Robot Line Formation

by

Francisco Porto Guerra e Vasconcelos

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
2009

ABSTRACT

Title of Thesis: Robot Line Formation

Francisco Proto Guerra e Vasconcelos, Master of Science, 2009

Thesis directed by: Tim Oates, Associate Professor
Department of Computer Science and
Electrical Engineering

Pattern formation is one of the main research topics in swarm robotics. Its goal is to control the motion of a set of robots in order to form the desired shape. In this thesis, a scalable solution is proposed for the particular case of line formation of differential drive robots, focused on minimizing formation time. The problem is divided into three steps: line formation, target assignment and navigation. The first step consists of choosing the specific line the robots should form, and principal components analysis is used to minimize perpendicular offsets of robot positions to the line. The second step is an assignment problem of N robots to N target positions along the chosen line, and an optimal solution is proposed to minimize formation time, assuming no collisions. The third step is the navigation algorithm that make the robots go from their position to their assigned target, and different algorithms are presented.

Simulations with different setups are done to compare the navigation algorithms and to observe how the target assignment algorithm behaves in a less idealistic scenario, where collisions have to be avoided.

ACKNOWLEDGMENTS

I would like to thank, in the first place, to my advisor Dr. Tim Oates, for all his support during the five months I worked on this thesis, that helped me to find the right research directions.

I would like to thank Dr. João Sousa for his helpful sugestions.

The CORAL lab and all its members created a great research environment, that motivated my work, and greatly developed my interest on Artificial Intelligence.

This thesis is also the result of an exchange program between FEUP (Faculdade de Engenharia da Universidade do Porto) and UMBC (University of Maryland, Baltimore County), that would not be possible without the commitment of Dr. António Moreira, Dr. Arlene Wergin and Dr. Sebastião Feyo.

Finally, I am grateful to my parents for all their support while I was far away from home.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Swarm robotics

Swarm robotics is a field that studies methods for controlling multi-robot systems. A swarm is generally made of simple robots that can generate high level behaviors through cooperation. Much of the work in this subject is inspired by the social behavior of insects, such as ants [11] or bees [12].

A swarm of robots is desired to be robust, scalable and flexible:

- In a **robust** swarm, the failure of single robots does not compromise the success of the collective task.
- In a **scalable** swarm, the number of robots can change without compromising the success of the collective task.
- A **flexible** swarm adapts its strategy when the environment changes.

L. Bayindir and E. Sahin [13] defined a taxonomy to classify research in swarm robotics. According to this taxonomy the problems in swarm robotics can be classified in the following categories: pattern formation, aggregation, chain formation, self-assembly, coordinated movement, hole avoidance, foraging and self-deployment. This paper focuses only on the pattern formation problem.

1.2 Pattern formation

Pattern formation is one of the main areas of research in swarm robotics. Its goal is to distribute a set of robots so that they form the desired shape.

Pattern formation problems can be divided in two kinds of approaches: centralized and distributed. In a centralized approach, there is a unit that collects the information from all robots, and plans the formation according to a wide view of the whole system. In a distributed approach, each robot has limited knowledge of the surrounding environment, and is guided by generic rules, while the overall system converges to some shape.

Generally, a centralized approach leads to more precise formations within a shorter period of time, but requires more sophisticated robot communications and more complex algorithms. In a distributed approach the robots are simpler, making the swarm easier to implement.

Many of the pattern formation algorithms use a virtual potential field as a control strategy. In this approach, the potential field is a combination of attractive forces (originated by the positions in the desired formation) and repulsive forces (originated by obstacles). Applications of this concept can be found in [18], with satellites, and in [17], with non-holonomic ground vehicles.

1.3 Line Formation

Most line formation research comes from particular cases of general pattern formation algorithms. There are others that use a distributed approach in which each robot computes a line equation based on the other robot positions (all of them, or only the ones in its vicinity), and moves in its direction [23].

The most influential line formation work for this thesis was done by A. Feldman [1]. He divided the line formation problem into the following steps:

1. Line formation
2. Target assignment
3. Navigation algorithm

The goal of the first step is to define a convenient set of target positions, given the initial distribution of the robots. Linear regression was used to find a line equation with the form $y = mx + b$. The target positions are points on that line, separated by a constant distance.

The target assignment decides which robot goes to each target, based on robot and target positions. Seven different algorithms were compared empirically through simulation.

The navigation algorithm guides the robots from their initial position to the assigned target positions, using the virtual force field method, which is suited for avoiding static obstacles. Since robots can be considered moving obstacles to each other, some problems arise, resulting in collisions.

In this work the existence of an optimal solution for the target assignment problem was sugested. It was also suggested that better results will be obtained if linear regression was computed with perpendicular offsets, instead of vertical offsets. Both suggestions will be studied and refined in this thesis.

1.4 Assignment Problems

In a centralized pattern formation approach, deciding which robot goes to each target can be done by solving an assignment problem.

In an assignment problem, the objective is to assign N robots to N targets in a way that minimizes some cost function. Each one of the N assignments has an associated cost that is used as input to compute the cost function.

$$\phi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}$$

$$X_\phi = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

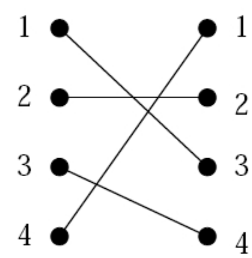


Figure 1.1: Different ways to represent an assignment

An assignment can be mathematically modeled as a permutation of N elements, which can be visualized in different ways, including bipartite graphs, their adjacency matrices, or permutation matrices (Figure 1.1) [21].

Assuming the assignment of N robots to N targets, where w_1, w_2, ..., w_N are the costs of each assigned path, the most common ways to define the cost function $f(w_1, w_2, ..., w_N)$ are the following:

- Total cost: $\sum_{i=1}^{N} w_i$ (linear sum assignment problem)
- Maximum cost: $max(w_1, w_2, ..., w_N)$ (bottleneck assignment problem)

In 1955, H. Kuhn proposed a solution to minimize the total cost, using the Hungarian algorithm [14].

Several solutions were proposed to the bottleneck assignment problem, the first one was in 1956 [15]. And U. Derigs, in 1983, proposed two more efficient algorithms [16]. A solution to this problem will be used in this thesis to minimize formation time.

1.5 Problem Definition

This thesis proposes a scalable solution to the line formation problem, with a centralized approach, focused on minimizing formation time.

The problem is divided into three steps: line formation, target assignment, and navigation. In line formation, a specific line is chosen given the initial distribution of robots. Target assignment decides to which particular position on the line each robot should go. And navigation controls the trajectory of each robot to its assigned target. Each step will be studied separately in a different chapter.

The line formation method proposed in this thesis has its main improvement in relation to prior centralized approaches by using the optimal solution to the bottleneck assignment problem adapted to differential drive robots.

Chapter 2

Robot Model

Through out this thesis, the robots will be considered equal to the model presented in this section, unless stated otherwise.

2.1 Motion Control

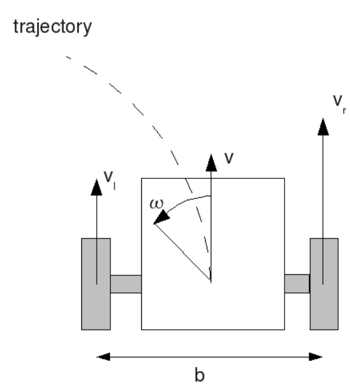


Figure 2.1: Differential drive robot

Differential drive robots are considered, which means that their movement is controlled by two independently driven wheels. As can be seen in Figure 2.1, the velocities of the two wheels (v_l and v_r) are used as inputs to control tangential velocity (v) and angular velocity (ω), according with the following equations:

$$v = \frac{v_r + v_l}{2} \tag{2.1}$$

$$\omega = \frac{v_r - v_l}{b} \tag{2.2}$$

If v_l and v_r are equal, the robot moves in a straight line, with $v = v_l = v_r$ and $\omega = 0$. If v_l and v_r are symmetric, then $v = 0$ and the robot rotates in place. The normal velocity is allways equal to zero, and therefore the robot is not holonomic.

2.2 Range Sensors

The robots are equipped with laser ranging sensors, that allow them to detect and locate obstacles within a range of 180 degrees and 8 meters. The output of these sensors will be used as input to collision avoidance algorithms.

2.3 Communication

It is assumed that robots can exchange data via wireless communication.

To compute target positions, and a target assignment, some processing unit must know the initial positions of all robots. In addition, one of the navigation algorithms requires each robot to know information about other robots close to it.

Chapter 3

Line Formation

In order to find a set of N target positions within a line, the following steps are required:

1. Choose a target line

2. Choose N target positions along the target line

3.1 Target Line

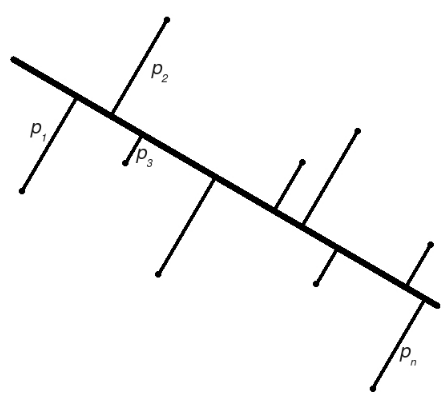


Figure 3.1: Perpendicular offsets

Given a set of robots distributed randomly over some bounded area, the target line should be as convenient as possible to improve the efficiency of the line formation. A. Feldman [1] used linear regression to find the target line, considering vertical offsets. However, this method gives different results depending on the chosen axis of reference. In order for the linear regression be axis independent, perpendicular offsets will be considered. In this case, the linear regression consists in finding the line that minimizes the squared sum of distances p_1, p_2, ..., p_n (Figure 3.1). The solution to this problem [6, 7] is a line with the following properties:

- Goes through the centroid of robot positions
- The orientation is given by the first principal component of the 2-D dataset composed by the coordinates of robots.

3.2 Target Positions

The target positions are distributed symmetrically in relation to the centroid, with a constant distance between them.

If N targets, with centroid in (x, y), are to be distributed along the line that makes an angle of θ with the horizontal axis, with a spacing d between them, the following equations can be used to find the positions (x_1, y_1), (x_2, y_2), ... , (x_N, y_N):

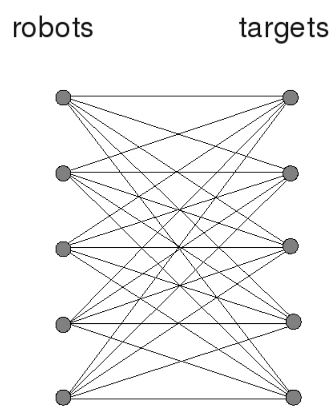
$$x_i = x - \frac{(N-1) \times d \times \cos\theta}{2} + i \times d \times \cos\theta \tag{3.1}$$

$$y_i = y - \frac{(N-1) \times d \times \sin\theta}{2} + i \times d \times \sin\theta \tag{3.2}$$

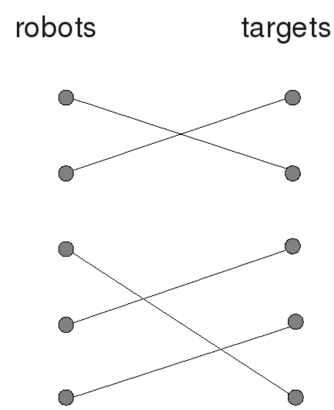
Chapter 4

Target Assignment

4.1 Bipartite Graphs



(a) Complete bipartite graph



(b) Perfect matching

Figure 4.1: Representation of the assignment problem with graphs

The problem of assigning N robots to N targets can be represented by a complete bipartite graph $G((R,T),E)$, where R (set of robots) and T (set of targets) are two disjoint sets, each with N vertices, and E is a set of N^2 edges connecting each vertex in R to each vertex in T, corresponding to all possible paths from each robot to each target (Figure 4.1a).

The solution to this problem can be given by a matching, that is a subset of E in which all edges have no common vertices . If a matching obtained from graph G is a set of N edges, every vertex in R is connected to exactly one vertex in T, and it is called a perfect matching (Figure 4.1b).

There is a total of $N!$ different perfect matchings that can be made out of the N vertices from each set. Each one represents a different assignment of robots to targets. To decide which matching should be chosen, a weighted bipartite graph is used. In this graph each edge has a value, and consequently each matching has a set of edge values, that can be used as inputs of a cost function. The best perfect matching is the one that minimizes this cost function.

There are several ways to attribute values to edges and to build cost functions, depending on what goal should be achieved. In this thesis is discussed the minimization of formation time.

4.2 Formation Time

The formation time is defined as the period that begins with assigning the robots to targets and ends when all robots are located at them, and is equal to the time required for the last robot to arrive at its target.

In this case, an edge (r_i, t_j) will have its weight determined by the time that robot r_i needs to reach target t_j, and the cost function of each matching will be equal to the maximum edge value it contains.

To determine the edge weights, some trajectory between the robot position and the target position must be assumed. The ideal trajectory would be a straight line between the two positions, but since differential motion robots are considered, this might not be possible, because they are not holonomic. They first need to turn until the correct bearing is achieved, and then they can move in a straight line to the target.

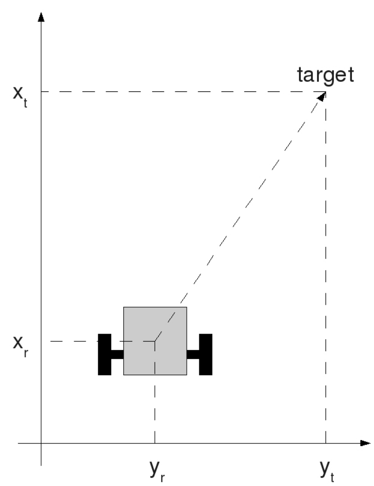


Figure 4.2: An edge is represented by the path connecting a robot position to a target position

The time required to perform this trajectory is $t_t + t_l$, where t_t is the turning time and t_l is the linear movement time. Assuming that the linear movement velocity is v meters per second, and that the rotational velocity is ω radians per second, the time that robot in Figure 4.2 needs to reach the target is equal to:

$$t = t_t + t_l = \sqrt{(y_t - x_t)^2 - (x_t - x_r)^2} \times v + atan2(y_t - y_r, x_t - x_r) \times \omega \qquad (4.1)$$

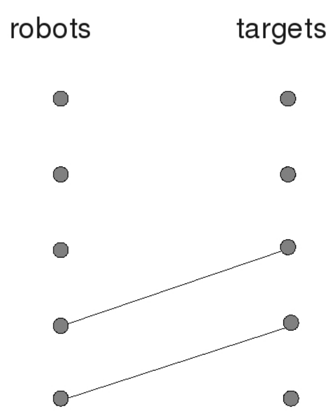
If the robots were holonomic instead, the robots wouldn't need to turn first, so the edge weights would be equal to $t_l = \sqrt{(y_t - x_t)^2 - (x_t - x_r)^2} \times v$, and if v is constant for all robots the edge weights could be equal to the distance between the associated robot and the target.

4.3 Maximum Matching Algorithm

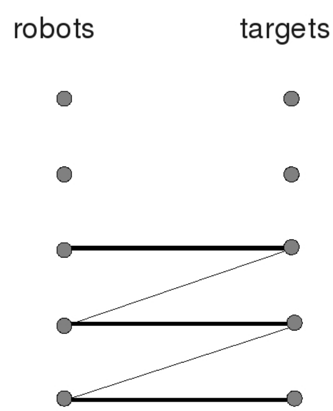
A maximum matching of a given graph is a matching that contains the maximum possible number of edges. A perfect matching is always maximum, which implies that if a maximum matching of a graph G is not perfect, then there are no perfect matchings in G.

The algorithm to find a maximum matching [2] is based on the notion of augmenting path. Given a matching M, an augmenting path is a sequence of edges with the following characteristics:

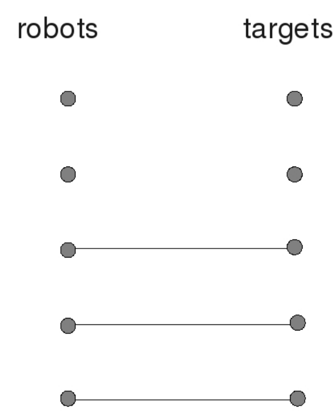
- The first and the last edges don't belong to M
- Edges alternate between belonging to M and not belonging to M



(a) Matching with 2 edges



(b) Augmenting path



(c) Matching with 3 edges

Figure 4.3: Using augmenting paths to find a matching with more edges

Given a matching M with N edges and an augmenting path containing the set of edges

S, the set $M \setminus S$ is a matching with $N + 1$ edges (Figure 4.3). A matching is maximum if and only if it is impossible to find an augmenting path.

The maximum matching algorithm follows these steps:

1. Start with an arbitrary matching
2. Try to find an augmenting path for that matching, using breadth first search.
3. If a path was found, use it to find a matching with one more edge and repeat step 2
4. If a path was not found, terminate and return the last matching

Given n robots ($2n$ vertices and n^2 edges), a breadth first search through all edges runs in $O(n^2)$, and the full algorithm runs in $O(n^3)$. This can be improved to $n^{2.5}$ using the Hopcroft-Karp algorithm [3], that finds more than one augmenting path each iteration.

4.4 Minimum Formation Time

One way to find the assignment with minimum formation time is to determine which edge values are an upper bound on the maximum value of at least one perfect matching, and find the minimum value of that smaller group of edges, which will be equal to the cost function result.

Checking if an edge can be an upper bound on the maximum value of a perfect matching can be done by assuming it is true, removing from the graph all edges with higher values, and then verifying if it is possible to find a perfect matching with the remaining edges, using a maximum matching algorithm.

The following properties can be used to optimize the search for the minimum formation time assignment:

1. If an edge value v is an upper bound on the maximum value of at least one perfect matching, then it is also an upper bound on the minimum formation time.

2. If an edge value v is not an upper bound on the maximum value of all perfect matchings, then it is a lower bound on the minimum formation time.

3. Given N robots and targets, the $(N-1)^{th}$ highest edge value is an upper bound on the minimum formation time.

4. Given N robots and targets, the $(N-1)^{th}$ lowest edge value is a lower bound on the minimum formation time.

Properties 1 and 2 allow to use a binary search to find the optimal solution, and properties 3 and 4 allow to discard a maximum number of $2(N-1)$ edges from the binary search.

This algorithm runs in $O(n^3 log(n^2))$ if it uses the simpler version of maximum matching algorithm, and $O(n^{2.5} log(n^2))$ if it uses the Hopcroft-Karp algorithm.

4.5 Target Assignment Algorithm

The algorithm described before only assigns one robot to a target. To complete a perfect matching, all the other robots must have a target assigned. Since only formation time is taken into account, after the robot that needs the maximum amount of time to reach a target is assigned, in theory, it doesnt matter to which target the other robots will go, as long as they are assigned to a shorter path than the first one. To build an algorithm that assigns a target to the other robots, other factors have to be considered.

Given a graph $G((R,T),E)$ with N robots and N targets, an edge e_{MAX} connecting robot r_i and target t_j, with weight t_{MAX}, can be obtained with the minimum formation time algorithm. Given that robot r_i is assigned to target t_j, these vertices can now be removed from G, turning this graph into a representation of a different target assignment problem, with $N-1$ robots and targets. The same algorithm can be used again, assigning one more robot to a target, and repeated until a perfect matching is made.

Proceding this way, all the chosen paths minimize formation time, given the remaining choices.

This optimal algorithm is completely independent from the target positions. Although only the line formation case is studied in detail, this algorithm is valid for any kind of pattern formation.

4.6 Minimum Formation Time: A Different Approach

Another algorithm to find the minimum formation time assignment was suggested by A. Feldman [1].

Given a complete bipartite weighted graph, in each iteration the edge with maximum weight is removed until there is a robot or a target with only one remaining edge, connected to a robot r_i and a target t_j, which will be called e_{MAX}. Although the weight of e_{MAX} was said in that work to be equal to the minimum formation time, it will be shown that this is not always the case.

If it is assumed that it is possible to find a perfect matching with the non-removed edges (which can be verified with the maximum matching algorithm), then the following facts follow:

1. The weight of e_{MAX} is the maximum weight in the perfect matching (all the edges with higher weights were removed)

2. Since there is only one remaining path connected to r_i or t_j, it means that e_{MAX} was the edge with the lowest weight connected to either r_i or t_j.

   Now, lets assume that there is an edge e_{TEST} with lower weight than e_{MAX} and that can be the maximum value of a perfect matching M:

3. In M, both r_i and t_j are connected to either e_{TEST} or an edge with a lower weight.

4. Both r_i and t_j are connected to an edge lower than e_{MAX}.

The statements 2. and 4. are incompatible, and prove that the edge e_{TEST} does not exist, making the weight of e_{MAX} equal to the minimum formation time.

However, there are situations in which it is impossible to find a perfect match just with the non-removed edges, and statement 1. is not valid anymore. In these cases, the weight of edge e_{MAX} is just a lower bound to the minimum formation time.

4.7 Collisions

Although the target assignment algorithm assumes that no collisions will occur, it can be used together with a collision avoidance navigation algorithm to produce a robust and complete line formation solution. But every time a robot has to avoid another, it can compromise the optimal solution.

Although it can be difficult while choosing which robot goes to each target to predict how collisions will affect the formation time, there are particular cases that can be identified as problematic.

In Figure 4.4 is one of those cases. Robot 1 might go to target 1 or target 2. Assuming that robots move with linear speed v and rotational speed ω, to go to target 1 it will have to turn an angle of π, and run the distance e_1, which takes a total time of $\pi \times w + e_1 \times v$

Figure 4.4: Problematic scenario

seconds. And to go to target 2 it will have to run distance $d - e_2$, spending $(d - e_2) \times v$ seconds.

If $\pi \times w + e_1 \times v > (d - e_2) \times v$, then it would be preferable for robot 1 to go to target 2. If robot 2 is in the symmetrical situation, it will prefer to go to target 1 instead of target 2. This particular target assignment will result in a collision, and it is highly probable that it would be better if the target assignment was different.

The worst case is when e_1 and e_2 are as close as possible to zero. If $e_1 \rightarrow 0^+$ and $e_1 \rightarrow 0^+$, then the condition for a collision occur is $\pi \times w > d \times v$.

To prevent this situation from happening, it must be that

$$d > \frac{\pi \times w}{v}$$

Chapter 5

Robot Navigation

After assigning each robot to a target, we require an algorithm to guide the robots through their paths. This algorithm will affect the trajectories of the robots, and therefore the efficiency of the line formation.

5.1 Linear Movement

The ideal navigation algorithm would be one that follows the trajectory model used to calculate the edge weights. This trajectory consists in the following steps:

1. Rotate until robot orientation points to target (POINTING)
2. Move in a straight line until target is reached (GOING)
3. Stop at target position (STOPPED)

To perform this movement, the finite state machine from Figure 5.1 was implemented. As can be seen from the diagram, the conventional sequence of events would be to start in the TURNING state, then change to GOING, and finally remain in the STOPPED state.

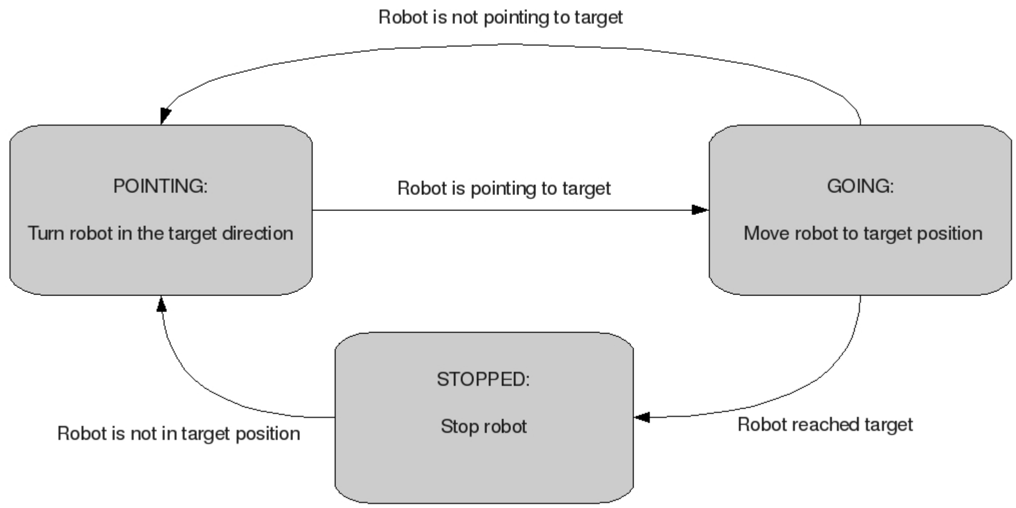


Figure 5.1: Linear movement algorithm

But other state transitions are also present, allowing to go from GOING or STOPPED to TURNING. The reason for this is that in a real scenario, unpredicted errors can occur, making the robot deviate from its path, and if the position or orientation are changed suddenly, the algorithm resets, correcting the trajectory.

This algorithm is only useful as long as there are no collisions. To avoid them, other navigation algorithms must be used, and the robot trajectories will diverge from the ideal constant speed linear movement.

5.2 Nearness Diagram

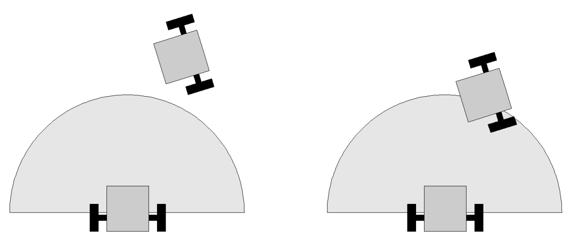
The Nearness Diagram Navigation [4, 5] is a reactive collision avoidance algorithm that uses range sensors information to classify the environment surrounding the robot and decide the best action. The obstacle distribution, robot position and target position are used as inputs to a set of binary criteria, and will result in one of five possible actions.

This algorithm is suited to unknown and dynamic environments, which is the case of the line formation problem, where the obstacles are moving robots.

5.3 Collision Avoidance Algorithms

The developed algorithms for collision avoidance are sets of rules that are checked each sample time to choose if the robot should move with the linear movement algorithm (normal mode) or with the nearness diagram algorithm (avoid mode).

5.3.1 Region Rule



(a) Normal mode (b) Avoid mode

Figure 5.2: Region defined for region rule

A half-circular region around the robot is defined (Figure 5.2). The robot will move with normal mode unless an obstacle is detected in this region, making it switch to avoid mode. In this approach, when no robot is detected in the defined region, it is considered that there is no risk of collision, allowing the robot to move directly to its target.

5.3.2 Priority/Region Rule

The priority/region rule uses two criteria to choose between normal mode or avoid mode. The first is the presence of robots in the inner or outer regions represented in Figure 5.3.

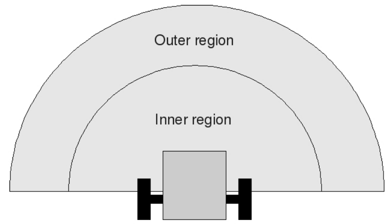


Figure 5.3: Regions defined for priority/region rule

Criteria		Output mode
priority	inner region detection	avoid mode
no priority	inner region detection	avoid mode
priority	outer region detection	normal mode
no priority	outer region detection	avoid mode
-	no detection	normal mode

Table 5.1: Priority/region rule

The second is the priority, when a robot detects another in one of the previously defined regions, the distance to the respective target of both robots defines which one has priority.

The combination of both criteria is used to decide the navigation mode of the robot according to Table 5.1.

Implementing this navigation algorithm requires more communication hardware than the previous one. When one robot detects another, it needs to exchange data with the other robots or with a central computer that gathers all the information in order to determine priority.

Chapter 6

Simulation

6.1 Player/Stage

Simulations were made in Player/Stage [8, 9], a c/c++ open-source simulation platform with a 2-D environment, that is well suited for multi-robot environments. The Player application is a network server that allows control of robot actuators and reading from robot sensors. Although it is primarily designed to interact with real hardware, the Stage plugin module allows to create virtual devices in a 2-D world, populated with robots and obstacles.

An executable file is used to compute all the algorithms, receiving sensor data and transmitting robot commands to player through the interface provided by the library *libplayerc++*.

6.2 Simulation setups

Each simulation starts with a random distribution of robots inside a square room, guaranteeing that they are 1 m away from each other, and 2.5 m away from the walls (Figure 6.1).

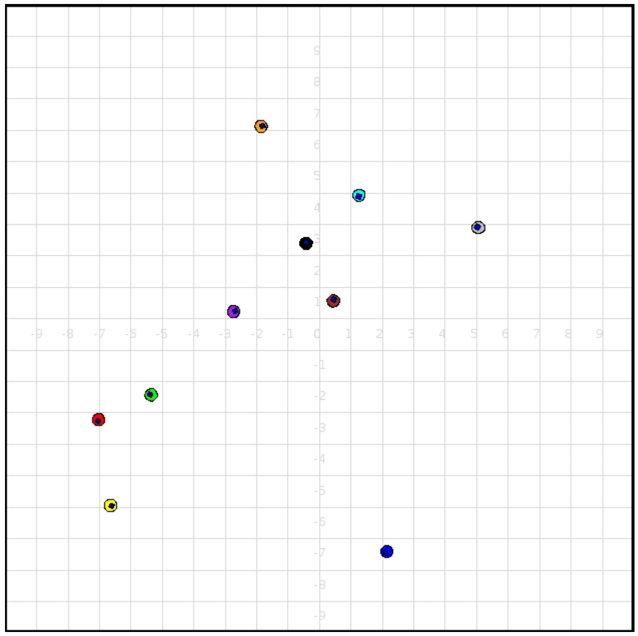


Figure 6.1: Simulation environment

The robots all have the same circular shape, with 0.2 m radius.

There are two configurations that were used, one with a 20 m side room, and distance between targets of 1 m (dense environment), and another one with a 30 m side room, and distance between targets of 1.5 m (sparse environment). Each configuration was simulated with 6, 8, 10, 12 and 14 robots, making a total of 10 setups.

Three different navigation algorithms will be tested 100 times in each of the 10 setups: the priority/region rule, the region rule, and no collision avoidance (linear movement).

For each simulation, after the targets are assigned, and before the robots start moving, the time it would take for each robot to reach its target if it followed the ideal trajectory described by equation 4.1 (t_i) was computed. During the simulation, the time it took for each robot to reach its target (t_s) was measured.

The maximum value of t_i among all robots is equal to the ideal formation time (f_i), and the maximum value of t_s is equal to the simulated formation time (f_s). The ratio $\frac{f_s}{f_i}$

represents how close the simulated trajectories got to the ideal non-collision situation, and it is used to compare the performance of the navigation algorithms.

After each simulation is over, the number of robots that successfully reached their target is also registered. If this number is equal to the total number of robots, the simulation is considered successful, otherwise it is a failure.

For each setup, the values obtained from the 100 simulations are used generate the following overall results:

- Number of failures
- Average f_s
- Average $\frac{f_s}{f_i}$
- Standard deviation of $\frac{f_s}{f_i}$

6.3 Results

Figures 6.2 and 6.3 show two steps of a successfull simulation. In the first one is represented the target line (in red) generated by the principal components method, and in the second are the paths travelled by the robots. It is visible that the robots furthest away from the target line, like the green and the purple ones, tend to go to the nearest point in the line, because they are more critical to minimize the formation time.

6.3.1 Number of failures

In the graphics from Figures 6.4 and 6.5 are presented the total number of failures (in 100 simulations) for each setup and navigation algorithm.

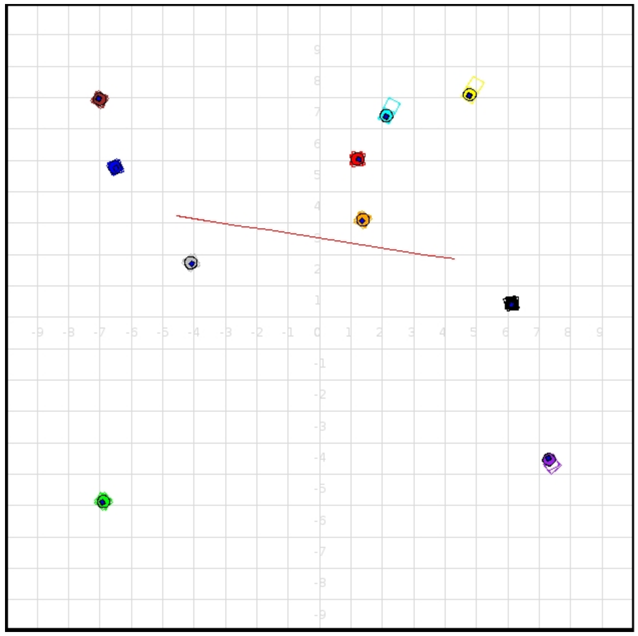


Figure 6.2: Computation of target line in simulation

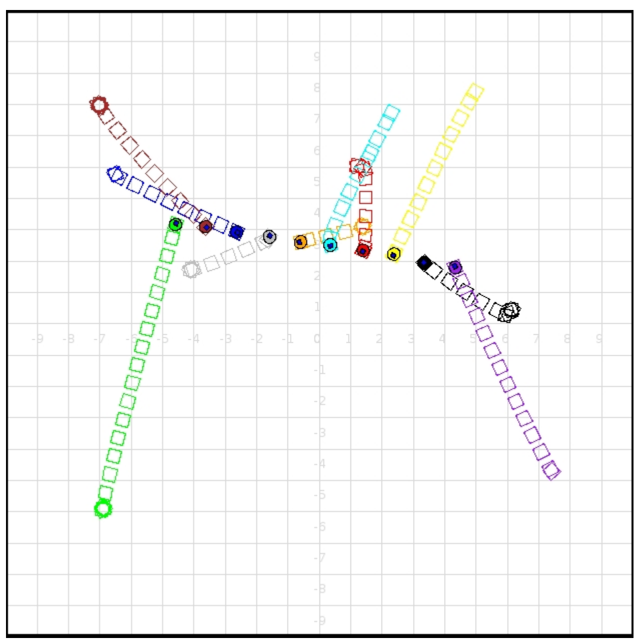


Figure 6.3: Successful simulation

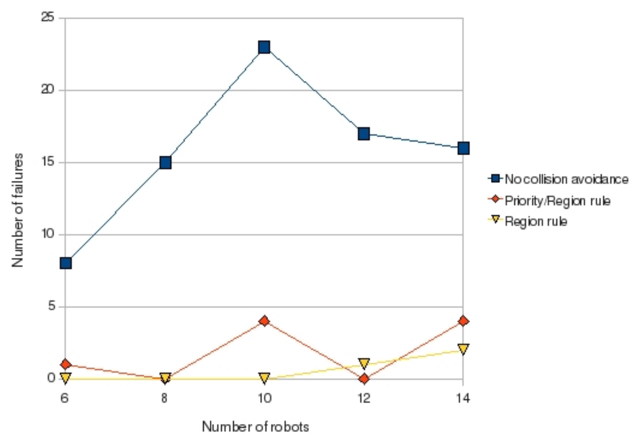


Figure 6.4: Number of Failures: Dense environment

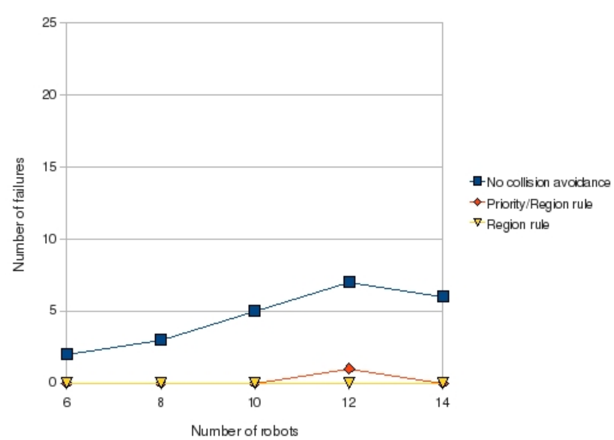


Figure 6.5: Number of failures: Sparse environment

In the sparse environment the number of collisions is lower than in the dense environment, for every navigation algorithm. It is also visible that many more failures occur when no collision avoidance is used (blue values).

Generally, the number of collisions grows when the number of robots increases.

6.3.2 Average formation time

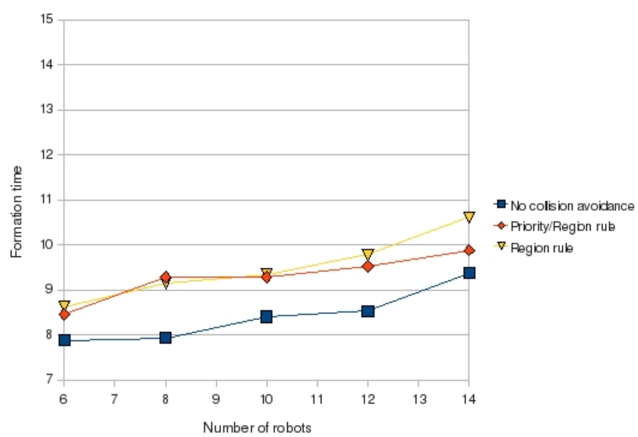


Figure 6.6: Average formation time: Dense environment

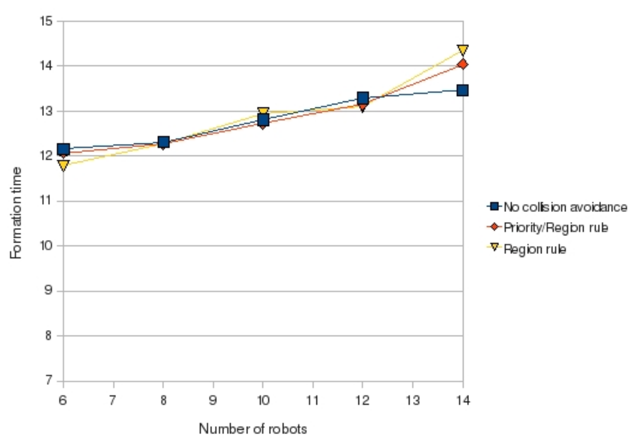


Figure 6.7: Average formation time: Sparse environment

In the graphics from Figures 6.6 and 6.7 is shown the average value of the simulated formation time (f_s) for each setup and navigation algorithm. The simulations that ended in failure were discarded to compute the averages.

There is an evident direct relation between the number of robots and the average formation time, for every navigation algorithm. In the sparse environment f_s values are higher than in the dense environment, and the values of the different navigation algorithms are almost equal.

6.3.3 Average $\frac{f_s}{f_i}$

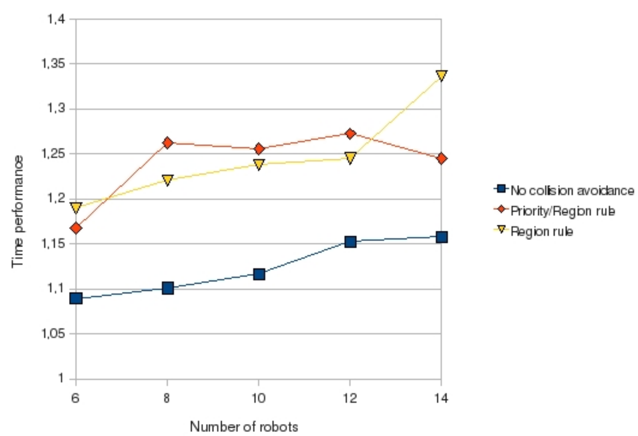


Figure 6.8: Average $\frac{f_s}{f_i}$: Dense environment

In the graphics from Figures 6.8 and 6.9 is shown the average value of $\frac{f_s}{f_i}$ for each setup and navigation algorithm. The simulations that ended in failure were discarded to compute the averages.

The relations between average performance, number of robots and the different navigation algorithms are very similar to the ones observed with average formation time. The main difference is that average $\frac{f_s}{f_i}$ is lower in the sparse environment, while the average formation time was lower in the dense environment.

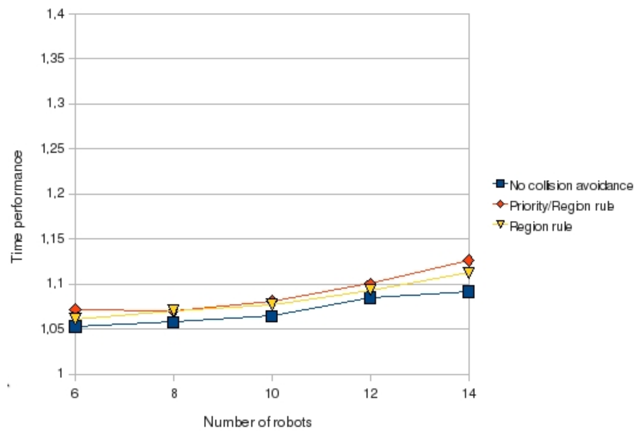


Figure 6.9: Average $\frac{f_s}{f_i}$: Sparse environment

6.3.4 Standard deviation of $\frac{f_s}{f_i}$

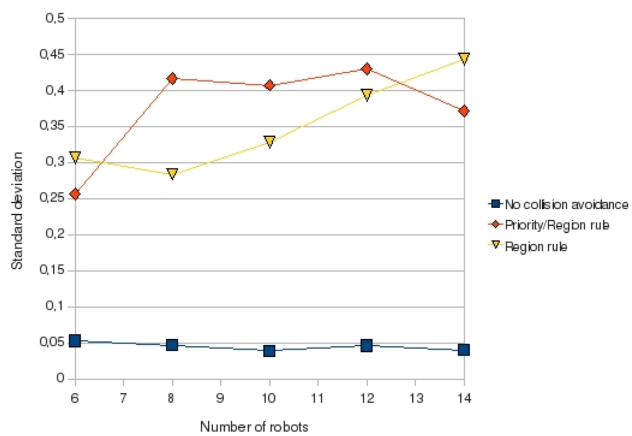


Figure 6.10: Standard deviation of $\frac{f_s}{f_i}$: Dense environment

In the graphics from Figures 6.10 and 6.11 is shown the standard deviation of $\frac{f_s}{f_i}$ for each setup and navigation algorithm. The simulations that ended in failure were discarded.

The standard deviation of $\frac{f_s}{f_i}$ in the sparse environment is significantly lower than in the dense environment, when collision avoidance is used. Without collision avoidance, the standard deviation is very low for every setup.

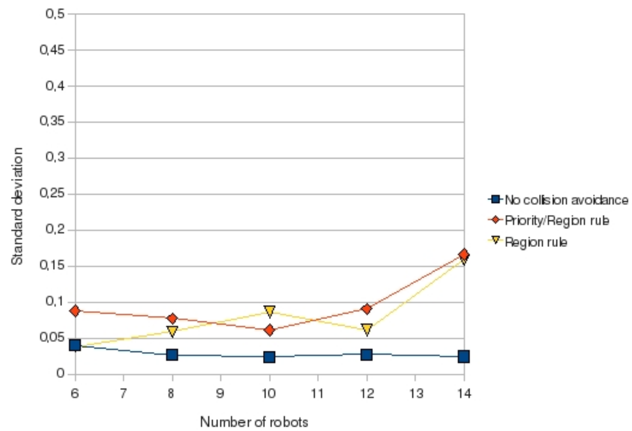


Figure 6.11: Standard deviation of $\frac{f_s}{f_i}$: Sparse environment

6.4 Interpretation of Results

6.4.1 No Collision Avoidance

The blue values in each graph, representing simulation results when no collision avoidance was used, have a special meaning.

In these cases, the simulation failures happen when robots collide before they reach their targets, and so the number of failures is a valid estimation of collision frequency.

In the other overall results (average time formation, average performance, and performance standard deviation), by excluding the failed simulations, collisions never occur, and robots go to their targets with a linear movement. These cases represent the ideal no collision scenario, and the results should be very close to the best that it is possible to achieve.

These particular properties of the values forming the blue curves are the reason behind the disparities when compared with the other curves, in almost every graph.

6.4.2 Sparse Environment vs Dense Environment

Simulations in sparse and dense environments showed different results.

In the dense environment, robots move closer to each other than in the sparse environment, so it is not surprising to find that the collision frequency is higher (Figures 6.4 and 6.5). Because of this higher collision frequency, the simulations using collision avoidance have less predictable behavior, which is shown by a higher standard deviation of the performance in the dense environment (Figures 6.10 and 6.11). And it can also explain why performance is better in the sparse environment (Figures 6.8 and 6.9): with a lower rate of collisions, it is expected that the performance gets closer to an optimal value, that happens when no collisions occur.

On the other hand, in the sparse environment, the robots are generally further away from their assigned targets, resulting in a higher average formation time (Figures 6.6 and 6.7).

6.4.3 Priority/Region Rule vs Region Rule

Comparing the two collision avoidance navigation algorithms, the region rule has generally less failures, and a lower average $\frac{f_s}{f_i}$ than priority/region rule, making it safer and more efficient.

The region rule algorithm typically spends more time during AVOID MODE than the priority/region rule, making it move in a more cautious way, which is confirmed by the lower number of failures. Common sense would say that this extra precaution would also slow down line formation, but this is not confirmed by the results, since the region rule generally has a lower average $\frac{f_s}{f_i}$.

6.4.4 Number of robots

Changing the number of robots has its predominant effect on collision frequency, especially when there are 10 robots or less. With higher numbers of robots, this relation is not as evident. Average formation time and $\frac{f_s}{f_i}$ also grow with the increase of the number of robots, in an almost linear fashion.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The three-step approach used in this work (line formation, target assignment and navigation) proved to be a complete solution to the line formation problem, including path planning and motion control. The steps are completely independent from each other, which was useful to study each one separately, and to compare different navigation algorithms without altering the other steps.

7.1.1 Line formation

The method used to generate the target line always generates a good line. By distributing the targets along the orientation with more variance of the robot distribution, the probability of a collision occurring is low, and by minimizing the squared sum of perpendicular offsets, the distances travelled by the robots are usually short.

7.1.2 Target Assignment

In the sparse environment, given the low frequency of collisions in the simulations, the target assignment algorithm is the optimal solution in the great majority of cases. In the dense environment, the frequency of collision occurrence achieved a maximum of 23% of the simulations, for the 10 robot setup. In this case, there is a considerable number of simulations where the robots might not have been assigned optimally. This shows that the target assignment algorithm, to be accurate, can only assume no collisions below a certain limit of robot density that depends on the room size, line spacing distance, robot size and number of robots.

The inclusion of the turning time in the costs used to compute the target assignment was essecial to produce an accurate solution,

7.1.3 Navigation

The collision avoidance algorithms are essential to this line formation approach. Although the frequency of collisions might not be very high, when many robots are used, the chances that at least two robots collide is considerable, and therefore, this algorithm can only be scalable if collisions are considered as an inevitable part of the problem.

The algorithm with better results in almost every set up was the region rule, which was the most simple, and the one that doesn't require communication between the robots. In addition, in the sparse environment this algorithm was successfull in all 500 simulations, making it very robust.

7.2 Future Work

7.2.1 Line formation

Using principal component analysis to find a target line proved to be usefull, however this method is not optimal, in the sense that minimizes the line formation time, and that opens the possibility of more improvements.

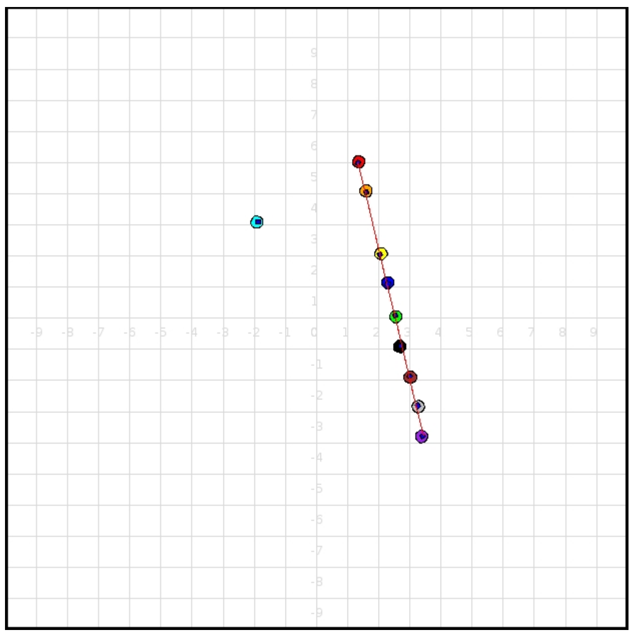


Figure 7.1: Simulation screenshot

In Figure 7.1 is represented a simulation where all robots are stopped at their targets except one, that still has to reach the line formation. It is obvious that some time would be saved if all the robots moved towards the one missing, forming a line more close to it. This situation is frequent because the middle point of the line formation is the centroid of the initial robot distribution, and it can lead to the situation of Figure 7.2, where the particular distribution of robots clearly shows that the formation time is not minimized.

Some improvement might be obtained if the middle point of the line formation was cho-

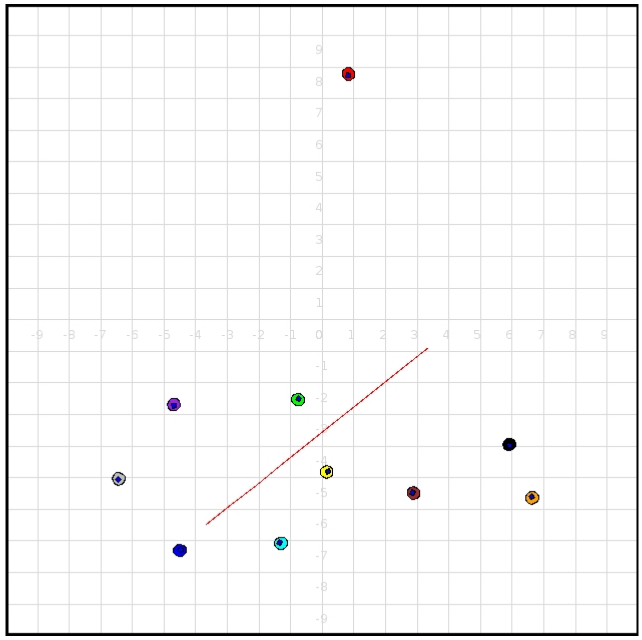


Figure 7.2: Target line computation

sen in a way that minimizes the maximum distance between a robot and the line. This can be obtained by solving the smallest enclosing circle problem [10], which is also equivalent to the 1-center problem.

7.2.2 Navigation Algorithm

There are many other options for collision avoidance navigation that could be used instead of the two proposed algorithms, including coventional reactive algorithms [20], and specific methods for multi-robot systems [19, 22], so there is a great margin of improvement. The most important breakthrough would be to find a failure free algorithm in any setup.

7.2.3 Other Pattern Formations

The target assignment algorithm is valid for any pattern formation, opening the possibility to validate its usefulness in the formation of many other shapes. It would be particularly interesting to analyze in which shape formations collisions are less frequent, making them more suitable for the optimal target assignment algorithm.

Appendix A

APPENDIX

A.1 SIMULATION RESULTS

A.1.1 OVERALL RESULTS

Number of robots	6	8	10	12	14
No collision avoidance	8	15	23	17	16
Priority/Region rule	1	0	4	0	4
Region rule	0	0	0	1	2

Table A.1: Dense environment: Number of failures

Number of robots	6	8	10	12	14
No collision avoidance	7.88	7.93	8.4	8.52	9.37
Priority/Region rule	8.46	9.28	9.29	9.52	9.88
Region rule	8.63	9.15	9.33	9.78	10.61

Table A.2: Dense environment: Average f_s

Number of robots	6	8	10	12	14
No collision avoidance	1.09	1.1	1.12	1.15	1.16
Priority/Region rule	1.17	1.26	1.26	1.27	1.24
Region rule	1.19	1.22	1.24	1.25	1.34

Table A.3: Dense environment: Average $\frac{f_s}{f_i}$

Number of robots	6	8	10	12	14
No collision avoidance	0.05	0.05	0.04	0.05	0.04
Priority/Region rule	0.26	0.42	0.41	0.43	0.37
Region rule	0.31	0.28	0.33	0.39	0.44

Table A.4: Dense environment: Standard deviation of $\frac{f_s}{f_i}$

Number of robots	6	8	10	12	14
No collision avoidance	6	8	10	12	14
Priority/Region rule	0	0	0	1	0
Region rule	0	0	0	0	0

Table A.5: Sparse environment: Number of failures

Number of robots	6	8	10	12	14
No collision avoidance	12.16	12.32	12.82	13.29	13.47
Priority/Region rule	12.07	12.28	12.73	13.16	14.05
Region rule	11.79	12.27	12.94	13.1	14.35

Table A.6: Sparse environment: Average f_s

Number of robots	6	8	10	12	14
No collision avoidance	1.05	1.06	1.06	1.08	1.09
Priority/Region rule	1.07	1.07	1.08	1.1	1.13
Region rule	1.06	1.07	1.08	1.09	1.11

Table A.7: Sparse environment: Average $\frac{f_s}{f_i}$

Number of robots	6	8	10	12	14
No collision avoidance	0.04	0.03	0.02	0.03	0.02
Priority/Region rule	0.09	0.08	0.06	0.09	0.17
Region rule	0.04	0.06	0.09	0.06	0.16

Table A.8: Sparse environment: Standard deviation of $\frac{f_s}{f_i}$

Bibliography

[1] A. Feldman, “Swarm Robotics: Parallelized Line Formation”, MS CMCS Thesis, UMBC, 2008.

[2] Douglas B. West, “Introduction to Graph Theory” (2nd Edition), Prentice Hall, August 2000.

[3] J. Hopcroft and R. Karp, “An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs”, *SIAM J. Comput.*, 2(4):225-231, Dec 1973.

[4] J. Minguez and L. Montano, “Nearness diagram navigation(ND): a new real-time collision avoidance approach”, *in Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and System(IROS)*, 2000.

[5] J. Minguez and L. Montano, “Nearness Diagram Navigation (ND): Collision Avoidance in Troublesome Scenarios”, *IEEE Transactions on Robotics and Automation*, Volume 20, Issue 1, Pages:45 - 59, 2004.

[6] K. Pearson, “On lines and planes of closest fit to systems of points in space”, *Philos. Mag.* 2:559572, 1901

[7] A. Miranda, Y. Le Borgne and G. Bontempi, "New Routes from Minimal Approximation Error to Principal Components", *Neural Processing Letters* Volume 27, Number 3, June 2008

[8] B. Gerkey and contributors, "The Player Robot Device interface", 1999-2007.

http://playerstage.sourceforge.net/doc/Player-2.1.0/player/

[9] R. Vaughan and contributors, "The Stage Robot Simulator", 1998-2008.

http://playerstage.sourceforge.net/doc/stage-3.0.1/

[10] M. Shamos and D. Hoey, "Closest-Point Problems", *Proceedings of the 16th Annual IEEE Symposium on Foundations of Computer Science*, 15 1 - 162, 1975.

[11] S. Mao and J. Yu, "Path Planning Based on Ant Colony Algorithm and Distributed Local Navigation for Multi-Robot Systems", *Proceedings of the 2006 IEEE Conference on Robotics, Automation and Mechatronics*, 2006.

[12] A. Purnamadjaja and R. Russel, "Pheromone communication: Implementation of Necrophoric Bee Behavoiur in a Robot Swarm", *Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics*, 2004.

[13] L. Bayindir and E. Sahin , "A Review of Studies in Swarm Robotics", *Turk J Elec Engin*, VOL.15, NO.2 2007.

[14] H. Kuhn, “The Hungarian Method for the Assignment Problem”, *Naval Research Logistics Quarterly*, 2(1-2):8397, 1955

[15] L. R. Ford and D. R. Fulkerson, “Maximal flow through a network”, *Canadian Journal of Mathematics*, 8, 399404, 1956.

[16] U. Derigs, “Alternate Strategies for Solving Bottleneck Assignment Problems - Analysis and Computational Results”, *Computing*, 33, 95-106, 1984.

[17] G. Elkaim and R. Kelbley, “A Lightweight Formation Control Methodology for a Swarm of Non-Holonomic Vehicles”, *IEEE Aerospace Conference*, 2006

[18] C. Pinciroli, M. Birattari, E. Tuci, M. Dorigo, M. Zapatero, T. Vinko and D. Izzo, “Self-Organizing and Scalable Shape Formation for a Swarm of Pico Satellites”, *NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 57-61, 2008.

[19] F. Kolushev and A. Bogdanov, “Multi-agent Optimal Path Planning for Mobile Robots in Environment with Obstacles”, 2000

[20] A. Chakravarthy and D. Ghose, “Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach”, *IEEE Transactions on Systems, Man, and CyberneticsPart A: Systems and Humans*, VOL. 28, NO. 5, 1998

[21] R. Burkard and E. Cela, “Linear Assignment Problems and Extensions”, 1998

[22] K. Krishna and H. Hexmoor, “Reactive Collision Avoidance of Multiple Moving Agents by Cooperation and Conflict Propagation”, IEEE *Proceedings of the International Conference on Robotics and Automation*, 2004.

[23] O. Albayrak, “Line and Circle Formation of Distributed Autonomous Mobile Robots with Limited Sensor Range.”, MS Electrical Engeneering Thesis, Naval Postgraduate School, 1996