray perpendicular to the point where the robot ends the turn in the touch and run strategy, and E_4 be the point in the exiting ray on the boundary of the circle centred at o with radius D. In Figure 7, these points are exemplified for sector 1. For a given sector i, the entering and exiting rays will have angle $i\alpha$ and $(i-1)\alpha$ with the x-axis, respectively. The position vector w_1 is given by the sum of the vector o, the vector going from o to E_1 and the vector of modulus $I_d/2$ from E_1 rotated $\pi/2$ anti-clockwise from the entering ray. Thus,

$$\begin{aligned} w_1 &= o + (E_1 - o) + \frac{I_d}{2}(\cos(i\alpha - \pi/2), \sin(i\alpha - \pi/2)) \\ &= o + D(\cos(i\alpha), \sin(i\alpha)) + \frac{I_d}{2}(\sin(i\alpha), -\cos(i\alpha)). \end{aligned} \tag{5}$$

The vector w_2 is given similarly to w_1 but using E_2. The distance from o to E_2 is $\sqrt{(r+s)^2 - (r + I_d/2)^2}$, because this is the same as $\overline{AC}$ in the Figure 4 (b), but using d instead of I_d. In other words, in this figure, we have the distance d_r from the target centre where the robots begin turning. By symmetry, this is the same distance from the target centre where the robots stop turning. From the right triangle ABC on that figure, we have $|\overline{AC}| = \sqrt{(r+s)^2 - (r + d/2)^2}$.

The value of r can be calculated from Figure 4 (b) replacing d by I_d as well. That is, we can see that the right triangle ABE has angle $\widehat{EAB} = \alpha/2$, hypotenuse $r + s$ and cathetus $r + d/2$. Hence, it directly follows that

$$sin(\alpha/2) = \frac{r + d/2}{r + s} \Rightarrow r = \frac{s\sin(\alpha/2) - I_d/2}{1 - \sin(\alpha/2)}, \tag{6}$$

after reordering and replacing d by I_d. Then,

$$
\begin{aligned}
w_2 &= o + (E_2 - o) + \frac{I_d}{2}(\cos(i\alpha - \pi/2), \sin(i\alpha - \pi/2)) \\
&= o + \sqrt{(r+s)^2 - (r + I_d/2)^2}(\cos(i\alpha), \\
&\quad \sin(i\alpha)) + \frac{I_d}{2}(\sin(i\alpha), -\cos(i\alpha)).
\end{aligned} \tag{7}
$$

We use similar reasoning to calculate w_3 and w_4, however, for the exiting ray, we use the angle $(i-1)\alpha$ instead of $i\alpha$. Thus,

$$
\begin{aligned}
w_3 &= o + (E_3 - o) + \frac{I_d}{2}(\cos((i-1)\alpha - \pi/2), \sin((i-1)\alpha - \pi/2)) \\
&= o + \sqrt{(r+s)^2 - (r + I_d/2)^2}(\cos((i-1)\alpha), \sin((i-1)\alpha)) + \\
&\quad \frac{I_d}{2}(\sin((i-1)\alpha), -\cos((i-1)\alpha)),
\end{aligned} \tag{8}
$$

and

$$
\begin{aligned}
w_4 &= o + (E_4 - o) + \frac{I_d}{2}(\cos((i-1)\alpha - \pi/2), \sin((i-1)\alpha - \pi/2)) \\
&= o + D(\cos((i-1)\alpha), \sin((i-1)\alpha)) + \frac{I_d}{2}(\sin((i-1)\alpha), \\
&\quad -\cos((i-1)\alpha)).
\end{aligned} \tag{9}
$$

For the sector i, the curve with radius r has centre c, which is distant by $r + s$ from the target centre. The vector $c - o$ has orientation $i\alpha - \frac{\alpha}{2}$ (Figure 7), so

$$
c = o + (r+s)\left(\cos\left(\left(i - \frac{1}{2}\right)\alpha\right), \sin\left(\left(i - \frac{1}{2}\right)\alpha\right)\right). \tag{10}
$$

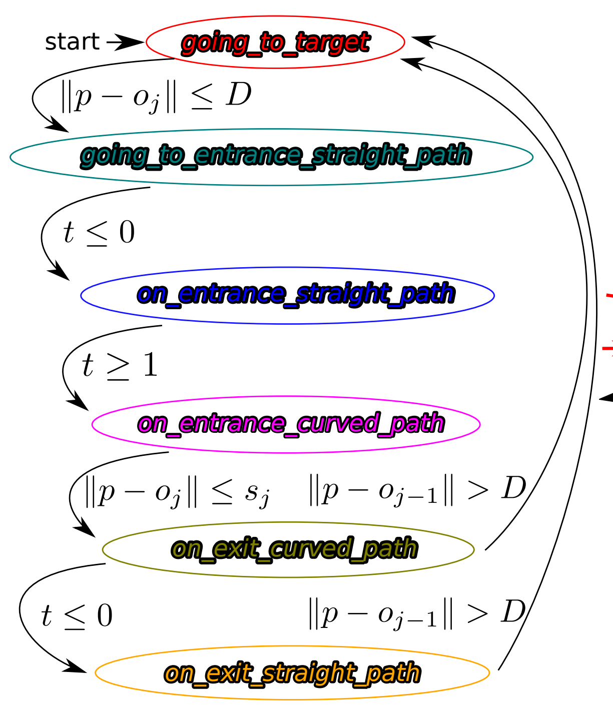
In order to keep track of the path following, we also use a state machine. Every robot has six states. The initial state is *going_to_target.* The robot remains in this state until it reaches the distance D from the target, then it changes to *going_to_entrance_straight_path.* In this state, the robot follows an orbit following vector field centred at the target centre o and radius D using Algorithm 3 with the parameter $w_f = w_1$. The robot changes to the state *on_entrance_straight_path* when the Algorithm 3 outputs $t \leq 0$. This state indicates that the robot is following the straight-line vector field towards the target region until it reaches a position after the line orthogonal to the following line at the waypoint w_2. This happens when the variable t returned by the Algorithm 2 is greater than or equal to 1.

When this occurs, the robot changes to *on_entrance_curved_path* and is impelled by the sum of two forces: the orbit following force for the curved trajectory with centre at c and radius r and an attraction force towards the target centre o. The second force was added because, next to the target region, as the number of robots grows, repulsive forces may push a robot on state *on_entrance_curved_path* far from the target. The orbit following force field by itself does not attract to the target region, so the other force is a counter-effect to this pushing. However, the attractive force magnitude must be higher than the orbit following force magnitude to avoid their vector sum being null. Consequently, we use fixed norm $1.5K_{TRVF}$ for that attractive force, sum these two vectors, then normalise the result to K_{TRVF}. For the Algorithm 3 in this state we use $w_f = w_3$.

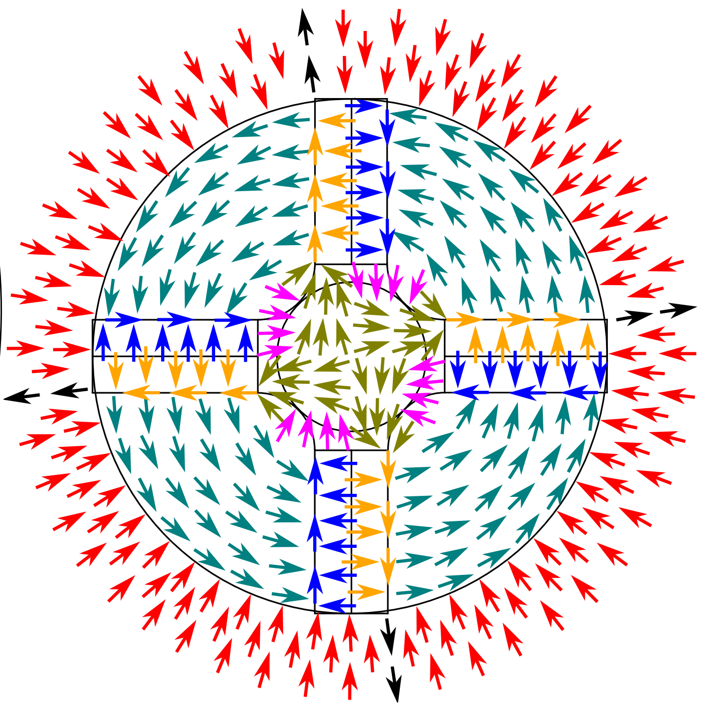
When the robot reaches the target region, it changes to *on_exit_curved_path.* In this state, the robot continues to follow the previous orbit following but we add an attractive force towards the waypoint w_3 by a similar normalised summation as described for the previous state. The transition to the next state, *on_exit_straight_-path*, is done when Algorithm 3 returns $t \leq 0$. Then, the robot follows the straight line following force field until it leaves the circle with radius D around the target centre, changing back to *going_to_target* and going to the next target region. We also added a transition from the state *on_exit_curved_path* to *going_to_target* because D could be not much greater than s, so the robot could leave the circle with radius D due to repulsive forces before making a transition to *on_exit_straight_path.*

As the robot is attracted to the next target when it is on *going_to_target*, depending on its position, it could be impelled to cross the previous target region again. Thus, we apply a repulsive force from the circle with radius D and centre at the previous target centre as if it was a huge obstacle. This makes the robots avoid the previous target region giving space to other robots entering or exiting. Figure 8 (a) summarises these transitions and Figure 8 (b) shows the vector field and the expected state for a robot at the displayed positions assuming no influence of the repulsive force of the other robots.

Finally, Algorithm 4 presents the TRVF algorithm. The condition at line 2 can be checked by a global boolean variable initialised as false before the robot goes to the next target. Without this condition, the robots would change lanes if they get pushed to another one, resulting in more congestions. At lines 9 and 32, we assume any chosen repulsive force function.



(a) State machine



(b) Vector field

Figure 8: Vector field and state machine for the TRVF algorithm. The colours in the vector field represent the expected robot state at the shown vector positions. The black vector indicates the effect of the repulsive force from the target region when the robot is more distant than D from the target already reached. The inner and outer circles have a radius of s and D, respectively. Here, we are using $K = 4$.

4 Experiments and Results

To evaluate our approach, we executed several simulations using the Stage robot simulator (Gerkey et al., 2003). We tested the algorithms for holonomic and non-holonomic robots in simulations. The robots measure $0.44 \times 0.44 \times 0.44$ m^3. When testing non-holonomic robots, we used the control equations given in Luca and Oriolo (1994). Our implementation considered the following equation for the repulsive forces (Siegwart and Nourbakhsh, 2004):

$$F = \begin{cases} -K_{Rep}\left(\frac{1}{d} - \frac{1}{I}\right)\frac{q-p}{d^3}, & \text{if } d < I, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where $K_{Rep} > 0$ is a constant, p the current position of the robot, q the position of the neighbour, $d = \|q - p\|$ the Euclidean distance between q and p, and I the influence radius.

We ran scenarios where robots are initially in a random position with a distance between 13 and 21 m from the target centre. After reaching the common target, robots will go towards the next one, which will be either to the left or to the right of the common target. This is decided randomly, according to a uniform probability (so roughly half of the robots would go to the left and half to the right). The new targets are aligned with the common target but far away in the x-axis.

Two kinds of experiments were performed, one for testing the SQF and TRVF algorithms and another to compare them with the state-of-the-art algorithms. Hyperlinks to the video of executions are available in the captions of each corresponding figure. They are in real-time so that the reader can compare the time and screenshots presented in the figures in this section with those in the supplied videos.[1]

4.1 Proposed algorithms

We compare here our algorithms with state-of-the-art algorithms for collision avoidance in the common target problem (Marcolino et al., 2017): PCC, EE, and PCC-EE. The default parameters used for the algorithms are shown in Table 1. For each experiment, we ran 40 executions, and we report in the graphs the average

[1]The source codes of each experimented algorithm are in `https://github.com/yuri-tavares/swarm-common-target-area-congestion`.

```
Input  : $K_{TRVF}$ : force magnitude;
         $K$: number of lanes;
         $D$ : the length of the corridor;
         $o_1 = (o_{1,x}, o_{1,y}), \dots, o_n = (o_{n,x}, o_{n,y})$ : a list of $n \geq 2$ circular target region centres;
         $s_1, \dots, s_n$ a list of $n$ circular target region radii;
         $j$ : the current target index;
         $I_d$ : influence of the robot;
         $k_s > 1, k_o > 1$: constant for exponentiation in the straight line following and orbit following
         vector fields, respectively;
         $K_r$ : constant for proportional angular speed controller;
         $v$: maximum linear speed.
Output: $F = (F_x, F_y)$ : force vector;
1  Get position of the robot $p = (p_x, p_y)$, its orientation $\xi$, and let $o = o_j = (o_x, o_y)$ and $s = s_j$;
2  if $w_1, \dots, w_4$, and $c$ were not calculated for target index $j$ then
3      $i \leftarrow \left\lfloor \frac{\text{atan2}(p_y - o_y, p_x - o_x)}{2\pi/K} \right\rfloor + 1$;
4      Use Eqs. 5-10 to calculate $w_1, \dots, w_4, r$ and $c$;
5  if state = going_to_target then
6      if $\|p - o\| \leq D$ then state $\leftarrow$ going_to_entrance_straight_path;
7      else
8          $F \leftarrow K_{TRVF} \frac{o-p}{\|o-p\|}$;
9          $F_R \leftarrow$ repulsiveForceFromTarget$(o_{j-1}, D)$;
10         return $K_{TRVF} \frac{F+F_R}{\|F+F_R\|}$;
11 if state = going_to_entrance_straight_path then
12     $(F, t) \leftarrow orbitPathFollowing(K_{TRVF}, o, D, p, \xi, w_1, k_o, K_r, v)$;
13     if $t \leq 0$ then state $\leftarrow$ on_entrance_straight_path;
14     else return $F$;
```

Algorithm 4: TRVF algorithm (continues on Algorithm 5).

and the confidence interval such that $\rho = 0.01$. An appropriate statistical test with such ρ is performed each time we mention statistical significance or say "significantly better". Since robots may not reach the target in some simulations, they were stopped after 60 minutes. This value was chosen because the total simulation time obtained at the runs where all robots reached the target without deadlocks was less than 45 minutes.

The next sections present the experiments and results for SQF and TRVF algorithms varying certain parameters, then a comparison between all algorithms is presented.

4.1.1 SQF algorithm

Figure 9 shows an execution of the SQF algorithm, with 100 non-holonomic robots with default parameters. Figure 9 (a) shows the initial positions of the robots around the target area (small circle in the centre). The larger circle around the target shows the distance D where the rotational field will take place. Hence, robots inside the larger circle perform a rotational movement towards the corridor and soon reach the state in Figure 9 (b), where the first robots reach the corridor. These robots will directly move towards the target area (Figure 9 (c)), and then leave the region following a different rotational field, in yellow in Figures 9 (d) and (e). Eventually, all robots can reach the target. Robots in black are at a distance greater than D and are now going towards their next target, randomly chosen between left and right sides.

We analyse the throughput for a growing number of robots, considering holonomic and non-holonomic robots. Figure 11 displays the results for the experiments in comparison with the least bound of asymptotic throughput (that is, the limit of the throughput when the number of robots tends to infinity) from the hexagonal packing corridor strategy that inspired the SQF algorithm. This value was derived in our concurrent theoretical work (Passos et al., 2022) and is given by

$$\frac{4vs}{\sqrt{3}d^2} - \frac{2v\cos(\theta - \pi/6)}{\sqrt{3}d}, \qquad (12)$$

for the linear speed v, the distance between each robot of d, an angle θ of the hexagonal packing arrangement

```
16 if $state$ = $on\_entrance\_straight\_path$ then
17     $(F,t) \leftarrow straightPathFollowing(K_{TRVF}, p, \xi, w_1, w_2, I_d, k_s, K_r, v)$;
18     if $t \geq 1$ then state $\leftarrow$ $on\_entrance\_curved\_path$;
19     else return F;
20 if $state$ = $on\_entrance\_curved\_path$ then
21     if $\|o-p\| \leq s$ then
22         Increment $j$, then let $o = o_j = (o_x, o_y)$ and $s = s_j$;
23         state $\leftarrow$ $on\_exit\_curved\_path$;
24     else
25         $(F_1,t) \leftarrow orbitPathFollowing(K_{TRVF}, c, r, p, \xi, w_3, k_o, K_r, v)$;
26         $F_2 \leftarrow 1.5K_{TRVF}\frac{o-p}{\|o-p\|}$;
27         return $K_{TRVF}\frac{F_1+F_2}{\|F_1+F_2\|}$;
28 if $state$ = $on\_exit\_curved\_path$ or $state$ = $on\_exit\_straight\_path$ then
29     if $\|p-o_{j-1}\| > D$ then
30         state $\leftarrow$ $going\_to\_target$;
31         $F \leftarrow K_{TRVF}\frac{o-p}{\|o-p\|}$;
32         $F_R \leftarrow$ repulsiveForceFromTarget($o_{j-1}, D$);
33         return $K_{TRVF}\frac{F+F_R}{\|F+F_R\|}$;
34     else
35         if $state$ = $on\_exit\_curved\_path$ then
36             $(F_1,t) \leftarrow orbitPathFollowing(K_{TRVF}, c, r, p, \xi, w_3, k_o, K_r, v)$;
37             if $t \leq 0$ then
38                 $F_2 \leftarrow 1.5K_{TRVF}\frac{w_3-p}{\|w_3-p\|}$;
39                 return $K_{TRVF}\frac{F_1+F_2}{\|F_1+F_2\|}$;
40             else state $\leftarrow$ $on\_exit\_straight\_path$;
41         if $state$ = $on\_exit\_straight\_path$ then
42             $(F,t) \leftarrow straightPathFollowing(K_{TRVF}, p, \xi, w_3, w_4, I_d, k_s, K_r, v)$;
43             return F;
```
Algorithm 5: TRVF algorithm (continuation).

and target region radius of s. This value is minimised by $\theta = \pi/6$.

In Figure 11, we are using it with the mean distance between each robot and its closest neighbour and mean linear speed in all experiments for each number of robots. These values do not follow a normal distribution. Thus, instead of plotting the confidence interval with $\rho = 0.01$, we used these means shifted to above and below by one standard deviation to calculate the interval shown in Figure 11. Observe in this figure that the results obtained from the experiments are still below the upper bound obtained by the mean values but inside the one standard deviation interval. Although the high variation in the distance between the robots and linear speed contributes to the difference, the SQF algorithm forms a corridor only in the upper part of the circle with a radius of D around the target. Hence, the robots still need time to get to the corridor. Also, the positions of the robots in the corridor are not too compacted.

4.1.2 TRVF algorithm

For the repulsive force away from the previous target region we used one similar to (11) but the distance is considered from the robot position and the circle with radius D centred at the target position, and the influence radius is D, that is,

$$F = \begin{cases} -K_{Rep}\left(\frac{1}{d} - \frac{1}{D}\right)\frac{1}{d^2}\frac{o-p}{\|o-p\|}, & \text{if } d < D, \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

where $K_{Rep} > 0$ is a constant, p the current position of the robot, o the target centre position, and $d = \|o-p\| - D$ is the distance from the robot position to the circle.

Parameter	**Notation**	**Value**	**Used by algorithm(s)**
Circular target area radius	s	3 m	All
Working radius of the algorithm around the target	D	13 m	All
Coefficient for repulsive forces	K_{Rep}	0.5	All
Default influence radius	I_d	3.0 m	All
Minimum influence radius	I_{min}	1.0 m	SQF
SQF force magnitude	K_{SQF}	2.5	SQF
Constant for proportional angular speed controller	K_r	3	TRVF
TRVF force magnitude	K_{TRVF}	2.5	TRVF
Maximum linear speed	v	1 m/s	TRVF
Constant for exponentiation in the straight line following	k_s	1.1	TRVF
Constant for exponentiation in the orbit following vector fields	k_o	1.1	TRVF
Angle of entry region	–	$\frac{2\pi}{3}$	EE, PCC-EE
Communication radius	–	3 m	PCC, PCC-EE
Radius of free region	–	3.7 m	PCC, PCC-EE
Angle of α-area for waiting robot	–	$\frac{23\pi}{36}$	PCC, PCC-EE
Angle of α-area for locked robot	–	$\frac{\pi}{4}$	PCC, PCC-EE
Radius of α-area	–	3 m	PCC, PCC-EE
Number of cycles before sending a message	–	25	PCC, PCC-EE
Number of cycles for testing if a waiting robot will change state	–	40	PCC, PCC-EE
Radius of free region	–	3.7 m	PCC, PCC-EE
Radius of danger region	–	5.2 m	PCC, EE, PCC-EE

Table 1: Default values for simulation parameters with the notation used in this paper and algorithms that use it.

Figure 12 displays an execution of the TRVF algorithm, with 100 non-holonomic robots and four lanes with default parameters. Figure 12 (a) shows the initial positions of the robots around the target area (small circle in the centre). The larger circle around the target shows the distance D, and the robots will avoid it when they go to the next target using (13). Each robot goes to the entering lane next to its position and follows the force field (Figure 12 (b)). The robots avoid crossing the target region and follow the border of the circle with radius D when going to the next target in Figures 12 (c), (d) and its continuation on Figure 13 so that all robots can reach the target. With this algorithm, the cluttering is distributed through the entrances and the target region according to the number of lanes, while the leaving lanes become free. However, without the TRVF algorithm, the clutter would concentrate only next to the target region.

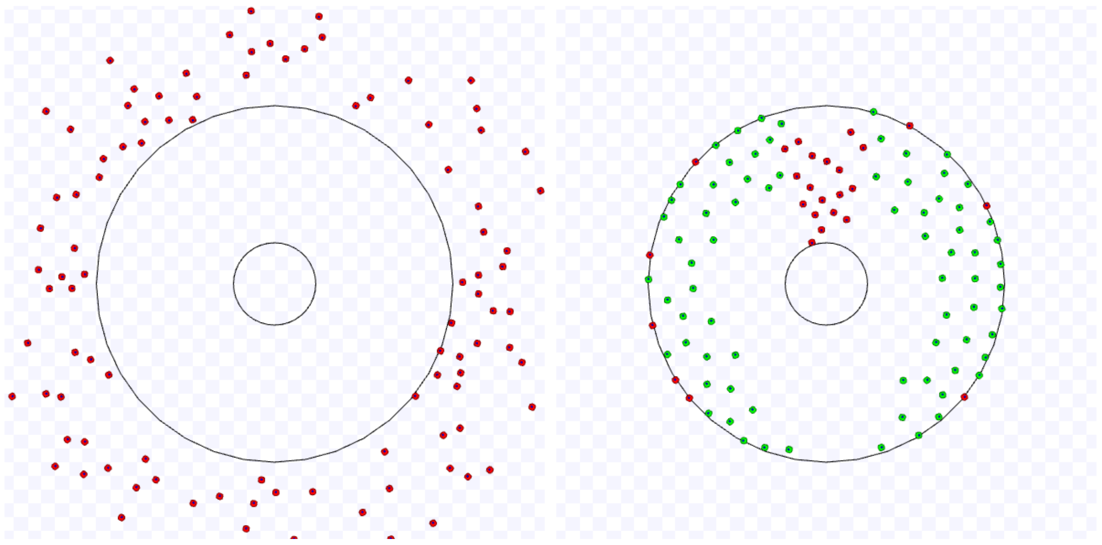
We performed simulations to compare different K values for the defaults parameters for a number of robots ranging from 20 to 300 with increments of 20 with the default values in Table 1. The results are in Figure 14.

Firstly, we compare with the results for the touch and run strategy by calculating the asymptotic throughput for the allowed values of K – that is, $K \in \{3, \ldots, 6\}$ – following the bound found in(Passos et al., 2022)

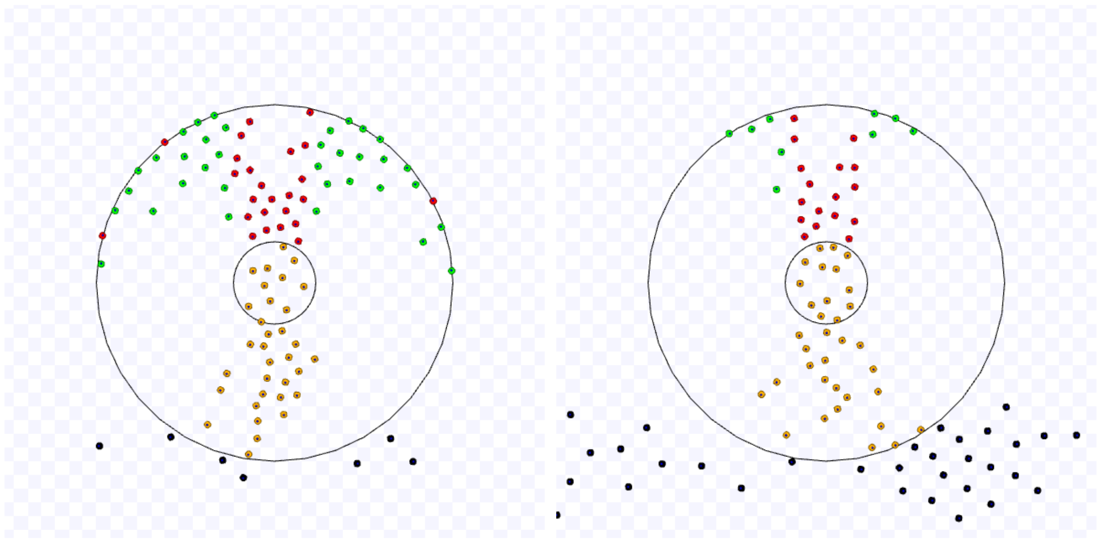
$$\frac{Kv}{\max(d, d')} \text{ for } d' \quad = \begin{cases} r(\pi - \alpha) + \frac{d - 2r\cos(\alpha/2)}{\sin(\alpha/2)}, & \text{if } 2r\cos(\alpha/2) < d, \\ 2r \arcsin\left(\frac{d}{2r}\right), & \text{otherwise.} \end{cases} \tag{14}$$

For K equals 3, 4, 5 and 6, the asymptotic throughput rounded to three decimal places are 0.994, 1.2, 1.099 and 1, respectively. Thus, $K = 4$ would be the best number of lanes for the touch and run strategy. However, as in the TRVF experiments, the distances between robots and linear speeds change over time, and the movement of the robots is influenced by the other robots in their neighbourhood. Therefore, the best K was 5 according to the results in Figure 14.

Figure 15 shows a screenshot in the middle of the execution for the allowed K values and 100 holonomic robots. Using $K = 6$, the default values $s = 3$ m and $d = I_d = 3$ m in (6) gives $r = 0$, so the TRVF algorithm will run but no curved trajectory is made. In this case, the entrance and exiting straight lanes lie exactly in the target region. This could be a shorter trajectory, as the robots do not need to make the curved turn, but congestions happen in the target region (Figure 15 (d)). In fact, for any K displayed in Figure 15, observe that some robots are trapped inside the target area. They were pushed inside the area due to repulsive forces from other robots, and they inflicted those forces on the robots getting near the target region from the curved trajectory, consequently causing congestion as well. However, for the best $K = 5$, congestion is minimised.


(a) 0s: Initial positions.

(b) After 49.8 s.


(c) After 142.8 s.

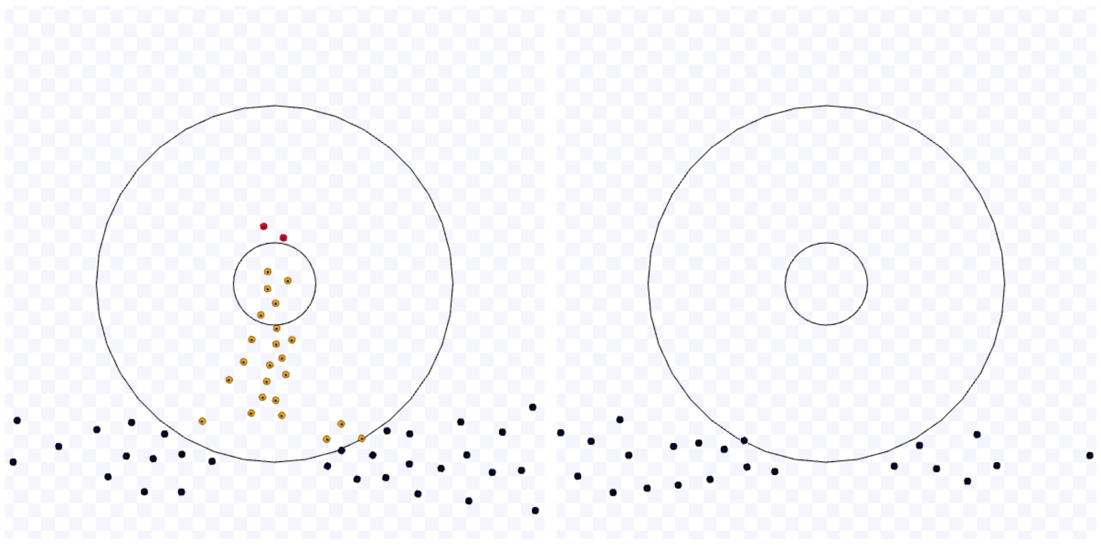
(d) After 214.3 s.

Figure 9: Screenshots of the SQF algorithm, with 100 non-holonomic robots with default parameters. The red, green and yellow robots are in state *going_to_target*, *going_to_corridor* and *leaving_target*, respectively. Black robots are going to their next target. This continues in Figure 10. Available on `https://youtu.be/3-d7Y7eViW4`.

As done for the SQF algorithm, we analyse the throughput for a growing number of robots, considering holonomic and non-holonomic robots. Figure 16 displays the results for the experiments in comparison with the theoretical maximum throughput value. The theoretical maximum throughput was obtained by (14) for $K = 5$ with the mean distance between the robots and mean speed in all experiments for each number of robots. As before, in order to plot the offset from the mean values, we used the standard deviation instead of the confidence interval here because these values do not follow a normal distribution. From that figure, the experimental throughput is still below the upper bound obtained by the mean values but inside the one standard deviation interval. The difference between the experimental data and the theoretical value obtained from the mean values occurs because, by using variable linear speed, the robots are not constantly paced towards the target. Additionally, next to the curve, the robots deviate from the trajectory to avoid bumping into each other.

4.2 Comparison with state-of-art algorithms

In this section, we compared the algorithms regarding (i) the throughput, (ii) the time the swarm reaches the target area, (iii) the time for leaving it averaged by the number of robots, and (iv) the total simulation time. We only consider the runs that succeed in terminating within 60 minutes in Section 4.2.1 and 20 minutes in Section 4.2.2. This termination time is lower in the later section because we use 100 robots, while in the former, up to



(e) After 285.7 s. (f) 357.2 s: ending of the simulation.

Figure 10: Continuation of Figure 9.

300. We observed that the robots running algorithms PCC, EE and PCC-EE for the experiments of Section 4.2.2 entered deadlock in executions longer than 20 minutes for 100 robots.

The total simulation time is obtained when the last robot in the experiment arrives at the outer circle with a radius of D after everyone has reached the target area. The reaching time is measured from the last robot in the swarm that gets to the target area. The leaving time is taken from the moment a robot reaches the target area until it arrives at the outer circle. As done in Marcolino et al. (2017), to measure the effectiveness of the algorithms on the crowd reduction for leaving the target area, we summed the leaving time of every robot and divided it by the number of robots in the experiment.

Additionally, we performed tests with the target area with constant radius (as in Table 1) in Section 4.2.1 and varying small values in Section 4.2.2.

4.2.1 Comparison for constant target size

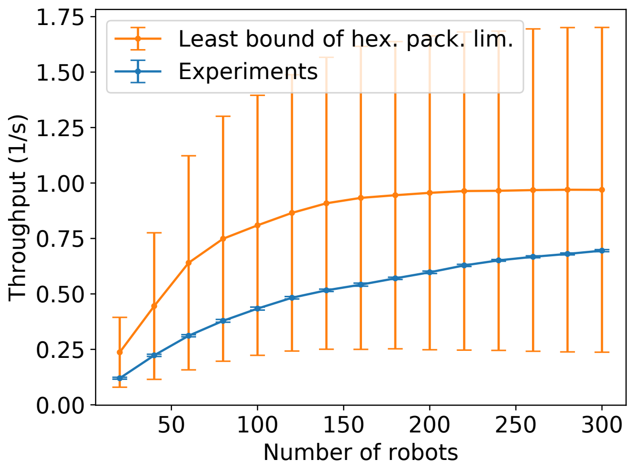
Comparison of the reaching time and throughput Figures 17 and 18 show the comparison for a varying number of robots of the target area throughput and the reaching time, respectively. For both types of robots, throughput and time increase with the number of robots. Additionally, observe in the comparisons that higher throughput reflects a lower arrival time.

Moreover, we notice in the throughput graph that, for holonomic robots from 240 robots, the SQF algorithm is significantly better than all other algorithms and, from 100 robots, for non-holonomic ones. In the reaching time graph, this only occurs from 280 for holonomic and 120 for non-holonomic ones with statistical significance by t-test. However, below these values, EE is still better by throughput. Excluding the EE algorithm, the TRVF algorithm is better than all the remaining algorithms from 40 to 120 individuals for holonomic robots but, for non-holonomic robots, it is only better than SQF and PCC algorithms just for 40 and 60 robots.

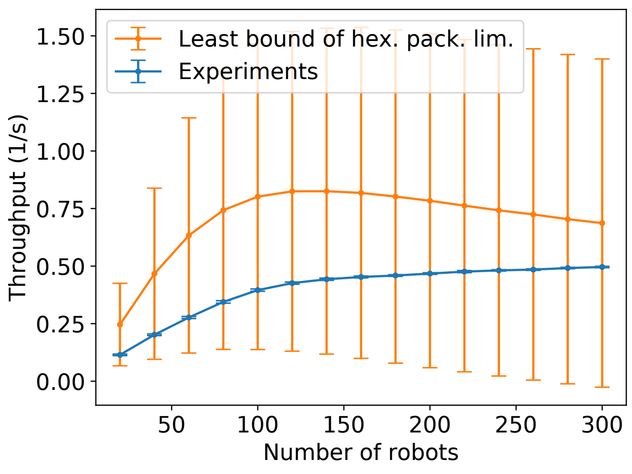
Furthermore, unpaired t-tests with $\rho = 0.01$ returned that the reaching time intervals of SQF and EE algorithms for 240 and 260 holonomic robots – Figure 18 (a)) – have the same mean. These tests showed that the reaching time intervals for 100 non-holonomic robots of SQF and EE algorithms – Figure 20 (b) – also have the same mean.

Comparing only TRVF and SQF by the target reaching time, the former is better than the latter until 120 individuals for holonomic robots and up to 40 for the non-holonomic case with statistical significance. Unpaired t-tests returned that SQF and TRVF reaching time mean intervals have different means except for 140 holonomic and 60 non-holonomic robots. As the SQF algorithm organises a queue above the target region, the robots must initially follow the way until reaching that queue. After that, the robots flow through it. On the other hand, in the TRVF, the robots go directly to the target region. However, as the number of robots increases, more congestions happen near this region because of the repulsive forces caused by the robots doing the curve.

Observe as well that, although SQF has higher throughput than TRVF from some number of robots, the comparison of their corresponding inspiration strategies, hexagonal packing and touch and run, had a different result with respect to the asymptotic throughput presented in (Passos et al., 2022). For calculating asymptotic throughput, we considered constant the maximum linear speed and the minimum distance between robots.



(a) Holonomic



(b) Non-holonomic

Figure 11: Throughput of SQF algorithm by the number of robots from 20 to 300 in steps of 20 for the experiments and the least bound (12) with $\theta = \pi/6$, using the mean distance between the robots and mean linear speed from experiments – bars represent the shift of the means by one standard deviation to above and below.

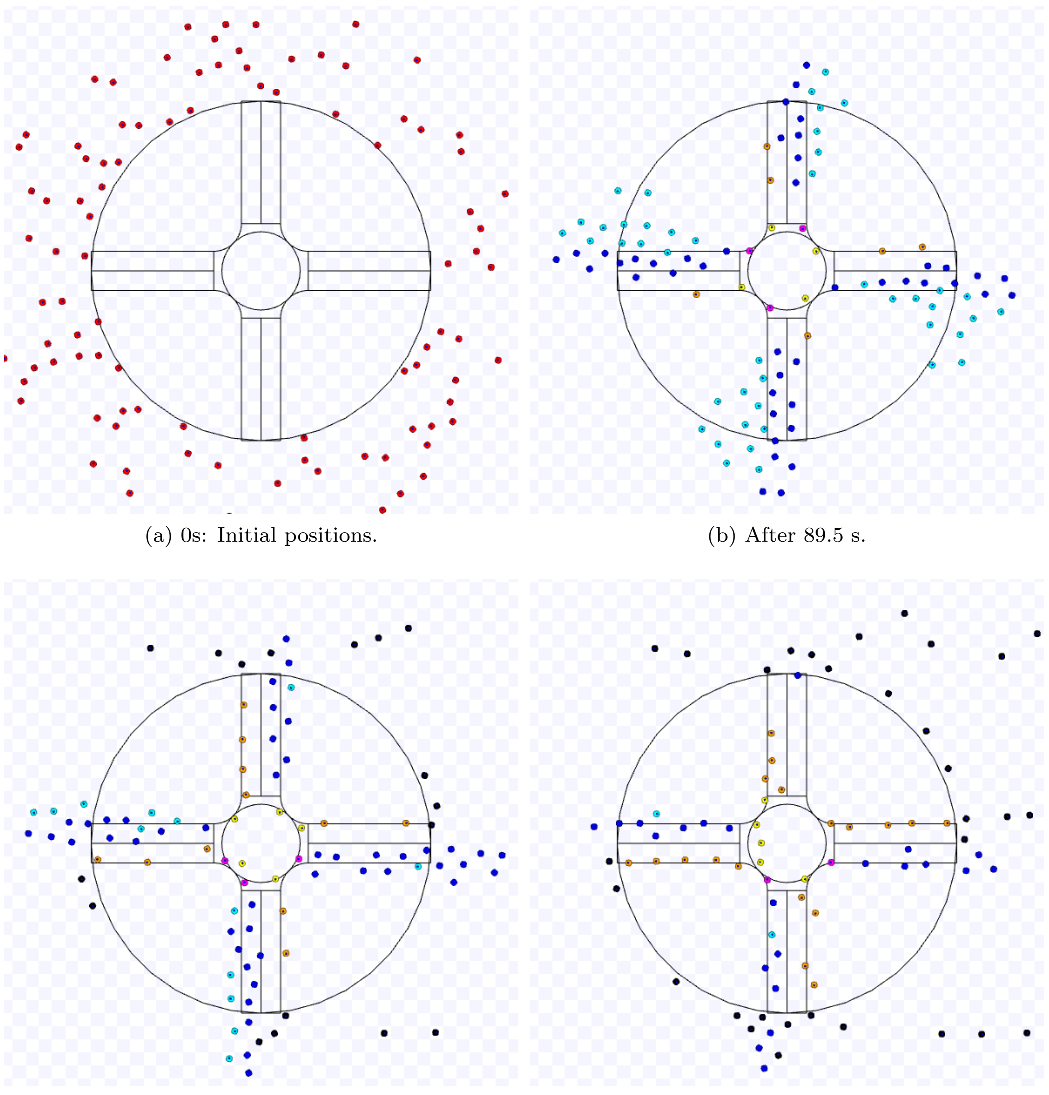
Using only asymptotic throughput, we can evaluate which one is the best strategy in a scenario with those fixed quantities. For robots using artificial fields, getting an explicit asymptotic throughput equation is difficult due to the changeability of the speed and the distance between the robots, but this does not prevent using experimental throughput for comparisons. Accordingly, we notice that this changeability and the effect of the other robots in the trajectory yield the SQF being better than TRVF, although the touch and run is better than hexagonal packing. Even so, the analytically calculated throughputs are still upper bounds on the ones obtained from simulations, as observed in Figures 11 and 16.

Comparison of the average leaving time Figure 19 displays the comparison for the number of robots versus the average leaving time. In Figure 19 (a), the TRVF algorithm is significantly better than all the others until 240 individuals for holonomic robots and 140 for non-holonomic ones. For holonomic robots, from 280 individuals, PCC-EE has less average leaving time than the other, while for non-holonomic robots, SQF is better from 180 individuals with statistical significance.

As the number of robots increases, the TRVF algorithm has higher average leaving time because more robots gather next to the target area, trying to go away. For non-holonomic robots, this number is lower as they demand more time to avoid other robots when moving under non-holonomic constraints.

For holonomic robots, robots using the SQF algorithm take more time to leave the target due to the curved path that they must follow caused by the rotational force field to leave the target area. By contrast, the robots with PCC-EE follow almost a straight line from the target to leave that area, as it frees regions for leaving (as seen in Figure 22 (b)). However, in the non-holonomic case, the robots need to make more turns or reduce linear speed to avoid others next to the target until they reach one of those regions where they can move in straight lines. With SQF, the rotational speed variation for leaving is low, and the robots can maintain linear speed most of the time, as few robots are moving in the opposite direction.

Comparison of the total simulation time Figure 20 shows the comparison for the total simulation time. Also, note that the average time to leave the target is less than the time to reach it – less than 15% – justifying the low difference in the shape of the graphs for the total simulation time and the reaching time. Apart from the shifting in the values on these graphs, SQF is better, regarding the simulation time, only starting from 280 holonomic robots (the same for the reaching time). Unpaired t-tests with $\rho = 0.01$ returned that the simulation time intervals of SQF and EE algorithms for 240 and 260 holonomic robots – Figure 20 (a) – have the same mean. For non-holonomic robots, the total time of SQF is less than the obtained from EE for 120 robots, as occurred for the reaching time. Likewise, unpaired t-tests showed that the simulation time intervals for 100 non-holonomic robots of SQF and EE algorithms – Figure 20 (b) – have the same mean (as happened for reaching time).


(a) 0s: Initial positions.

(b) After 89.5 s.

(c) After 179 s.

(d) After 268.5 s.

Figure 12: Screenshots of the TRVF algorithm, for 100 non-holonomic robots, four lanes and default parameters. The colour of the robots are the same for their states shown in Section 3, that is, red, light blue, blue, magenta, yellow and orange for *going_to_target*, *going_to_entrance_straight_path*, *on_entrance_straight_path*, *on_entrance_-curved_path*, *on_exit_curved_path* and *on_exit_straight_path* states, respectively. Black robots are going to their next target. This continues in Figure 13. Available on `https://youtu.be/MRzXS_9I2Ls`.

4.2.2 Comparison for varying target sizes

We now analyse the throughput and total simulation time of a varying target size using 100 robots to show that the SQF algorithm outperforms the other algorithms as the target size gets smaller. Only total simulation time is shown here because the leaving time is small compared to the reaching time, as we showed before. In this experiment, we did not use the TRVF algorithm because the restriction in the lower and upper values of K (Passos et al., 2022) forbids the usage of the small values of target area radius for the influence radius used here.

First, we show that current algorithms fail to complete executions for small target sizes. For a given target area radius s, we respectively used $s + 0.7$ m and $s + 2.2$ m for the radius of free and danger regions to PCC, EE and PCC-EE algorithms Marcolino et al. (2017). Figure 21 shows the percentage of failed simulations for different target sizes. For experiments with non-holonomic robots, PCC fails for all target sizes. EE and PCC-EE failed to terminate every run for a target size below 0.6 m, but the number of unfinished experiments decreases until the radius is less than 0.9 m. The SQF can complete all executions for the displayed target sizes. For the holonomic robots experiments, from $s = 0.9$ m all algorithms complete the execution within the time limit and the decrease in the number of failed executions per radius size is greater than the non-holonomic robots experiments.

When the radius of the target area is small, more robots tend to concentrate around it. The attractive force

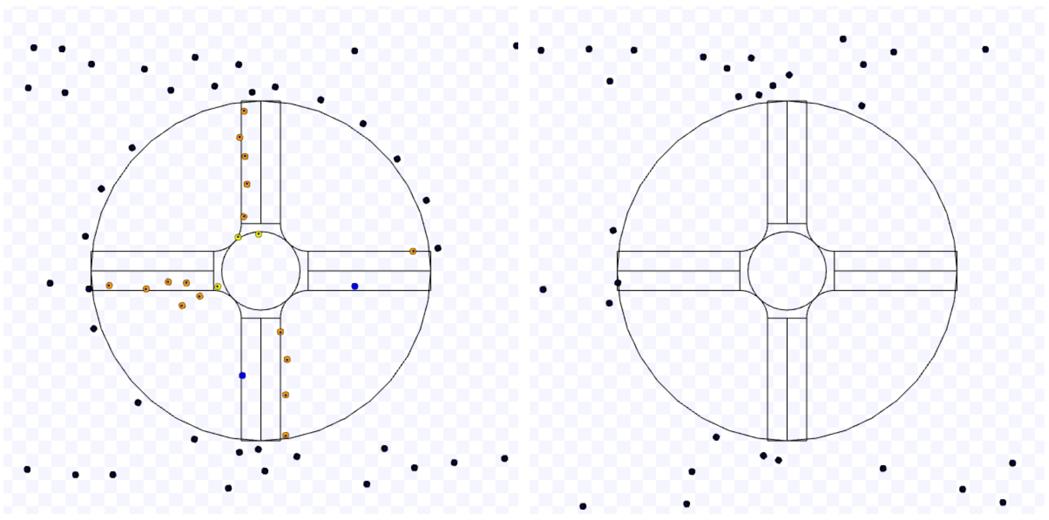
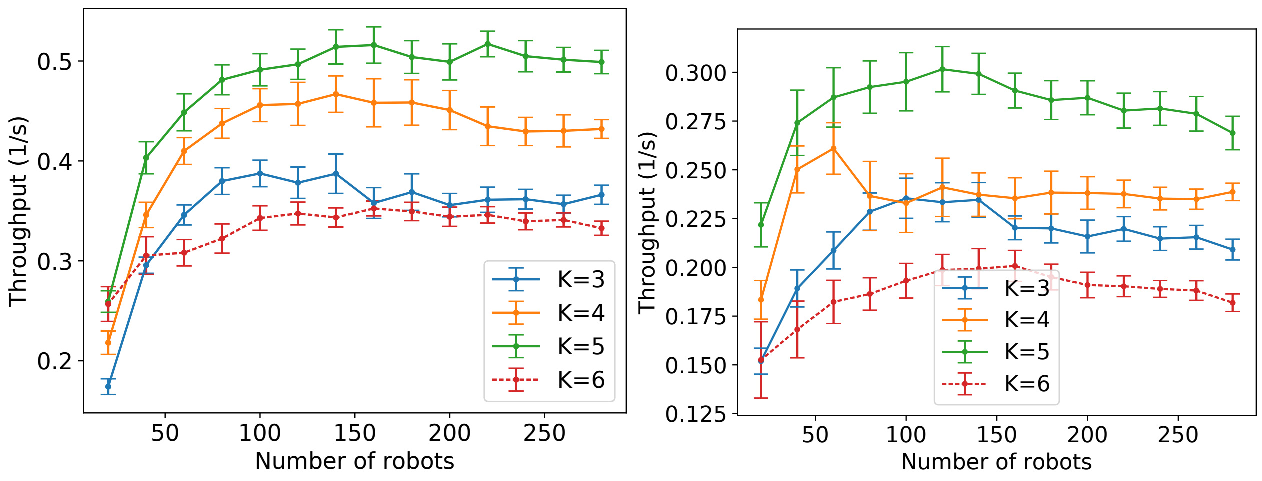

(e) After 358 s. (f) 447.5 s: ending of the simulation.

Figure 13: Continuation of Figure 12.



(a) Holonomic (b) Non-holonomic

Figure 14: Throughput of TRVF algorithm by the number of robots from 20 to 300 in steps of 20 for different values of the number of lanes (K).

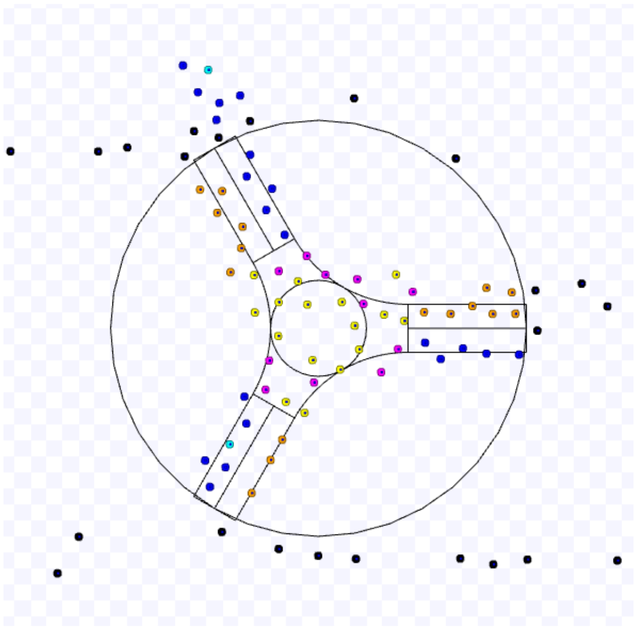
towards the target centre is not enough to counteract the repulsive forces from the nearby robots. This does not create a zero force vector but reduces the attraction to the target centre notably, and the robots slowly and erratically circle around the target. Figure 22 shows examples of this situation.

Figures 23 and 24 display the throughput and time to complete by the radius of the target area. The missing points are caused by failed executions, as shown above. The confidence intervals are calculated taking in account only the successful runs. As shown in Figure 21 (b), PCC failed for all radius values when we used non-holonomic robots, so it is not presented in Figures 23 (b) and 24 (b). SQF significantly outperforms all other algorithms for any target size in both non-holonomic and holonomic cases.

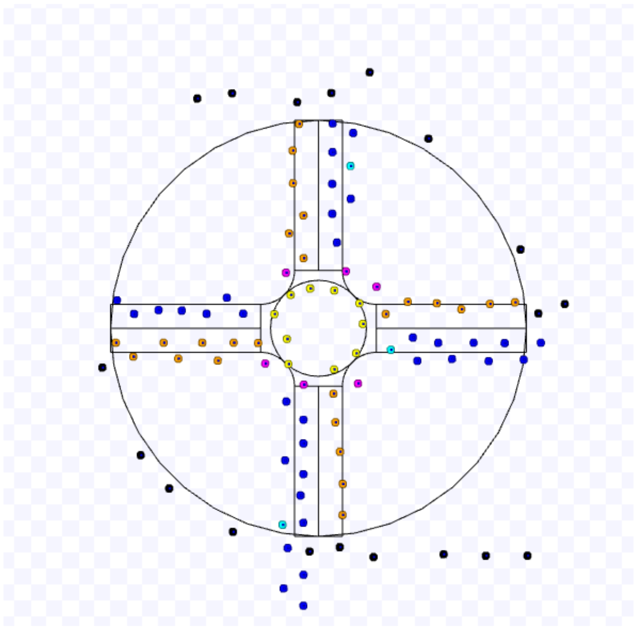
5 Conclusion

The throughput of the common target area is a measure of the effectiveness of algorithms to minimise congestion in a swarm of robots trying to reach the same goal. In a concurrent work (Passos et al., 2022), we analysed two strategies to maximise it in a common circular target region, assuming the robots running at constant maximum linear speed and having a fixed minimum distance between each other. One strategy uses a corridor with robots to enter that region. On the other, the robots follow curved trajectories to touch the region boundary. We used those strategies as inspiration to propose two new algorithms for real-world scenarios: SQF and TRVF.

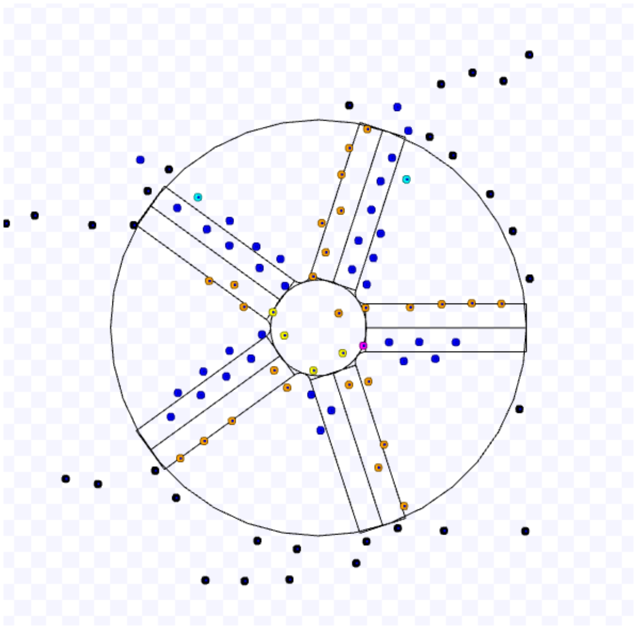
We demonstrated by simulations in the Stage platform that SQF outperforms TRVF and the state-of-art algorithms PCC, PCC-EE and EE by time for reaching the target for a large number of robots, from 280 and



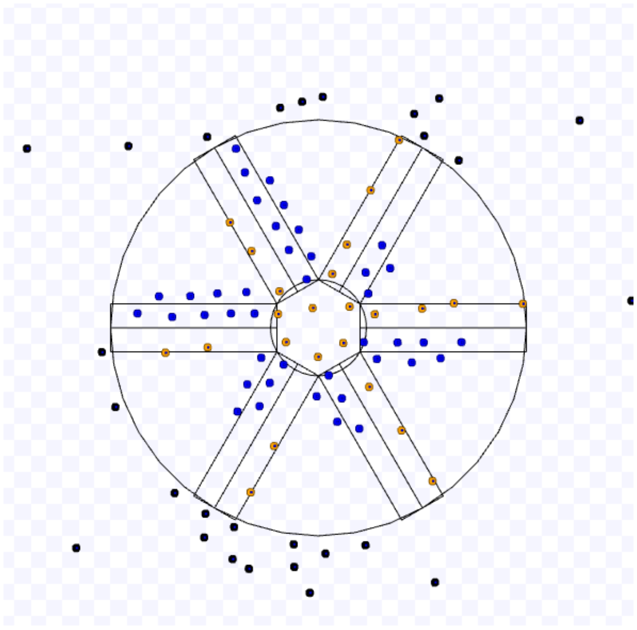
(a) K = 3 and after 202.8 s. Available on `https://youtu.be/0U6ajiBtYw8`.



(b) K = 4 and after 155.5 s. Available on `https://youtu.be/D-_2VK5JRYg`.



(c) K = 5 and after 156.1 s. Available on `https://youtu.be/z92SNJ8ugHs`.

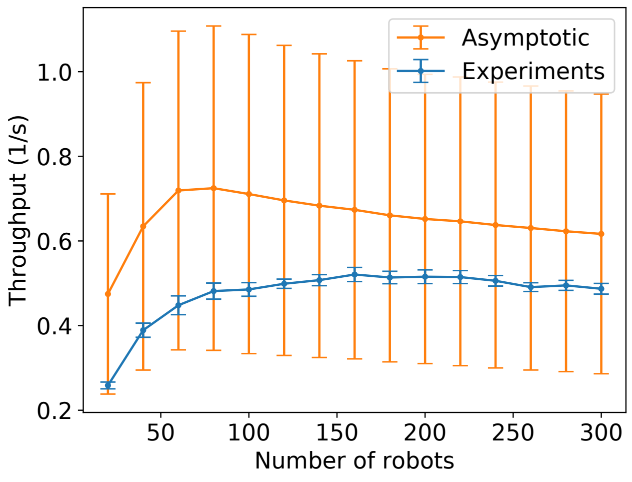


(d) K = 6 and after 177.3 s. Available on `https://youtu.be/BYd8VncXnCQ`.

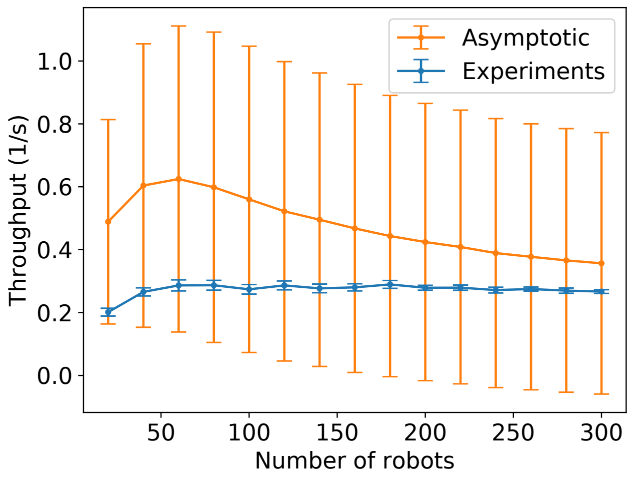
Figure 15: Screenshots of the TRVF algorithm, for 100 holonomic robots, default parameters and different numbers of lanes (K) at the middle of the execution.

120 individuals for holonomic and non-holonomic robots, respectively. From these same numbers of robots, swarms running the SQF algorithm also use less time for total simulation time (i.e., reaching plus leaving the target) than the other algorithms. SQF also outperforms these algorithms concerning the average target leaving time from 180 non-holonomic robots. However, for holonomic robots, TRVF is better up to 240 individuals and PCC-EE from 280 robots. Additionally, we showed that those others might completely fail for circular target regions with an area fitting less than five times the area of a robot, and they are outperformed in throughput by SQF when they succeed.

Even though SQF has more throughput than TRVF for a number of robots greater than 140 for holonomic robots and 60 for non-holonomic ones, the comparison of their corresponding inspiration strategies (hexagonal packing and touch and run, respectively) had reversed outcomes with respect to asymptotic throughput. When we assumed constant maximum linear speed and fixed minimum distance between robots, we could provide analytical calculations of the throughput for a given time and the asymptotic throughput for the different

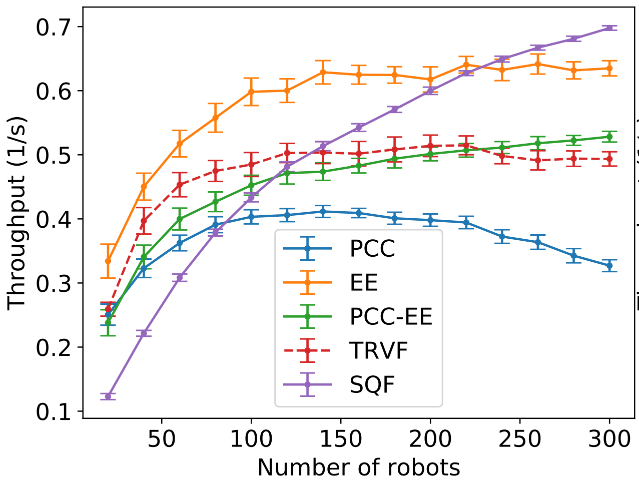


(a) Holonomic.

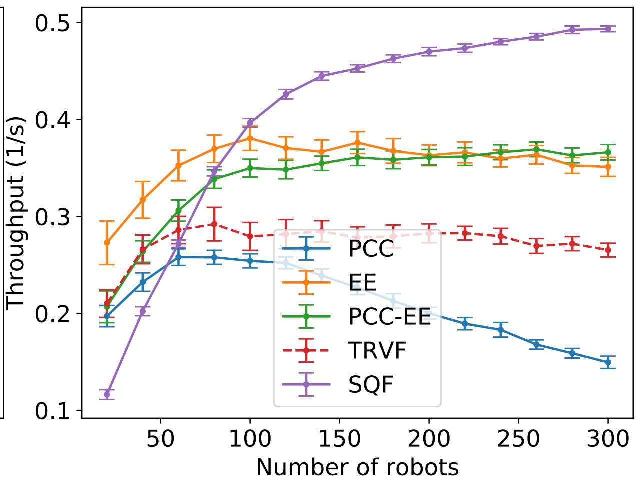


(b) Non-holonomic.

Figure 16: Throughput of TRVF algorithm by the number of robots from 20 to 300 in steps of 20 for the experiments, and asymptotic throughput using $K = 5$, the mean distance between the robots and mean linear speed from experiments – bars represent the shift of the means by one standard deviation to above and below.



(a) Holonomic



(b) Non-holonomic

Figure 17: Throughput comparison of the algorithms for a number of robots from 20 to 300 in steps of 20.

theoretical strategies. Based solely on these calculations, we could compare which strategy is better. However, for robots using artificial potential fields, it is not straightforward to get explicit throughput equations due to the changeability of those quantities previously assumed constant. Then, in the lack of closed asymptotic equations, we performed simulations to get experimental throughput and compare algorithms for varying linear speed and inter-robot distance. As shown by the experimental data, their variation and the effect of the other robots in the trajectory affect the throughput, but the analytically calculated throughputs are still upper bounds on the ones obtained from simulation.

Acknowledgements

Yuri Tavares dos Passos gratefully acknowledges the Faculty of Science and Technology at Lancaster University for the scholarship and the Universidade Federal do Recôncavo da Bahia for granting the leave of absence to finish his PhD.

References

G. Dudek, M. Jenkin, E. Milios, D. Wilkes, A taxonomy for swarm robots, in: Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93), volume 1, 1993, pp. 441–447. doi:10.

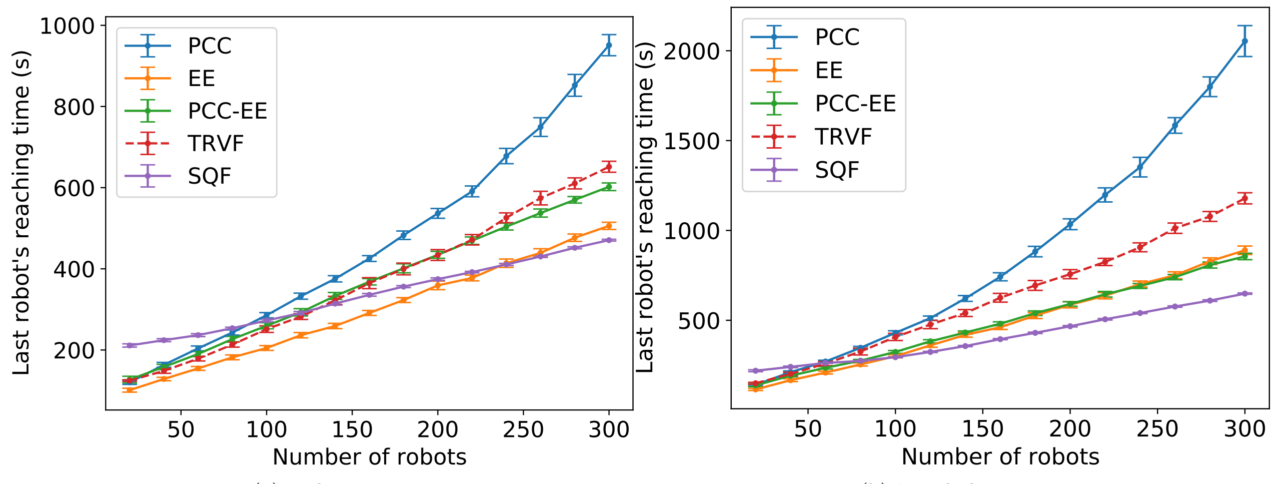


Figure 18: Comparison of the time to reach the target of the algorithms for a number of robots from 20 to 300 in steps of 20.

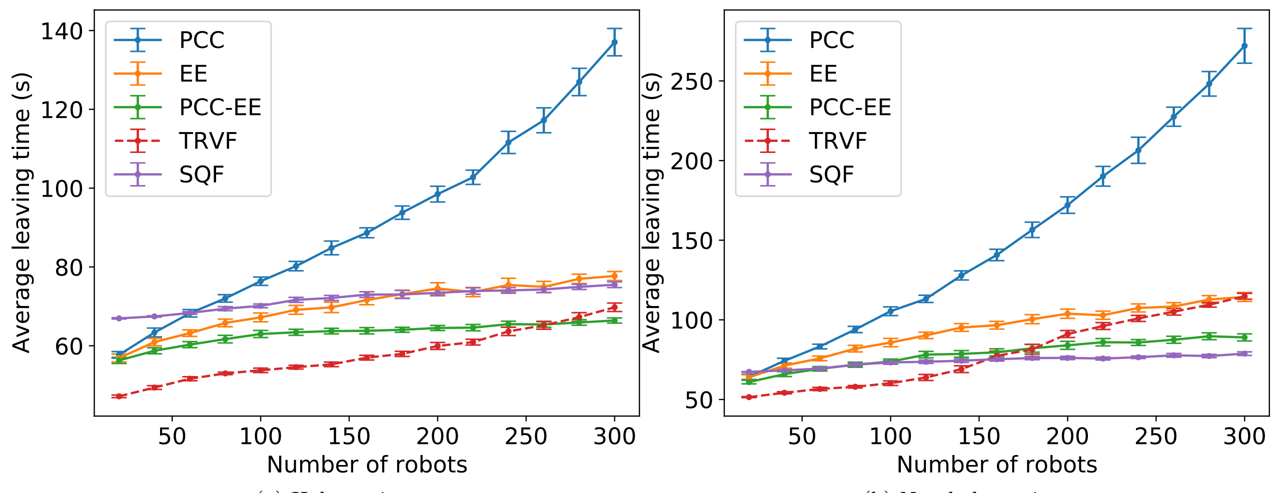


Figure 19: Comparison of the algorithms for a number of robots from 20 to 300 in steps of 20 versus the sum of the time for leaving the target area of all robots in the experiment divided by the number of robots.

1109/IROS.1993.583135.

G. Beni, J. Wang, Swarm intelligence in cellular robotic systems, in: P. Dario, G. Sandini, P. Aebischer (Eds.), Robots and Biological Systems: Towards a New Bionics?, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993, pp. 703–712. doi:10.1007/978-3-642-58069-7.

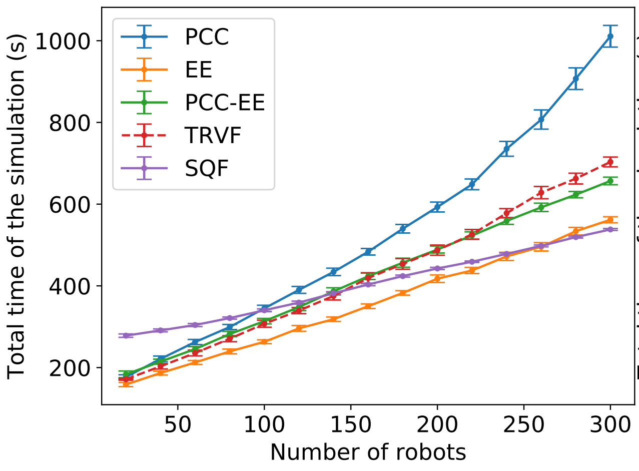
M. J. Mataric, Designing and understanding adaptive group behavior, Adapt. Behav. 4 (1995) 51–80. doi:10.1177/105971239500400104.

S. Panait, Liviu Alexandru Luke, Evolving foraging behaviors, in: Second International Workshop on the Mathematics and Algorithms of Social Insects, 2003, pp. 131–138.

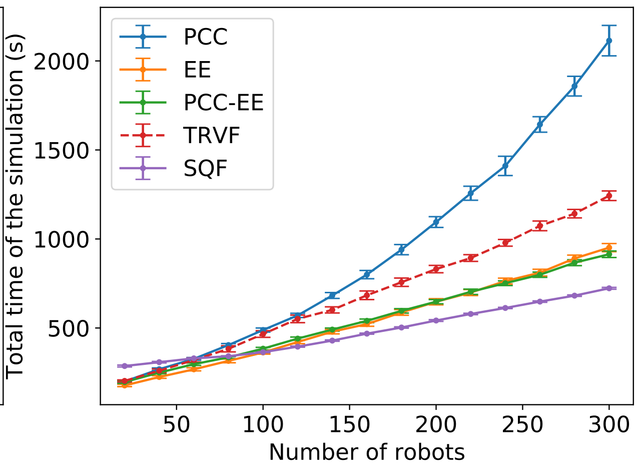
I. Navarro, F. Matía, An introduction to swarm robotics, International Scholarly Research Notices 2013 (2013). doi:10.5402/2013/608164.

S. Garnier, J. Gautrais, G. Theraulaz, The biological principles of swarm intelligence, Swarm Intelligence 1 (2007) 3–31. doi:10.1007/s11721-007-0004-y.

A. Treuille, S. Cooper, Z. Popović, Continuum crowds, ACM Trans. Graph. 25 (2006) 1160–1168. doi:10.1145/1141911.1142008.

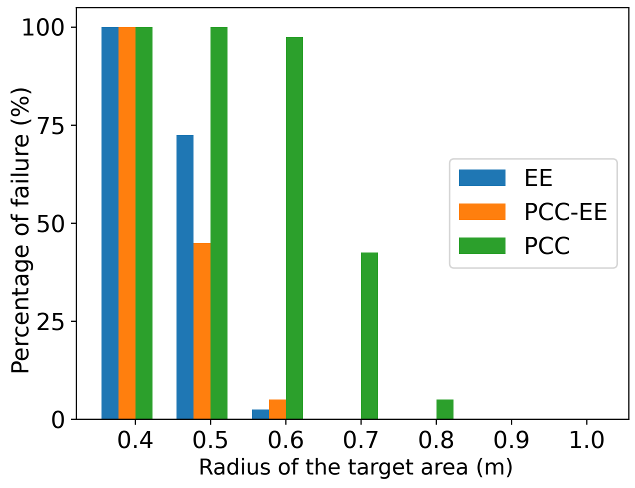


(a) Holonomic

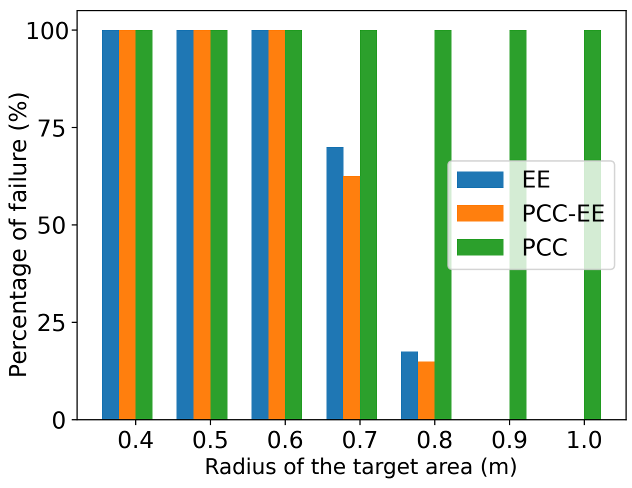


(b) Non-holonomic

Figure 20: Total simulation time comparison of the algorithms for a number of robots from 20 to 300 in steps of 20.



(a) Holonomic.



(b) Non-holonomic.

Figure 21: The number of runs that fail to complete in a simulation time less than 20 minutes in relation to the target size and algorithm for holonomic and non-holonomic robots. SQF had zero percentage of failure in all runs.

V. Graciano Santos, L. Chaimowicz, Hierarchical congestion control for robotic swarms, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 4372–4377. doi:10.1109/IROS.2011.6094640.

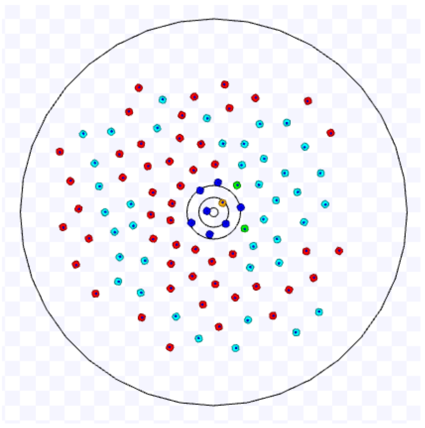
Z. Yan, N. Jouandeau, A. A. Cherif, A survey and analysis of multi-robot coordination, International Journal of Advanced Robotic Systems 10 (2013). doi:10.5772/57313.

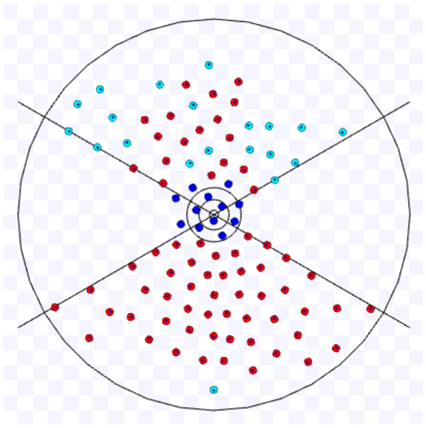
D. Grossman, Traffic control of multiple robot vehicles, IEEE Journal of Robotics and Automation 4 (1988) 491–497. doi:10.1109/56.20433.

L. S. Marcolino, L. Chaimowicz, No robot left behind: Coordination to overcome local minima in swarm navigation, in: 2008 IEEE International Conference on Robotics and Automation, IEEE, Pasadena, CA, USA, 2008, pp. 1904–1909. doi:10.1109/ROBOT.2008.4543485.

M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, A. L. Christensen, Evolution of collective behaviors for a real swarm of aquatic surface robots, PLOS ONE 11 (2016) 1–25. doi:10.1371/journal.pone.0151834.

F. Ducatelle, G. A. D. Caro, C. Pinciroli, F. Mondada, L. Gambardella, Communication assisted navigation


(a) PCC. Available on https://youtu.be/kLOLOENvnqU.


(b) PCC-EE. Available on https://youtu.be/UvzSqFyXB7E.

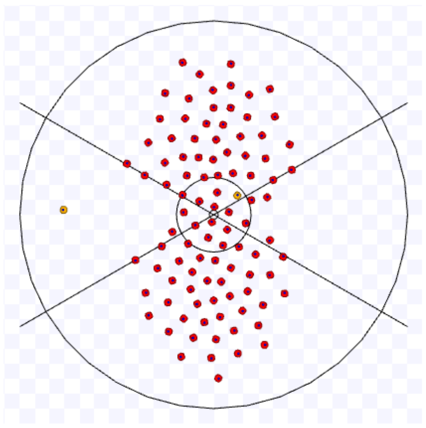
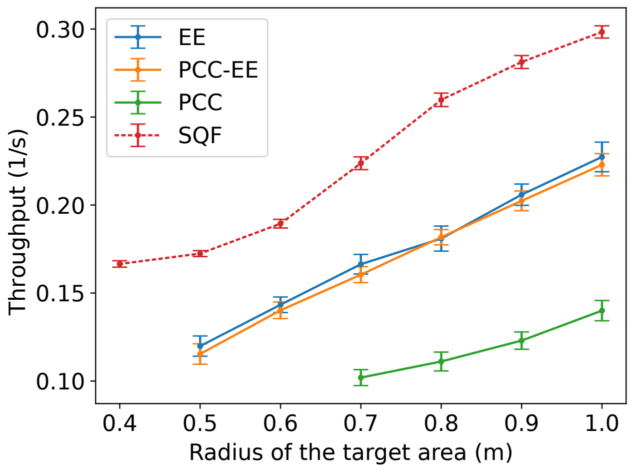
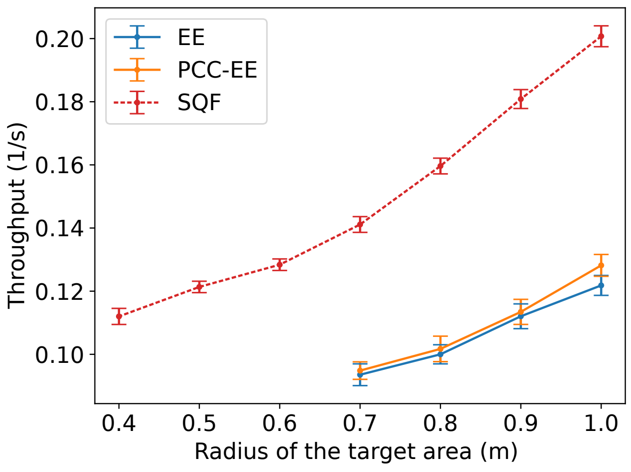

(c) EE. Available on https://youtu.be/BuTcsBNGCag.

Figure 22: Executions with non-holonomic robots and $s = 0.3$ m showing situations where the robots cannot proceed within the time limit of 20 minutes. Here the colours have the same meaning as in (Marcolino et al., 2017). The robots circle around the target area for too long time.



(a) Holonomic



(b) Non-holonomic

Figure 23: Throughput comparison of the algorithms for a varying circular target area radius.

in robotic swarms: Self-organization and cooperation, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 4981–4988. doi:10.1109/IROS.2011.6094454.

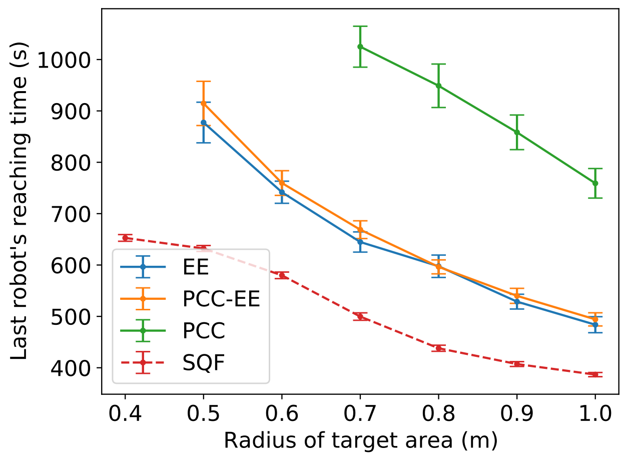
R. Fujisawa, G. Ichinose, S. Dobata, Regulatory mechanism predates the evolution of self-organizing capacity in simulated ant-like robots, Communications Biology 2 (2019) 25. doi:10.1038/s42003-018-0276-3.

S. Batra, Z. Huang, A. Petrenko, T. Kumar, A. Molchanov, G. S. Sukhatme, Decentralized control of quadrotor swarms with end-to-end deep reinforcement learning, in: A. Faust, D. Hsu, G. Neumann (Eds.), Proceedings of the 5th Conference on Robot Learning, volume 164 of *Proceedings of Machine Learning Research*, PMLR, 2022, pp. 576–586.

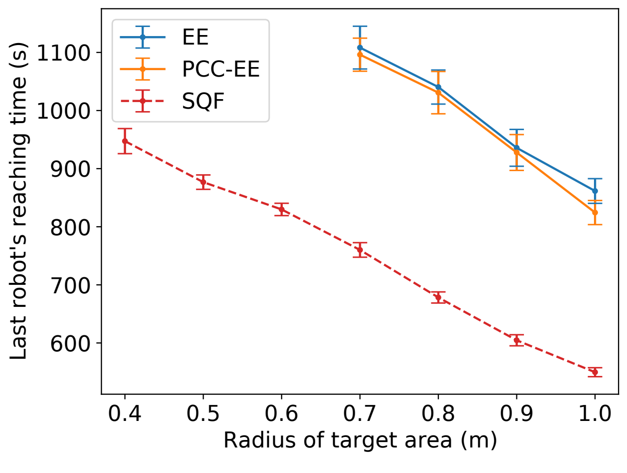
N. Majcherczyk, A. Jayabalan, G. Beltrame, C. Pinciroli, Decentralized connectivity-preserving deployment of large-scale robot swarms, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 4295–4302. doi:10.1109/IROS.2018.8594422.

U. Borrmann, L. Wang, A. D. Ames, M. Egerstedt, Control barrier certificates for safe swarm behavior, IFAC-PapersOnLine 48 (2015) 68–73. doi:10.1016/j.ifacol.2015.11.154, analysis and Design of Hybrid Systems ADHS.

L. S. Marcolino, Y. T. Passos, A. A. F. d. Souza, A. d. S. Rodrigues, L. Chaimowicz, Avoiding target congestion on the navigation of robotic swarms, Autonomous Robots 41 (2017) 1297–1320. doi:10.1007/s10514-016-9577-x.



(a) Holonomic



(b) Non-holonomic

Figure 24: Simulation time comparison of the algorithms for a varying circular target area radius.

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

Y. T. d. Passos, X. Duquesne, L. S. Marcolino, On the throughput of the common target area for robotic swarm strategies, Mathematics 10 (2022) 2482. doi:10.3390/math10142482.

B. P. Gerkey, R. T. Vaughan, A. Howard, The Player/Stage project: Tools for multi-robot and distributed sensor systems, in: Proceedings of the 11th International Conference on Advanced Robotics, ICAR, 2003, pp. 317–323.

S. Kato, S. Nishiyama, J. Takeno, Coordinating mobile robots by applying traffic rules, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 1992, pp. 1535–1541. doi:10.1109/IROS.1992.594218.

P. Caloud, W. Choi, J.-C. Latombe, C. Le Pape, M. Yim, Indoor automation with many mobile robots, in: Proceedings of the IEEE International Workshop on Intelligent Robots and Systems, IROS, 1990, pp. 67–72. doi:10.1109/IROS.1990.262370.

U. Masud, F. Jeribi, M. Alhameed, A. Tahir, Q. Javaid, F. Akram, Traffic congestion avoidance system using foreground estimation and cascade classifier, IEEE Access 8 (2020) 178859–178869. doi:10.1109/ACCESS.2020.3027715.

S. Hoshino, H. Seki, Multi-robot coordination for jams in congested systems, Robotics and Autonomous Systems 61 (2013) 808–820. doi:10.1016/j.robot.2013.04.011.

S. Hoshino, Multi-robot coordination methodology in congested systems with bottlenecks, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2011, pp. 2810–2816. doi:10.1109/IROS.2011.6094438.

D. V. K. Viswanath, K. Madhava Krishna, Towards load-balanced de-congested multi-robotic agent traffic control by coordinated control at intersections, Intelligent Service Robotics 2 (2009) 81–93. doi:10.1007/s11370-009-0035-x.

Y. Zhou, H. Hu, Y. Liu, Z. Ding, Collision and deadlock avoidance in multirobot systems: A distributed approach, IEEE Transactions on Systems, Man, and Cybernetics: Systems 47 (2017) 1712–1726. doi:10.1109/TSMC.2017.2670643.

M. Saska, D. Hert, T. Baca, V. Kratky, T. Nascimento, Formation control of unmanned micro aerial vehicles for straitened environments, Autonomous Robots 44 (2020) 991–1008. doi:10.1007/s10514-020-09913-0.

M. Yoshimoto, T. Endo, R. Maeda, F. Matsuno, Decentralized navigation method for a robotic swarm with nonhomogeneous abilities, Autonomous Robots 42 (2018) 1583–1599. doi:10.1007/s10514-018-9774-x.

D. Carlino, S. D. Boyles, P. Stone, Auction-based autonomous intersection management, in: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 2013, pp. 529–534. doi:10.1109/ITSC.2013.6728285.

G. Sharon, P. Stone, A protocol for mixed autonomous and human-operated vehicles at intersections, in: G. Sukthankar, J. A. Rodriguez-Aguilar (Eds.), Autonomous Agents and Multiagent Systems - AAMAS 2017 Workshops, Best Papers, volume 10642 of *Lecture Notes in Artificial Intelligence*, Springer International Publishing, New York, 2017, pp. 151–167. doi:10.1007/978-3-319-71682-4_10.

G. Sharon, M. W. Levin, J. P. Hanna, T. Rambha, S. D. Boyles, P. Stone, Network-wide adaptive tolling for connected and automated vehicles, Transportation Research Part C: Emerging Technologies 84 (2017) 142–157. doi:10.1016/j.trc.2017.08.019.

G. Sharon, M. Albert, T. Rambha, S. Boyles, P. Stone, Traffic optimization for a mixture of self-interested and compliant agents, in: Proceedings of the 32nd Conference on Artificial Intelligence, AAAI, 2018, pp. 1202–1209. doi:10.1609/aaai.v32i1.11444.

A. Cenedese, C. Favaretto, G. Occioni, Multi-agent swarm control through kuramoto modeling, in: 2016 IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 1820–1825. doi:10.1109/CDC.2016.7798529.

A. Ma, M. Ouimet, J. Cortés, Hierarchical reinforcement learning via dynamic subspace search for multi-agent planning, Autonomous Robots 44 (2020) 485–503. doi:10.1007/s10514-019-09871-2.

E. Sahin, Swarm robotics: From sources of inspiration to domains of application, in: E. Sahin, W. M. Spears (Eds.), Swarm Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 10–20. doi:10.1007/978-3-540-30552-1_2.

E. Sahin, S. Girgin, L. Bayindir, A. E. Turgut, Swarm Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 87–100. doi:10.1007/978-3-540-74089-6_3.

J. C. Barca, Y. A. Sekercioglu, Swarm robotics reviewed, Robotica 31 (2013) 345–359. doi:10.1017/S026357471200032X.

M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 (2013) 1–41. doi:10.1007/s11721-012-0075-2.

L. Bayındır, A review of swarm robotics tasks, Neurocomputing 172 (2016) 292–321. doi:10.1016/j.neucom.2015.05.116.

S. Chung, A. A. Paranjape, P. Dames, S. Shen, V. Kumar, A survey on aerial swarm robotics, IEEE Transactions on Robotics 34 (2018) 837–855. doi:10.1109/TRO.2018.2857475.

M. Hoy, A. S. Matveev, A. V. Savkin, Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey, Robotica 33 (2015) 463–497. doi:10.1017/S0263574714000289.

H. S. Hewawasam, M. Y. Ibrahim, G. K. Appuhamillage, Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments, IEEE Open Journal of the Industrial Electronics Society 3 (2022) 353–365. doi:10.1109/OJIES.2022.3179617.

E. Ferrera, J. Capitán, A. R. Castaño, P. J. Marrón, Decentralized safe conflict resolution for multiple robots in dense scenarios, Robotics and Autonomous Systems 91 (2017) 179–193. doi:10.1016/j.robot.2017.01.008.

R. Siegwart, I. R. Nourbakhsh, Introduction to Autonomous Mobile Robots, Bradford Company, Scituate, MA, USA, 2004.

D. Nelson, D. Barber, T. McLain, R. Beard, Vector field path following for small unmanned air vehicles, in: 2006 American Control Conference, 2006, pp. 5788–5794. doi:10.1109/ACC.2006.1657648.

A. D. Luca, G. Oriolo, Local incremental planning for nonholonomic mobile robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 1994, pp. 104–110. doi:10.1109/ROBOT.1994.351003.