Distributed Formation Control of Networked Mobile Robots in Environments with Obstacles

Whye Leon Seng[†], Jan Carlo Barca[°], and Y. Ahmet Şekercioğlu[†]

[†]Department of Electrical and Computer Systems Engineering
[°]Clayton School of Information Technology
Monash University, Melbourne, Australia
whyeleon.seng@gmail.com, jan.barca@monash.edu, asekerci@ieee.org

Abstract

A distributed control mechanism for ground moving nonholonomic robots is proposed. It enables a group of mobile robots to autonomously manage formation shapes while navigating through environments with obstacles. The formation can be maintained without the need of any inter-robot communication. Obstacle avoidance is designed to be performed by the individual robots themselves. Formation scaling is implemented to ensure the formation shape is maintained for as long as possible. If the formation fails to hold its shape when navigating through environments with obstacles, formation morphing has been incorporated to preserve the interconnectivity of the robots, thus reducing the possibility of losing robots from the formation.

The algorithm has been implemented on a nonholonomic multi-robot system for empirical analysis. Experimental results demonstrate formations completing an obstacle course within 12 seconds with zero collisions. Furthermore, the system is capable of withstanding up to 25% sensor noise.

1 Introduction

In recent years there has been increasing interest in automated control and coordination of multi robot systems. The advancement in technology and increasing safety awareness have pushed for the development of such systems, particularly in tasks where human lives are at stake such as in reconnaissance and search and rescue. Existing solutions are often costly, require monitoring from a base station, and/or only deploy small numbers of robots with limited geographical coverage, therefore severely reducing their effectiveness. One solution to this problem is to introduce a distributed swarm-based system. With multiple identical lesser robots working as a group, a swarm is capable of performing tasks beyond the capabilities of a single robot. Such a system would significantly drive down the cost of deployment as each robot only requires the most basic sensor equipment. It also allows the system to be scalable and possess self-repairing abilities should several robots within the swarm fail. This translates to a much higher reliability and reduced risk of mission failure [1].

Several investigations on this issue have been considered [2, 3]. Shao et al. [2] consider a one-leader constraint formation control, which produces a non-rigid formation control graph.

It employs adjacency and parameter matrices which simplifies the definitions of local leader-follower relationships and the overall formation shape. However, non-rigid formation control does not allow the formation shape to be maintained as the formation moves. The obstacle avoidance method considered is also insufficient in dealing with more complex obstacles and may cause robots to be trapped behind obstacles. Formation control with both one-leader and two-leaders constraints was presented in [3]. Their work focuses on how transitions between formations can be realized via a transition matrix. How control graphs can be classified on the basis of the number of followers with one and two leaders is also investigated. However, as in [2], they do not explain how one can select ideal formation shapes in environments with obstacles.

We have also looked into several obstacle avoidance methods in this research. Obstacles are regarded as virtual leaders in [4, 5], forcing robots to maintain a distance from obstacles, in a manner similar to how the robots maintain formation shapes by keeping a distance from their Local Leaders (LL). The downside is that the formation may no longer be rigid as robots may have to severe connections with their LLs in order to apply the distance constraints for the obstacles. A behaviour-based formation control is proposed in [6], where a swarm of robots navigates about obstacles simply by rotating and scaling the overall formation. However, rotating a formation of nonholonomic robots can be time consuming and inefficient. Kuppan et al. proposes that on detection of an obstacle, the robot is elected as the temporary Formation Leader (FL), allowing it to steer the formation away from the obstacle [7]. The drawback with this approach is that selecting the temporary leaders complicates the control process significantly when the formation detects multiple obstacles from different directions. A semi-rigid obstacle avoidance approach is presented in [8], where the follower is allowed to vary the angle constraint from the leader in order to steer clear of obstacles. The downside with this approach is that robots are unable to tackle obstacles efficiently when they have two or more LLs, with multiple distance constraints to abide by. Another technique is to utilize potential fields where interaction between robots and obstacles are represented by repulsive and attractive forces [9–12]. However, the algorithms only considered the distances of the obstacles in determining the magnitude of repulsive forces, which may cause the robots to slow down excessively even when the obstacles are not necessarily blocking them.

The common problem with the techniques discussed above is that they have not been implemented on real robots [1–6, 9–12], where issues such as sensor noise and kinematics must be considered in order to accurately evaluate the performance of control mechanisms for the robots. The second problem is the negligence of additional constraints posed by nonholonomic robots. The solutions to obstacle avoidance are presented as net force vectors in [10–12], indicating the instantaneous velocity that a robot should take. However, nonholonomic robots are not capable of switching from one state to another arbitrary one immediately. In our work, we have explicitly detailed the algorithm and its output in meaningful physical terms such as velocities and angular velocities, which can be included effortlessly in the control systems of existing robots. We have implemented the control mechanisms on the eBug multi robot system [13] to properly evaluate the algorithm. Parameters for the algorithm were determined more conservatively to minimize the impact of issues such as sensor noise, wheel slippage and a noisy wireless channel.

In Section 2, the proposed formation control method is described. Section 3 introduces the implemented obstacle avoidance techniques, followed by a discussion of formation rebuilding in Section 4. The performance of the algorithm measured from conducted experiments is discussed in Section 5. Section 6 concludes the paper as well as the possible future research

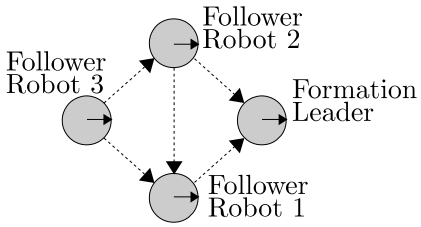


Figure 1: Inter-robot relationships are represented as a directed graph, where the directed edges go from the follower robots to their local leaders.

work that can be done.

2 Formation Control

The proposed formation control algorithm is an improvement on the work presented in [14]. It employs graph theory based formation structures that branches out from a FL to Follower Robots (FR), which are each assigned LLs according to their allocated positions in the formation, or Formation Position (FP).

The inter-robot relationships are described with tree diagrams or directed graphs [2, 15, 16], as shown in Fig. 1. If a path from the root (FL) to every vertex (all the other robots) exists, it implies that all the robots are connected and are part of the formation. The directed edges indicate the direction of the connection to be made. In other words, every FR only needs to know the positions of the LLs relative to itself, in order to maintain the formation. The formation shape is formed by assigning distance and angle constraints from the LLs, to every FP. To maintain the formation shape as the formation moves, rigid graphs are used in the formation structure. A rigid graph can be thought of as a network of agents which are interconnected to one another by rigid bars of length defined by the assigned distance constraints to each agent [17]. To further reduce the complexity of the network, formation structures have been designed to be minimally rigid, such that if any edge is removed from the graph, the formation will no longer be rigid [17]. Anderson et al. state that the number of edges (connections) required is defined by the number of vertices (robots) using

$$n_e = 2n_v \tag{1}$$

where n_e and n_v are the number of edges and number of vertices respectively.

2.1 Formation Position Assignment

Before the formation control system can take effect, the robots have to be assigned to FPs. First, the robot that is the closest to a predefined destination is automatically assigned as the FL, or the first FP. This positions the other robots behind the FL relative to the destination, allowing the other robots to easily get to their assigned FPs as the FL starts moving [9]. Alternatively, if no destination has been predefined, any robot can be elected as FL under the discretion of the user.

Next, the remaining robots are assigned to FPs sequentially, starting with the second FP whose LL is the FL. Robots are assigned to FPs based on their individual character costs, which are the Euclidean distance errors, d_{error} between their current positions and a FP [18].

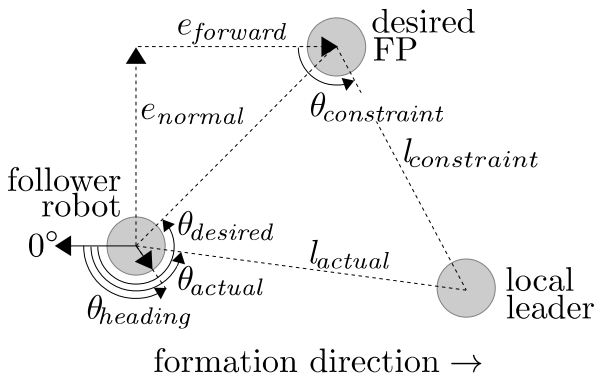


Figure 2: Reducing discrepancy between FR and its FP with respect to LL.

An important improvement over the work by Chen is that we only require local information to calculate the cost, rather than using global coordinates, as seen in Fig. 2. These FPs are determined by predefined distance and angle constraints (which are dependent on the starting formation shape chosen by the user) relative to the corresponding LLs for the FPs. To begin, a robot first detects the distances and angles of the LLs relative to it. If neither LLs can be detected, i.e. beyond the detection range of the FR, R_{scan}, it will not be able to content for this FP. It then calculates the cost to the FP using equations by Barca et al. [14]

$$e_{forward} = l_{constraint}\cos(\theta_{constraint}) - l_{actual}\cos(\theta_{actual}) \tag{2}$$

$$e_{normal} = l_{constraint}\sin(\theta_{constraint}) - l_{actual}\sin(\theta_{actual}) \tag{3}$$

$$d_{error} = \sqrt{e_{forward}^2 + e_{normal}^2} \tag{4}$$

where

$e_{forward}$	:	distance error in direction of formation movement,
e_{normal}	:	distance error in direction perpendicular to formation movement,
$l_{constraint}$	:	required distance between robot and LL for this FP,
l_{actual}	:	actual distance between robot and LL for this FP,
$\theta_{constraint}$	:	required angle from robot to LL for this FP,
θ_{actual}	:	actual angle from robot to LL for this FP, and
d_{error}	:	Euclidean distance error from robot to this FP.

Note that the angles under the formation control section are all measured with respect to the opposite direction of where the formation is heading. This process is performed by every robot that has not been assigned a FP and the FP is finally assigned to the robot with the lowest d_{error} for that FP. If a FP requires two LLs, the robots content for the FP with the average d_{error} from both LLs. The aforementioned steps are then repeated for the next FP, with its associated distance and angle constraints. The assignments of FP are recorded in an adjacency matrix which describes the leader-follower relationships, and a parameter matrix which stores the assigned constraints. These matrices are described in detail by [2, 3].

2.2 Control System

To maintain the shape of the formation as it moves, the FRs first calculates the $e_{forward}$ and e_{normal} to their FPs (Equations 2 and 3). These values are then used by a Velocity

Controller (VC) and an Angular Velocity Controller (AVC) to calculate the required responses for reducing d_{error} between a FR and its FP.

2.2.1 Velocity Controller

Velocities are calculated with a non-linear controller

$$v = \begin{cases} V_{leader} \log_{10}(e_{forward} + 1) \cos(\theta_{desired} - \theta_{heading}) & \text{if } e_{forward} \geq 0 \\ V_{leader} \log_{10}(-e_{forward} + 1) \cos(\theta_{desired} - \theta_{heading}) & \text{otherwise} \end{cases} \tag{5}$$

where

v	:	velocity output,
V_{leader}	:	maximum velocity of formation leader,
$e_{forward}$	:	distance error in direction of formation movement,
$\theta_{desired}$	:	angle of FP measured from FR, and
$\theta_{heading}$	:	heading angle of FR,

to provide rapid convergence towards the desired FP. A cosine multiplier which considers the angle that the FR faces is added to reduce unnecessary movements in the direction perpendicular to formation movement.

2.2.2 Angular Velocity Controller

The AVC has two states, which are

$$\omega = \begin{cases} K_\omega \alpha & \text{if } d_{error} > R_{dzone} \\ K_\omega \left(\beta - (\beta - \alpha) \frac{d_{error}}{R_{dzone}} \right) & \text{otherwise} \end{cases} \tag{6}$$

with

$$\alpha = \theta_{desired} - \theta_{heading} \tag{7}$$

$$\beta = 180^\circ - \theta_{heading} \tag{8}$$

where

$\theta_{desired}$	:	angle from FR to its FP,
$\theta_{heading}$	:	heading angle of FR,
ω	:	angular velocity output,
K_ω	:	angular velocity constant for angular velocity control,
d_{error}	:	Euclidean distance error from FR to the its FP, and
R_{dzone}	:	radius of dead zone.

When the FR is at a R_{dzone} distance away from its FP, the controller steers the FR directly towards the FP. However, as it enters the dead zone, the controller attempts to steer the FR in the direction that the formation is travelling in. This second state is introduced to reduce the oscillations that arise from the tendency of the AVC to overcorrect the heading angle of the FR when it is too close to the FP. R_{dzone} should only be slightly wider than the diameter of a FR as the main function of AVC is to maintain the formation shape by keeping the FR as close to its FP as possible.

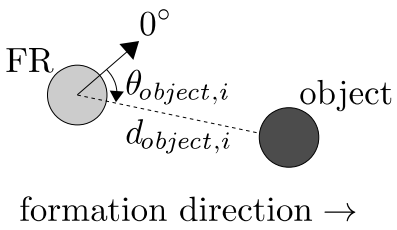


Figure 3: Measurements taken for obstacle avoidance.

3 Obstacle Avoidance

Our implementation of obstacle avoidance technique is detailed in this section. We first describe the basic potential field based technique, followed by how it can be used to support formation scaling and morphing, wall following and escaping from local minima. To better explain our obstacle avoidance technique, we define three new zones around the robot with the radii being

R_{avoid}	:	robot starts avoiding detected obstacles,
R_{wf}	:	robot takes more evasive measures such as wall-following, and
R_{stop}	:	robot stops entirely as the output of VC is 0.

Obstacles detected in these zones will trigger the robot to behave differently as described above. The radii of the three zones are dependent on the size of the robot and the formation velocity to ensure that the robots have sufficient distance to perform obstacle avoidance.

3.1 Potential Field Based Obstacle Avoidance

The potential field based obstacle avoidance technique in [9–12] considers two parameters:

1. how close the obstacle is to the robot (magnitude of repulsion force), and
2. where the obstacle is located in terms of angle relative to the robot (direction of repulsion force).

These parameters result in a net repulsion force with a magnitude and a direction. However, it is difficult to expect a nonholonomic robot to react to the repulsion force by moving in the repulsed direction instantaneously. Hence, in the following subsections, we provide the details of some modifications to the VC and AVC that allow a nonholonomic robot to react as if there is a repulsion force acting on it. Note that these modifications are only applied when the robot is required to perform obstacle avoidance.

Obstacle avoidance is performed in two stages by considering the distance, d_{object} and the angle of an object, θ_{object} measured from the robot. The stages are

1. reducing velocity, and
2. turning away.

An object is deemed blocking the robot if it is within R_{avoid} from the robot, and is in the forward 180° arc of the robot. Forward is defined to be either at the front of the robot if the

VC output is positive, or back of the robot if the VC output is negative. In the first stage, the robot only considers the closest object and reduces its velocity with a multiplier

$$v_{multiplier1} = 1 - e^{-\frac{K_{rise}}{R_{avoid}}(d_{object} - R_{wf})} \tag{9}$$

where

K_{rise}	:	exponential curve rising time constant,
R_{avoid}	:	radius of obstacle avoidance zone,
d_{object}	:	distance between robot and closest object, and
R_{wf}	:	radius of wall-following zone.

However, the reduction in velocity can be excessive if the object is located nearer to the side of the robot rather than blocking it directly in the forward direction. Hence, a second multiplier is added to dampen the effect of the first multiplier:

$$v_{multiplier2} = \frac{|\theta_{object}|}{90^\circ}\left(\frac{1 - v_{multiplier1}}{v_{multiplier1}}\right) + 1 \tag{10}$$

where θ_{object} is the angle of the closest object relative to the robot, with its value bounded by

$$-180^\circ < \theta_{object} \leq 180^\circ \tag{11}$$

Unlike in formation control, θ_{object} is measured with respect to the centre of the robot's forward arc (see Fig. 3) as obstacle avoidance is performed on an individual basis rather than collectively as a formation. If an obstacle is located at the side of the robot, with θ_{object} being 90°, $v_{multiplier2}$ would have a value of $\frac{1}{v_{multiplier1}}$, hence completely negating the deceleration caused by $v_{multiplier1}$. The final velocity is then calculated using

$$v_{final} = v \cdot \max\left(0, v_{multiplier1} \cdot v_{multiplier2}\right) \tag{12}$$

where v is the output of the VC.

The next step is to steer the robot away from the object. When avoiding an object, the robot disregards the output of the AVC. Instead, for every object within R_{avoid}, the robot uses the θ_{object} to calculate a $\omega_{avoid,i}$ with the equation

$$\omega_{avoid,i} = \begin{cases} 90^\circ - \theta_{object,i} & \text{if } 0^\circ < \theta_{object,i} \leq 180^\circ \\ -90^\circ - \theta_{object,i} & \text{otherwise.} \end{cases} \tag{13}$$

This is further improved upon by considering the d_{avoid} in calculating the $\omega_{avoid,i}$, which gradually turns the robot away more as the object approaches (14). The robot then finally calculates and turns away with an angular velocity of ω_{final} using

$$\omega_{multiplier,i} = d_{object,i}\left(\frac{\omega_{min} - \omega_{max}}{R_{avoid}}\right) + \omega_{max} \tag{14}$$

$$\omega_{final} = \sum_{i=1}^{n} \omega_{multiplier,i} \cdot \omega_{avoid,i} \tag{15}$$

where

$\theta_{object,i}$	:	angle from robot to object i,
$d_{object,i}$	:	distance between robot and objecti,
ω_{min}	:	minimum angular velocity output,
ω_{max}	:	maximum angular velocity output,
R_{avoid}	:	radius of obstacle avoidance zone, and
ω_{final}	:	final angular velocity output.

3.2 Formation Scaling

Formation scaling reduces the size of the formation when confronted with external objects. This reduces the repulsive forces experienced by the formation and allows the robots to maintain their original formation shape for a longer period. Scaling is triggered whenever any robot detects obstacles within R_{scan}. This information is conveyed to all other robots within the formation and the distance constraints on the robots are continuously reduced by a factor of K_{scale} till a minimum value of K_{scale_min} is reached, as expressed by

$$l_{constraint}(t+1) = \begin{cases} l_{constraint}(t) \cdot K_{scale} & \text{if } \frac{l_{constraint}(t)}{K_{scale\_min}} > l_{constraint}(0) \\ l_{constraint}(t) \cdot K_{scale\_min} & \text{otherwise} \end{cases} \tag{16}$$

where

$l_{constraint}(t)$	:	required distance between robot and its LL at time t,
K_{scale}	:	rate at which formation size scales, and
K_{scale_min}	:	minimum formation scaling factor.

K_{scale_min} is selected such that the robots do not get within R_{avoid} of other robots. When no obstacles are detected, the process is reversed using

$$l_{constraint}(t+1) = \begin{cases} \frac{l_{constraint}(t)}{K_{scale}} & \text{if } \frac{l_{constraint}(t)}{K_{scale}} < l_{constraint}(0) \\ l_{constraint}(0) & \text{otherwise} \end{cases} \tag{17}$$

K_{scale} governs the rate at which the formation size scales, hence its value is empirically determined based on the environment layout.

3.3 Formation Morphing

When subjected to environments with obstacles, repulsive forces from objects cause the robots to deviate from their ideal FPs even with formation scaling. This may lead to some FRs being disconnected from their LLs. Formation morphing is introduced as a fail-safe should formation scaling fails to maintain the formation structure. The use of character costs and character set matrix for every formation shape forms the basis of the implemented formation morphing solution as in [17].

Whenever a robot detects an object within R_{avoid}, it broadcasts a signal to trigger the consideration for formation morphing. The robots go through the same sequence as when assigning FPs (see Section 2.1) multiple times for all predefined formation shapes. For every formation shape, the distance error to ideal FPs of each robot is shared so that the total distance error for every formation shape is calculated and recorded. The robots then select the shape that has the lowest distance error and the new FP constraints are assigned to them. To ensure that formation morphing occurs successfully without the risk of losing any robots, a transition matrix is applied to the adjacency matrix as described in [3].

However, experiments show that the formation may alternate rapidly between two formation shapes which have similar total distance errors. Two mechanisms are introduced to prevent this issue. The first mechanism is a morph timer which prevents the formation from morphing to another shape if the elapsed time of the morph timer is less than T_{morph}. This timer resets whenever the formation successfully morphs to a different shape. The second mechanism makes use of hysteresis when deciding on a new formation shape to morph to.

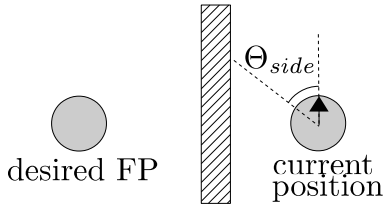


Figure 4: Situation where a robot may not be able to move forward.

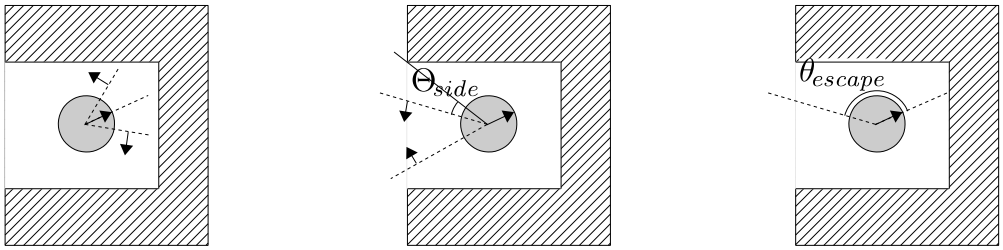


Figure 5: Finding θ_{escape}: (a) search begins from the centre of the front arc and gradually expands in both left and right directions by one resolution each iteration, (b) the aim is to find an angle where all obstacles within R_{avoid} are beyond Θ_{side}, and (c) this angle is set to be θ_{escape}.

The formation will only morph to a new shape if the total distance error of the current shape is greater than the total distance error of the new shape by a factor of K_{morph}. Both T_{morph} and K_{morph} are determined empirically and are proportional to the size of the formation.

3.4 Wall Following

A robot may occasionally find itself in a situation where it is facing roughly 90° away from its FP, and that there are obstacles in between the robot and its FP, as depicted in Fig. 4. While the AVC tries to turn the robot towards its FP, the obstacle would in return push the robot in the opposite direction. Coupled with the small output from the VC, the robot is unable to move to its FP. To overcome this, a wall following technique based on [19] has been implemented. When an obstacle is detected within R_{wf} and sits beyond a threshold angle, Θ_{side} (measured from the centre of the forward arc), the robot is forced to travel in the forward direction at V_{lock}, which a sufficiently small value which allows it to navigate around tight corners. However, should any obstacle angles falls within Θ_{side} or the obstacle distance is within the safety margin of R_{stop} from the robot, wall following will be aborted to prevent potential collisions. Θ_{side} should be set such that the robot has a sufficiently wide opening to move forward without colliding into the obstacles when they are initially detected at R_{wf}.

3.5 Escaping Local Minima

There is always a possibility that the robot will be trapped in local minima, where it is unable to move towards its FP. To address this issue, a robot first has to recognize that it is trapped by monitoring its velocity constantly. Should the velocity fall below a threshold value, V_{trap} (close to 0) for a time, T_{trap}, the robot assumes that it is trapped and attempts to free itself. The hold time is introduced as the robot's velocity can momentarily fall below V_{trap} when it

Table 1: Parameters for implementation specific variables

Variable	Value	Variable	Value	Variable	Value
$l_{constraint}(0)$	360 mm	R_{stop}	5 mm	T_{morph}	0.5 s
V_{leader}	30	K_{rise}	10	K_{morph}	1.5
K_{ω}	1.5	ω_{min}	1	Θ_{side}	60°
R_{dzone}	100 mm	ω_{max}	2.5	V_{lock}	70 mm/s
R_{avoid}	100 mm	K_{scale}	0.999	T_{trap}	1.5 s
R_{wf}	20 mm	K_{scale_min}	0.5	T_{escape}	10 s

is performing a zero radius turn, rather than being trapped. Hence, T_{trap} is set to be slightly greater than the time the robot takes to perform a 180° turn to prevent the misinterpretation.

It first finds an angle that is clear of obstacles. Starting with the centre of the robots front arc defined as 0°, the sequence checks for the angles of all surrounding obstacles that are within R_{avoid}, and determine if they are all beyond an angle, $\Theta side$, which is measured from the current angle of interest (0°). If the condition is not satisfied, the robot will then consider the next two angles which are 1° further away on both left and right sides, from the centre of the front arc $(\pm 1°, \pm 2°, \ldots, \pm 179°, 180°)$. This process is repeated until an angle, θ_{escape}, meets the condition where all obstacles are beyond the $\Theta side$ threshold. The robot then turns towards θ_{escape} and is forced to move at a positive velocity which is the magnitude of its last calculated v_{final}. It does this for a period, T_{escape}, after which the robot would assume that it is no longer trapped and resumes running the VC. The process is illustrated in Fig. 5. The value of T_{escape} is dependent on the layout of the obstacle course. It should be long enough to prevent the robot from travelling back to the same local minimum but short enough so that the robot can still rerun the formation control system in time to catch up with the rest of the formation.

4 Formation Rebuilding

The formation is capable of rebuilding itself should the number of operating robots changes. The term rebuild is defined as the ability for the formation to reform the last formed formation shape by considering the number of operating robots at any time instant. Rebuilding can be done without affecting the whole formation to increase scalability. This is achieved by running the sequence for assigning FPs (see Section 2.1) starting from a particular FP which is determined by one of the two following conditions.

Firstly, when a new robot is added to the formation, the new robot detects the closest neighbouring robot and the FP that it is assigned to. The new robot then signals the formation to rebuild itself from that FP onwards. However, if the FP belongs to the FL, this new robot instead considers the next closest FP to prevent a change in leadership.

The second condition is met when a robot fails. If it is a LL, its FRs will be able to detect the failure and signal the formation to rebuild itself. The FP which the robot was previously assigned to is the point where the rebuilding starts.

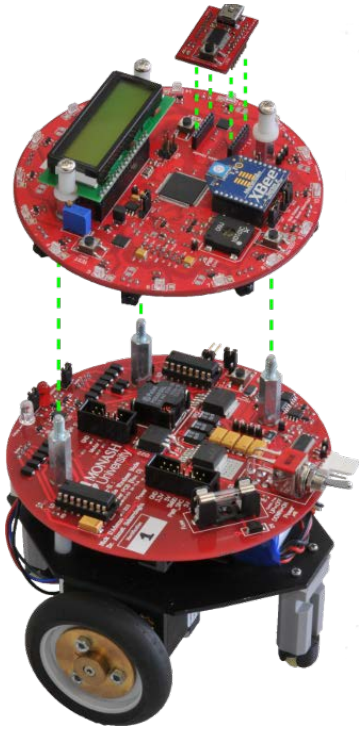


Figure 6: Experimental robot eBug. It has a diameter of 120 mm and is equipped with two stepper motors driving two wheels independently. Further details about its hardware and firmware can be found in [13].

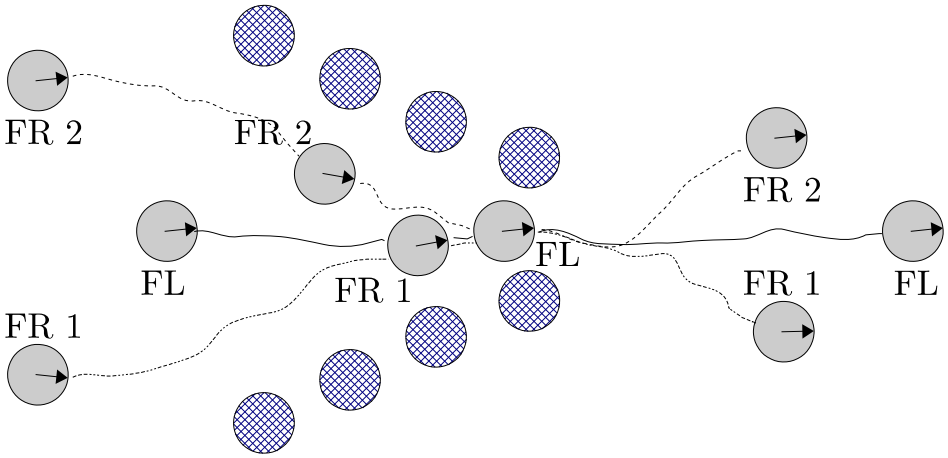
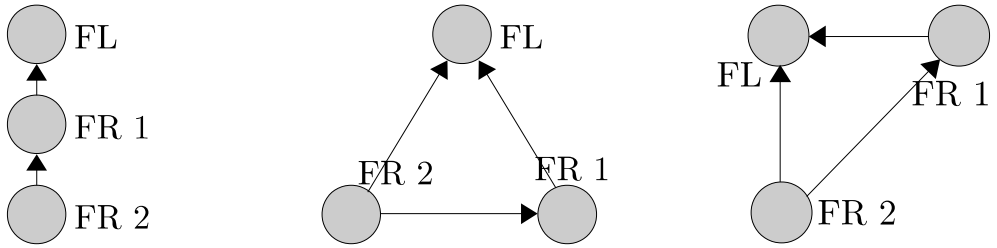
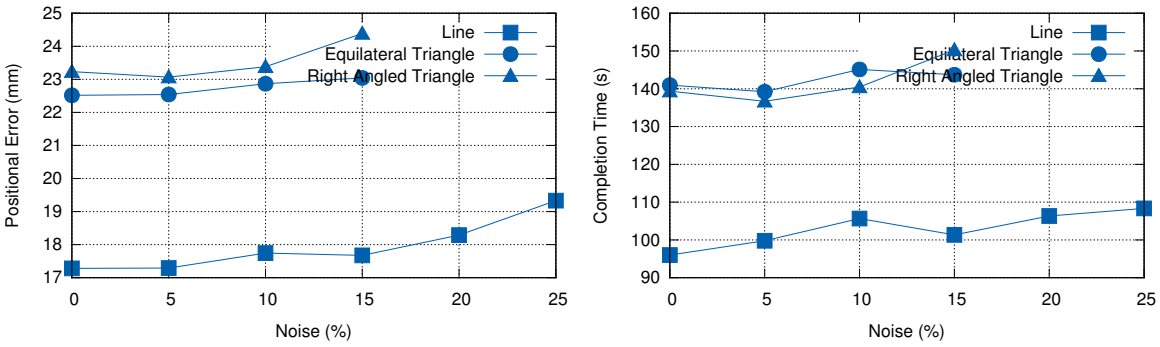


Figure 7: Equilateral triangle formation going through a funnel-shaped obstacle course.



(a) Line (b) Equilateral triangle (c) Right-angled triangle

Figure 8: Formation shapes



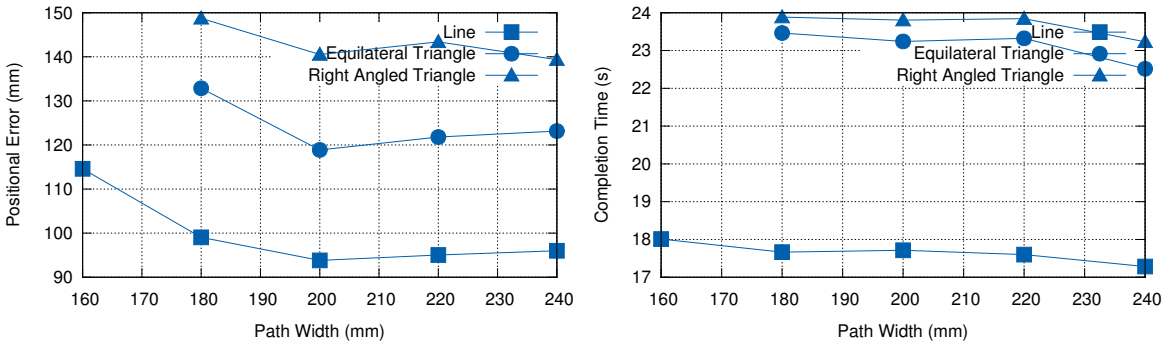
(a) Average distance between actual and ideal FPs (b) Time taken to complete the obstacle course

Figure 9: Noise performance of the algorithm.

5 Experiment Design and Results

The algorithm was implemented on a group of three eBugs, which are ground moving non-holonomic robots (Fig. 6) [13]. A base station was used for algorithm computation as the eBugs themselves do not have sufficient processing power yet. Each eBug is treated as a separate process by the base station for a decentralized approach and movement commands are sent to the eBugs wirelessly via ZigBee packets. Laser range finders with a 360° field of view on the robots were emulated with the use of the BCH marker system [20] and an overhead camera to track the coordinates of the robots. The FL was controlled through a funnel-shaped obstacle course as seen in Fig. 7, with the FRs following autonomously using the algorithm.

Fig. 8 shows the three different starting formation shapes that were considered. The performance of the algorithm was measured in terms of i) average deviation from ideal FP and ii) the time taken for formation to go through the obstacle course and rebuild itself to the starting shape. We looked into the robustness of the algorithm in three criteria: i) sensor noise, ii) path width and iii) formation velocity. The corresponding variables in the aforementioned criteria are systematically varied in the experiments until the formation breaks, which is when a FR fails to detect any of its LL within R_{scan}. The values used for implementation specific variables in the experiments are presented in Table 1.



(a) Average distance between actual and ideal FPs (b) Time taken to complete the obstacle course

Figure 10: Performance of the algorithm in tightly confined spaces.

5.1 Robustness to Sensor Noise

Accuracy of the range sensors may deteriorate if the obstacle is too close to the sensor or the surface property is non-ideal. This adversely affects the performance of our distributed algorithm where every robot is highly reliant on local information such as the measured distances of nearby objects. In this experiment, noise was either added to or subtracted from (determined randomly with a 50% probability) the measurements of the range sensors as described by the equation

$$d_{object} = d_{actual} \pm N \tag{18}$$

where

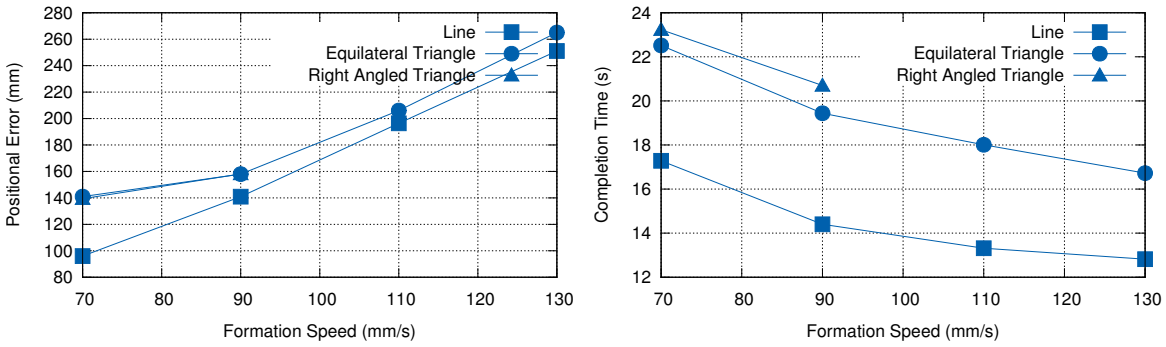
d_{object} : measured distance between robot and object,
d_{actual} : actual distance between robot and object, and
N : maximum amount of noise in percentage.

The amount of noise was also randomly determined, with a value between 0 and $\eta \cdot d_{actual}$. η is increased at 5 % intervals until the formation breaks. The experiment is performed 30 times for every value of η.

Fig. 9 presents the results of the experiment, showing an increasing trend in the deviation from ideal FPs and completion time. The robots tend to move about indecisively when the measured object distances were fluctuating randomly. At 20 % noise level, the FRs in the equilateral and right-angled triangle formation often lost tracked of their LLs even before entering the funnel, breaking the formation. The line formation finally broke at a noise level of 30 %. These results are well beyond our expectations and the limitations of most range sensors. An interesting observation was made with the presence of noise. Near the exit, the obstacles that were placed symmetrically exerted repulsive forces on the robots with a net vector in the opposite direction of formation movement. This stopped the robots from moving forward or turning. Addition of noise occasionally helped to create a net repulsive force pointing slightly away from the opposite of formation movement, allowing the robots to go through the narrow point.

5.2 Robustness to Path Width

The second experiment tested the ability of the formation go through tightly confined areas without breaking the formation and losing any robots. Starting at 240 mm, which is twice the



(a) Average distance between actual and ideal FPs (b) Time taken to complete the obstacle course

Figure 11: Performance of the algorithm as formation speed increases.

diameter of an eBug, the path width is reduced at 20 mm intervals for each set of experiment. The algorithm is tested 30 times for each path width and the path width is continuously reduced until the formation consistently breaks.

As the path width narrowed, the FRs were repelled by the obstacles and eventually formed a line formation to go through the exit. This explains the superiority of the line formation over the other two shapes as illustrated in Fig. 9, as the FRs had to travel extra distances to get to the new FPs after morphing. Theoretically, the smallest gap that the formation is able to go through is equal to the diameter of the robot plus the safety margin of $2R_{stop}$, or 130 mm. Fig. 10 shows that the smallest width that the formation managed to go through is 160 mm. At that point, the FRs could no longer morph quickly enough to keep up with the FL, causing the equilateral and right-angled triangle formations to break. The line formation finally broke when the path width was set at 140 mm, which is close to the diameter of the robots. At this width, the robots spent considerable amount of time to wiggle their way through the exit due to a low VC output, which causes the FRs to lag behind the FL.

5.3 Robustness to Formation Speed

Lastly, we tested the algorithm's ability to keep up with the FL at increasing movement speeds while maintaining the formation shape. It is important to investigate this issue as the algorithm is intended to be deployed onto different types of robots and missions, each with its own operation speed. In our experiment, we started by having the FL travelling at 70 mm/s and slowly increased its velocity at intervals of 20 mm/s, with each interval tested 30 times. This formation speed is allowed to increase until the formation broke.

The most immediate observation in Fig. 11 is how quickly the average deviation from ideal FP scaled up with speed. This is understandable as V_{leader} was optimized for the default speed of 70 mm/s. Also noted is the decrease in completion time as the FL increases its speed. The right-angled triangle formation was the first to break at a speed of 110 mm/s. Triangle formation fared better due to the shape of the funnel which resembles a triangle shape, hence the FRs were able to scale down in size first before morphing, significantly reducing the deviation from FPs at higher speeds. Line and triangle formations eventually broke at the speed of 150 mm/s as the VC was unable to cope with the large d_{error}.

6 Concluding Remarks

A novel adaptive formation control algorithm for wireless mobile robot networks has been created. The algorithm allows a multi robot system to preserve its formation in the presence of obstacles. The process of forming and maintaining different formation shapes throughout navigation has been thoroughly investigated. A robust obstacle avoidance approach has also been incorporated into the algorithm, allowing the formation to go through obstacle courses without any collisions. The robots are also able to maintain formation structures consistently via formation scaling and morphing in order to avoid obstacles. Results show that the FRs deviate from their ideal FP by as low as 96 mm, which is less than the diameter of our robots. Our future research will address how i) the distance and angle constraints can be determined autonomously based on the environment and the instantaneous number of robots, and ii) how formation scaling and morphing can be made more scalable.

7 Acknowledgments

This research was supported in part by the *Lise and Arnfinn Hejes Grant for Education and Research*.

References

[1] J. C. Barca and Y. A. Şekercioğlu. "Swarm Robotics Reviewed". In: *Robotica* (2012).

[2] J. Shao, G. Xie, and L. Wang. "Leader Following Formation Control of Multiple Mobile Vehicles". In: *IET Control Theory and Applications* 1.2 (2007), pp. 545–552.

[3] J. P. Desai, J. P. Ostrowski, and V. Kumar. "Modeling and Control of Formations of Nonholonomic Mobile Robots". In: *IEEE Transactions on Robotics and Automation* 17.6 (Dec. 2001), pp. 905–908.

[4] M. N. Soorki, H. A. Talebi, and S. K. Y. Nikravesh. "A Robust Dynamic Leader-Follower Formation Control with Active Obstacle Avoidance". In: *IEEE International Conference on Systems, Man, and Cybernetics*. Anchorage, Alaska, USA, 2011, pp. 1932–1937.

[5] M. N. Soorki, H. A. Talebi, and S. K. Y. Nikravesh. "A Robust Leader-Obstacle Formation Control". In: *IEEE International Conference on Control Applications*. 2011, pp. 489–494.

[6] S. Hou, C. Cheah, and J. Slotine. "Dynamic Region Following Formation Control for a Swarm of robots". In: *IEEE International Conference on Robotics and Automation*. 2009, pp. 1929–1934.

[7] C. R. M. Kuppan, M. Singaperumal, and T. Nagarajan. "Distributed Planning and Control of Multirobot Formations with Navigation and Obstacle Avoidance". In: *IEEE Recent Advances in Intelligent Computational Systems*. 2011, pp. 621–626.

[8] A. S. Brandao, M. Sarcinelli-Filho, R. Carelli, and T. F. Bastos-Filho. "Decentralized Control of Leader-Follower Formations of Mobile Robots with Obstacle Avoidance". In: *IEEE International Conference on Mechatronics*. 2009, pp. 1–6.

[9] J. C. Barca and Y. A. Şekercioğlu. "Generating Formations with a Template Based Multi-Robot System". In: *Australasian Conference on Robotics and Automation*. 2011.

[10] Y. Liang and H.-H Lee. “Decentralized Formation Control and Obstacle Avoidance for Multiple Robots with Nonholonomic Constraints”. In: *American Control Conference*. 2006.

[11] J. Wang, X.-B. Wu, and Z.-L. Xu. “Decentralized Formation Control and Obstacles Avoidance Based On Potential Field Method”. In: *International Conference on Machine Learning and Cybernetics*. 2006, pp. 803–808.

[12] J. P. Desai, J. P. Ostrowski, and V. Kumar. “Formation Control and Obstacle Avoidance Algorithm of Multiple Autonomous Underwater Vehicles (AUVs) Based on Potential Function and Behavior Rules”. In: *IEEE International Conference on Automation and Logistics*. 2007, pp. 569–573.

[13] N. D’Ademo, W. H. Li, D. Lui, Y. A. Şekercioğlu, and T. Drummond. “eBug - An Open Robotics Platform for Teaching and Research”. In: *Australasian Conference on Robotics and Automation*. 2011.

[14] J. C. Barca, Y. A. Şekercioğlu, and A. Ford. “Controlling Formations of Robots with Graph Theory”. In: *12th International Conference on Intelligent Autonomous Systems*. 2012, pp. 1–8.

[15] P. Ogren and N. E. Leonard. “Obstacle Avoidance in Formation”. In: *IEEE International Conference on Robotics and Automation*. Vol. 2. 2003, pp. 2492–2497.

[16] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor. “A Vision-Based Formation Control Framework”. In: *IEEE Transcations on Robotics and Automation* 18.5 (2002), pp. 813–825.

[17] B. Anderson, C. Yu, B. Fidan, and J. Hendrickx. “Rigid Graph Control Architectures for Autonomous Formations”. In: *IEEE Control Systems* 28.6 (2008), pp. 48–63.

[18] Y.-C. Chen and Y.-T. Wang. “Obstacle Avoidance and Role Assignment Algorithms for Robot Formation control”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 4201–4206.

[19] F. Yang, F. Liu, S. Liu, and C. Zhong. “Hybrid Formation Control of Multiple Mobile Robots with Obstacle Avoidance”. In: *8th World Congress on Intelligent Control and Automation*. 2010, pp. 1039–1044.

[20] *ARToolKitPlus*. URL: http://studierstube.icg.tugraz.at/handheld_ar/artoolkitplus.php.