Formation Control: A Centralized and Decentralized Approach

Ajay Ahir, Ben Philps and Sumaiyah Kola

***Abstract*— Multi-robot formations have proved useful in applications for exploration, surveillance and 'search and rescue'. In such domains reliable global communication is not a guarantee. In this paper we implement a centralized and decentralized approach to multi-robot formation control that includes a reactive formation switching strategy. The robot formations successfully navigate through obstacle fields, tight corners and narrow corridors.**

I. INTRODUCTION

In application domains such as exploration, surveillance and 'search and rescue', coordination and control mechanisms for multiple robots have been shown to provide cost effective and fault tolerant solutions [1].

Inspired by the natural coordinated behavior of bird flocking and ant swarming, formation control of multiple robots can improve surveillance coverage by combining sensor readings from individual agents. The main objective is for multiple robots to traverse the environment while maintaining an explicitly specified spacing relationship between agents.

In this paper we provide a centralized behavior-based approach to formation control with a formation switching strategy. This solution relies on each agent transmitting and receiving information from a global controller. We also provide a fault-tolerant decentralized approach, adopting a message passing technique, suitable when communication is restricted.

A. BACKGROUND

Balch and Arkin provide a behavior-based approach to formation control, dividing the task into behavioral components, referred to as 'motor schemas' [2].

Motor Schemas: Given the current sensor inputs and robot positions, each motor schema generates a vector, $\vec{V}_s$, as a behavioral response. $\vec{V}_s$ indicates the direction and magnitude of desired movement. Motor schemas for goal navigation, static/dynamic obstacle avoidance and formation maintenance generate vectors $\vec{V}_{goal}$, $\vec{V}_{static\_obs}$, $\vec{V}_{dynamic\_obs}$, $\vec{V}_{form}$, respectively. A gain value g_s dictates the contribution of each schema to the overall behavior.

The overall behavioral response, $\vec{V}$, is calculated by weighting each vector with its gain value then summing and normalizing the result. To overcome local minima, an additional schema generating noise, $\vec{V}_{noise}$, is included.

Formation Maintenance: Each robot R_i, in position P_{R_i}, has a desired formation position F_i from which $\vec{V}_{form}$ is defined:

$$\vec{V}_{form} = F_i - P_{R_i}$$

.

Different referencing schemes exist to find F_i. Unit-center computes the average P_R as the centre of the formation, Leader-referencing defines the formation relative to a leader robot's pose, and Neighbour-referencing defines each robot's position relative to a predefined neighbour.

$|\vec{V}_{form}|$ is determined by the distance between P_{R_i} and F_i. Zones are defined around F_i as seen in Fig. 1.

Within the *dead zone*, the robot is considered to be in formation and so $|\vec{V}_{form}| = 0$. Within the *control zone*, the magnitude linearly increases from the inner edge. The magnitude at the outer edge of the *control zone* is propagated throughout the *ballistic zone*.

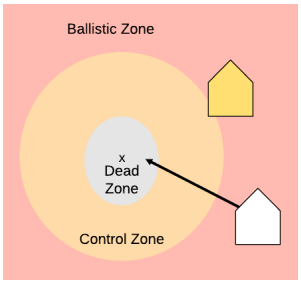

Fig. 1. Zones for computing $\vec{V}_{form}$ with F_i marked as 'x'

II. APPROACH

We implement a Leader-referenced behavior-based approach to formation control. Additionally, we extend the work of Balch and Arkin to incorporate a formation switching strategy and a decentralized approach [6].

A Leader-referenced approach allows the formation position of each robot to be determined with only knowledge of the position of the leader, R_0. In contrast, a Unit-center-referenced approach would require complete knowledge of all robot positions. Purely decentralizing this would require transmitting a substantial amount of state between robots.

We define motor schemas for goal nagivation, obstacle avoidance and formation maintenance returning vectors $\vec{V}_{goal}$, $\vec{V}_{form}$, $\vec{V}_{obs}$, respectively. Vectors from each schema are weighted and summed into an overall behavioural response $\vec{V}$ for each robot.

In a centralized system, where all robot poses are known, the addition of logical rules handles the avoidance of other robots.

A. EXPERIMENTAL SET-UP

We conduct experiments on 5 TurtleBot3 [3] robots controlled using ROS [4] with Gazebo [5] as the simulation

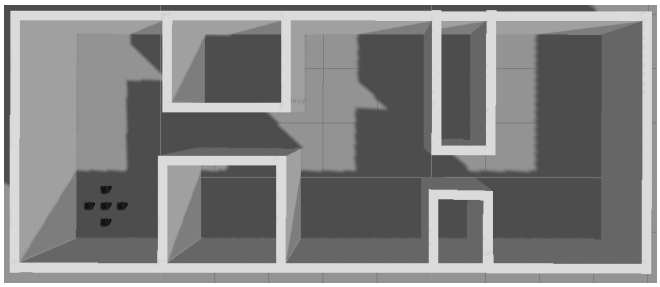

Fig. 2. The *diamond* formation in an arena with tight corridors

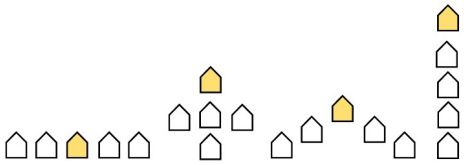

Fig. 3. Formations L-R: *line*, *diamond*, *wedge*, *column* with leader colored

environment. We assume all robots are identical and labeled, with R_0 designated as leader and $R_1, ..., R_n$ as followers.

Each simulated robot is a non-holonomic differential-drive robot with control inputs u and ω, corresponding to linear and angular velocities, respectively. There are 5 sensors per robot (`right`, `front_right`, `front`, `front_left`, `left`) evenly distributed around the front half of the robot.

We use feedback linearization to convert $\vec{V}$ from coordinates in the global frame into control inputs u and ω. This allows us to assume control of each robot via a holonomic point held at a distance of ε. Here ε is 0.2m.

We empirically tune system parameters to provide a general solution capable of handling obstacles, narrow passages and tight corners across a range of arenas. In our evaluation we consider two arenas that highlight split and merge as well as the formation switching strategy (Fig. 2 and 4).

B. FORMATIONS

We consider the following formations for 5 robots: *diamond*, *column*, *line* and *wedge* (Fig. 3). Each formation is defined relative to the leader. The number of robots per formation and the intra-formation distance between robots can easily be adjusted to suit the environment.

C. MOTOR SCHEMAS

The objective of the leader, R_0, is to reach the goal, avoiding obstacles and other robots. The objective of the followers, $R_1, ..., R_n$, is to maintain formation while avoiding obstacles.

Each motor schema has an associated gain value (Table I). The values were tuned empirically. Prioritizing obstacle avoidance above formation maintenance permits the desired split and merge behaviour (see *Obstacle Avoidance*).

Consequently, the behavioral response $\vec{V}$ for robot R_i is:

$$\vec{V} = \begin{cases} g_{goal}\vec{V}_{goal} + g_{obs}\vec{V}_{obs} + \vec{V}_{noise} & \text{if } R_0 \\ g_{form}\vec{V}_{form} + g_{obs}\vec{V}_{obs} + \vec{V}_{noise} & \text{if } R_1, ..., R_n \end{cases}$$

Motor Schema		Value
Goal Navigation	g_goal	0.35
Obstacle Avoidance	g_obs	0.22
Formation Maintenance	g_form	0.20

TABLE I
MOTOR SCHEMA GAIN VALUES g_s

where $|\vec{V}_{noise}| = 0.05$. A useful heuristic when robots are very close to obstacles is to temporarily replace $\vec{V}_{form}$ with $\vec{V}_{goal}$. Velocities switch back to $\vec{V}_{form}$ when followers are clear of the obstacle, to correctly navigate the followers.

Goal Navigation: $\vec{V}_{goal}$ points from R_0 to the next point along a path from P_{R_0} to the goal. $|\vec{V}_{goal}| = 1$.

We generate a path using a variant of the rapidly-exploring random trees (RRT) algorithm, RRT*. RRT* explores the search space by building a space-filling tree. Additional rewiring and cost minimization steps can generate an approximate shortest path to the goal. Finding the optimal path requires many iterations and RRT* becomes inefficient. To avoid this overhead, we precompute an optimal path and store it.

Obstacle Avoidance: For the centralized approach, a Braitenberg controller was adopted. $\vec{V}_{obs}$ is a weighted sum of smoothed sensor measurements. Sensor measurements are smoothed using the tanh function to eliminate infinities. The sensors `front_left` and `front_right` are weighted to contribute the highest to $\vec{V}_{obs}$.

Since robots can detect others with their sensors, robots are filtered out from each sensor's view. Where robot R_B is in front of a sensor on robot R_A, and there is no other obstacle in the direction of R_A's sensor, R_A sensor's value is set to its maximum measurement. This allows robots to avoid obstacles while remaining in formation close to other robots.

When detecting an obstacle close on the front sensor, robots determine a preferred direction proportional to each smoothed sensor measurement and its position in the formation (e.g a robot on the left hand side of the formation will turn to the left to avoid a head on collision). This allows the robots to split-and-merge efficiently, which also reduces the chance of robots crossing paths. It further encourages wider exploration of the environment.

Filtering robots from sensors is not possible in the decentralized approach without knowledge of every robot's position. A less aggressive rule-based controller, that allows robots to get closer to obstacles before avoiding, was implemented. This allows robots to stay in formation even when their sensors detect other robots. Both Braitenberg and rule-based controllers return a vector V_{obs} scaled in the range $[0, 1]$.

Formation Maintenance: We implement the motor schema defined in section I-A. Table II contains the parameters used.

To compute F_i we transform the vector of relative formation positions, by rotating it to match the leader's orientation and translating it to the leader's position.

	Parameter	Value (m)
(1)	Robot Radius	0.05
(2)	Spacing Distance	0.8
(3)	Dead Zone Radius	$1.5 \times (1) = 0.075$
(4)	Control Zone Radius	$(2) + (3) = 0.875$

TABLE II
FORMATION MAINTENANCE PARAMETERS

We scale $|V_{form}|$ to be in $[0,1]$.

When follower robots are not in their dead zone, g_{goal} is decreased, so that the leader waits for the followers to regain their desired formation position.

D. DYNAMIC OBSTACLE AVOIDANCE

To prevent robots from colliding with each other, we set $|\vec{V}| = 0$ for R_i if it is behind some R_j. If robots are approximately adjacent, the robot with lowest ID halts while the other can move, preventing deadlock.

E. FORMATION SWITCHING STRATEGY

We implement a reactive formation switching strategy. This determines the safest formation for the robots, given the current environment

If the leader, or at least half of the followers, detect a corridor, the formation is switched to a *column*. As the narrowest formation, *column* is chosen. This formation is maintained until the robots exit the corridor, at which point it is safe to return to the default formation.

A robot is said to 'detect' a corridor if the environment ahead of the `front` sensor is clear and the `left` and `right` sensors report measurements below a threshold, here 0.4m.

F. DECENTRALIZED

The centralized algorithm relies on knowledge of all robot positions as well as their combined view of the world. Our decentralized solution achieves this with limited message passing between robots to communicate the required state.

Communication links are defined between certain robots in each formation. For *line*, *column* and *wedge*, links are defined between adjacent robots. For *diamond*, all robots are arranged so they can communicate with the center robot.

Our consensus algorithm uses each robot's ID to decide the leader. The robot who's ID has been assigned to position $(0,0)$ in the formation is the leader.

Robot ID's are unique. Only the leader initiates message passing when their state updates, and messages are propagated across the formation links.

III. RESULTS

Both systems successfully split and merge around obstacles, navigate tight corners and switch formation to handle corridors. In a video we further demonstrate this by adding an obstacle added at run-time [6].

We conducted experiments in multiple arenas. We present results below for the square (Fig. 4, 5) and corridor (Fig. 2, 6) arenas. The square arena features multiple obstacles, while the corridor challenges the robots through narrow passages. Fig. 5 shows *diamond* splitting around narrow obstacles in the square arena, and Fig. 6 shows *wedge* switching to *column* in the corridor arena.

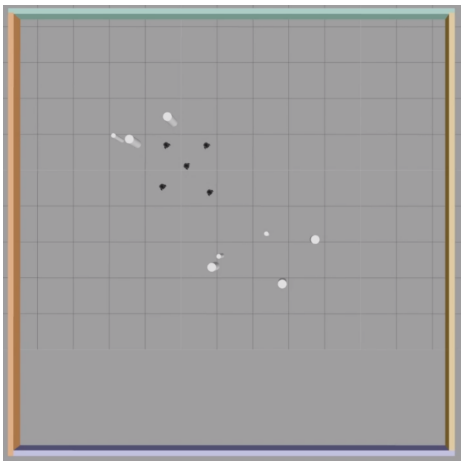

Fig. 4. *Diamond* navigating an obstacle field in the square world

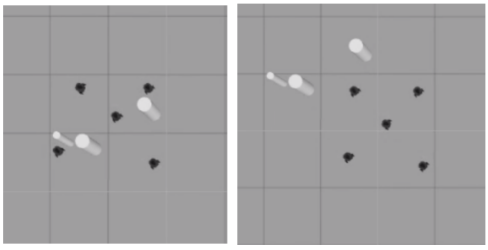

Fig. 5. *Diamond* splitting (LHS), then merging (RHS) around obstacles

In the square arena, the formation traverses a straight line path from top left to bottom right. In the corridor arena, the formation begins as shown in Fig. 2; robots traverse a path through both corridors.

We define the position error for R_i as $|F_i - P_{R_i}|$. Fig. 7 and 8 illustrate the position errors of follower robots traversing the square arena in a *line* formation using the centralized and decentralized systems, respectively. For each formation in each arena, we average the position error across time for all follower robots (Table III and IV). Additionally, the tables contain the time taken for all robots to reach the goal.

In all cases, *column* reaches the goal fastest with smallest

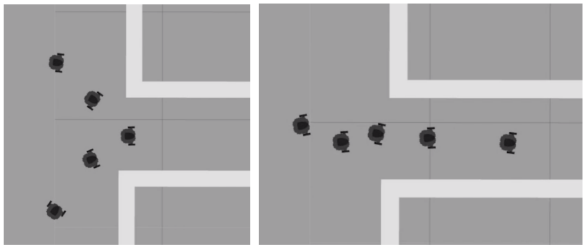

Fig. 6. *Wedge* (LHS) transitioning to *column* (RHS) upon corridor entrance

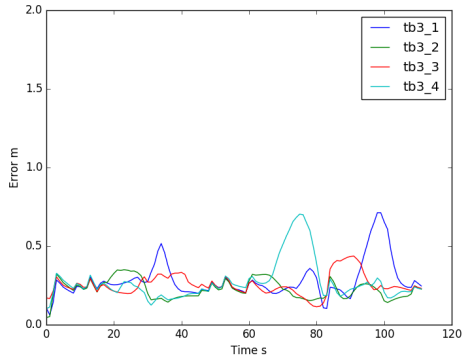

Fig. 7. Centralized system, *line* formation, square arena, running at half speed, to compare to decentralized

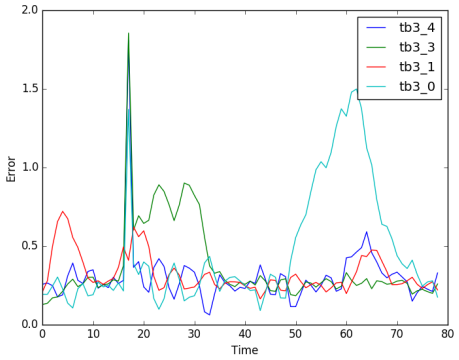

Fig. 8. Decentralized system, *line* formation, square arena

	Centralized		Decentralized	
Formation	Time to Goal (s)	Position Error (m)	Time to Goal (s)	Position Error (m)
line	68	0.64	116	1.12
column	63	0.28	76	0.62
wedge	64	0.57	109	1.18
diamond	76	0.40	101	1.05

TABLE III

PERFORMANCE IN THE CORRIDOR ARENA

	Centralized		Decentralized	
Formation	Time to Goal (s)	Position Error (m)	Time to Goal (s)	Position Error (m)
line	65	0.26	87	0.74
column	43	0.22	68	0.42
wedge	62	0.26	82	0.85
diamond	63	0.27	75	0.62

TABLE IV

PERFORMANCE IN THE SQUARE ARENA

position error. In contrast to other formations, *column* does not need to slow down to switch formation upon encountering a corridor. However *column* offers no arena exploration benefit over a single robot as $R_1,...,R_n$ follow R_0's path. As the widest formation, *line* offers a better exploration of the arena. However, for robots further from the leader (who has a safe RRT* path), they are more likely to encounter difficult obstacle situations, resulting in higher time to goal and worse formation accuracy. In general *diamond* and *wedge* offer a compromise, with smaller position errors while exploring more of the arena. In table III, for the centralized system, we see the *line* formation with a position error of 0.64m while the *wedge* and *diamond* produce smaller values of 0.57m and 0.40m, respectively.

In Fig. 7 and 8, larger curve peaks indicate the formation splitting around an obstacle and attempting to merge. We see that the centralized system achieves a lower average position error, likely due to robots regaining formation faster and deviating less when splitting around obstacles. In Fig. 7, all robots merge into formation after a split within 15s. In Fig. 8, some robots take up to 25s to achieve this. For each formation the centralized system achieves lower position error and in most cases a faster time to reach the goal III and IV. In the square arena, the *diamond* in the centralized solution reaches the goal in 63s with an error of 0.27m and 75s with an error of 0.62m in the decentralized system.

IV. CONCLUSION

We have implemented centralized and decentralized formation control. Each approach visibly maintains formation whilst the robots traverse the arena. The centralized approach reaches the goal in shorter time, and with lower average error. The decentralized approach is less effective at split and merge and avoiding obstacles as the robots are unaware of others' positions. This can be mitigated by increasing the spacing distance between robots in the decentralized case, so they are less likely to enter the detection range of each others' sensors. However, both systems can reach the goal effectively across a variety of arenas.

An alternative mechanism for path generation could be implemented as RRT* is costly and does not scale for new larger arenas.

The gain values were tuned for each arena and spacing distance. A reinforcement learning based approach could be used to locate optimal weights.

V. ACKNOWLEDGMENT

- Ajay Ahir - Worked on setting up the multi-robot environment and obstacle avoidance. Added the RRT* component and worked on combining velocities. Decentralized the solution and produced the error plots.
- Ben Philps - Multi-robot obstacle avoidance for centralised, decentralised systems experimenting with different approaches. Multi-robot avoidance. End-to-end testing, ensuring general solution. Results collection, footage.
- Sumaiyah Kola - Implemented formation maintenance schema. Designed experimental set-up and methodology. Worked on multi-robot obstacle avoidance and combining velocities. Built multiple simulation arenas.

REFERENCES

[1] J. S. Jennings, G. Whelan and W. F. Evans, "Cooperative Search and Rescue with a Team of Mobile Robots", 8th International Conference on Advanced Robotics, Proceedings ICAR'97, Monterey, CA, USA, pp. 193-200, Jul. 1997.

[2] T. Balch and R. C. Arkin, "Behavior-based Formation Control for Multi-robot Teams", IEEE Transactions on Robotics and Automation, vol. 14, no. 6, pp. 926-939, Dec. 1998.

[3] TurtleBot3, http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

[4] ROS - Robot Operating System, https://www.ros.org/

[5] Gazebo, http://gazebosim.org/

[6] Code Repository, https://github.com/DoodleBobBuffPants/RobotProject