Path Planning and Control of Swarm Robots

Peratchi Hariharasudhan Kannan
Peratchi.Kannan@anu.edu.au

Shreyansh Singh
Shreyansh.Singh@anu.edu.au

Rishab Jain
Rishab.Jain@anu.edu.au

***Abstract*—The project aims and experiments to design and maintain formation in swarm robotics and tries to implement the same while following a path/ escorting a target. The project studies various strategies to implement control methodologies like probabilistic path planning, kalman filter based controllers to achieve the solution.**

***Index Terms*—Robotics, path planning, kalman filter, swarm control**

I. Introduction

The formation control is one of the most important in any swarm robotics problem. This document presents a few strategies to implement such control methodologies. The literature in this field presents numerous solutions to this problem but there is plenty room for development as new technology emerges from time to time. Fundamentally, swarm movement optimisation is based on the concept of emergent behavior. ”An emergent behavior is essentially new, and generated by the combination of two or more different thingsnone of which displayed the behavior individually. An example of emergent behavior can be found in the insect world, where colonies of simple creatures like ants or termites will form together spontaneously to build very complex structures such as underground colonies or mounds that reach meters into the air, including elaborate cooling ducts and heat-dispersing fins.” [4] This means that each agent within the swarm is programmed to perform simpler tasks which leads to a complex behaviour within the group. The course presented strategies that can be followed (Kalman filter, model predictive control)to be implemented as a solution. In the following sections, we present the system model, problem we are trying to address and the solutions that we tried implementing in *MATLAB*. We used a computer powered by *Intel Core i5* with 8GB of RAM and AMD Radeon R5 M430.

II. Previous Work

Particle Swarm Optimisation (PSO) has been around for a couple of decades and the original and first study in this field is accredited to Kennedy and Eberhart where they discussed the concept for optimization of non-linear functions with the help of particle swarm methodology [1]. The motivation for the paper was to study and model human social behaviour. Unlike the beliefs and attitudes of human, which can be identical for two individuals, the same cannot be said for the position of two birds that want to occupy the same space without collision. Thus, the concept of change in human social behaviour can be analogous to the bird movement in physical space [1]. The first approach in [1] talks about nearest neighbour velocity matching which simulated a random population of birds on torus pixel grid with *X* and *Y* velocities and with every loop, each agent/bird was assigned the nearest neighbour’s *X* and *Y* velocities in accordance with the target agent.

Although, the agents reached a unanimous and unchanging direction swiftly, the simulation seemed more artificial than lifelike. This gave rise to another approach which was called the Cornfield vector. When food is put out for birds, in a matter of minutes it can be observed that an entire flock is aware without any prior knowledge of its location. This concept was applied in the approach where the best value in the pixel grid was known to each agent once a member had found. The result of this approach showed that the intensity of randomness in the *X* and *Y* directions made the flock of birds to surround the target faster or slower [1]. This paper gave rise to a rapid growth in swarm optimisation from thereon and has been the fundamental concept for various existing algorithms.

Another work in this field can be seen in [2] which is an extended version of [1]. Sengupta et. al. have used [1] as the basis of their model and implemented it to hybridization models for better and improved particle swarm optimization. The focus here is the initialization of the agents. The term *‘inertia weight’* is defined as the control parameter that modulates the momentum of the swarm. This is necessary in order to obtain a balance between exploitation and exploration behaviour of the swarm which will result in convergence. Sengupta et. al. use different techniques to assign the inertia weight such as random selection, linear-time varying, non-linear time varying, fuzzy adaptive and so on [2]. The underlying fact is to ensure the convergent behaviour of the swarm at all times. This is governed majorly by the velocity and position update equations of the PSO algorithm given as follows,

$$v_{ij}^{t+1} = \omega v_{ij}^{t} + C_1 r_1 (xBest_i^t - x_i^t) + C_2 r_2 (gBest_i^t - x_i^t) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^t.t \quad (2)$$

where r_1 and r_2 are independent and identically distributed random numbers, C_1 and C_2 are cognition and social acceleration coefficients, x_i^t and v_i^t are position and velocity of the i^{th} agent, $xBest_i^t$ and $gBest_i^t$ are the personal and global best locations in the t^{th} iteration [2]. The second phase of the paper focuses on the hybridization techniques which integrate social, co-operative character of the algorithm with existing verified optimization strategies [2]. They discuss and outline number of algorithms with PSO such as Genetic Algorithms, Differential Evolution, Simulated Annealing, Ant Colony Optimization, Artificial Bee Colony, Bat Algorithm, Firefly Algorithm and so on [2]. In conclusion, the paper compares and analyses the various hybridization models that can be implemented for various applications.

One of the very significant works carried out in swarm intelligence is evident in [3]. The underlying concept of our project has been adapted from this paper by Dang et. al. The paper suggested algorithms for V-shape and circular shape formations as shown in figure 1.

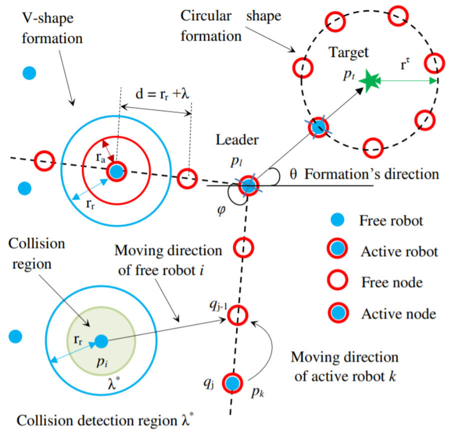


Fig. 1. Formation control method following the V-shape and circular shape desired structure [3]

III. System Model and Problem Definition

The system that we assume as our model is a differential drive robot governed by the following equations,

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

$$\dot{\xi}_R = R(\theta).\dot{\xi}_I$$

$$R(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

These equations [6] define the differential drive robot where R matrix gives the orientation of the robot, $\dot{\xi}_I$ give the robot position with respect to the world coordinates and the derived $\dot{\xi}_R$ gives the robot position in local frame. The Probabilistic path planning followed in our project uses the graph search methods to plan the path of a robot, avoiding the obstacles in the environment.

The problem that we are trying to address in this project is to understand and develop a controller for path planning and formation control. This formation control can be implemented alongside robots and other modern vehicles, which can be used for escort duties, target chasing, deliveries and many more. We present our experimentation and validation results of the algorithms in Probabilistic path planning [5] and Kalman filter in the following section.

IV. Discussion

In this section, we would like to present the trials that we ran during the second half of this semester. When performing literature review, we came across multiple solutions to this problem. Most of the problems addressed in tracking and finding a non-moving target which is the first step of our project. We used *Mobile Robot Simulation Toolbox* offered by *MATLAB*, which gave us few insights in how a differential drive model is constructed and how probabilistic path planning is used in tracking a non-moving target.

In probabilistic path planning algorithm [5] the algorithm 2 continuously plans a feasible path in the environment. A random map is generated and later on connected to the neighbouring nodes and further queried by Dijkstra's shortest path algorithm. In the demonstration, you could see the robot following a path which is not a straight line between two points, which is due to the number of nodes that given to generate the world random map. Similarly, When the robot is tracking a moving target, it almost follows a straight line as the distance between the goal robot(the robot that moves randomly in the environment moving from a start to a target) and tracking robot (the robot that is being controlled) is really small and can be connected through one or two edges of the road map. In probabilisitic robotics, the robotic system is

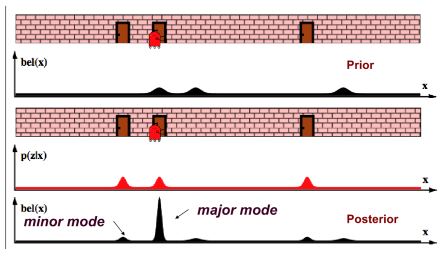


Fig. 2. Robot movement [6]

considered to of gaussian in nature. Here 2, starts with equal prior, and as the robot makes measurements and understands its surroundings, the robot localization converges to its current position of the robot.

The kalman filter algorithm is one of the effective algorithms used in the field of robotics. The following algorithm shows the steps involved in estimating the state μ of the robot explained by the state transition (motion of the robot),

where A_t is matrix that describes how a state evolves from t-1 to t without controls and noise, B_t is matrix that how a control changes u_t from t-1 to t, C_t is matrix describing how

```
$N \leftarrow \emptyset$
$E \leftarrow \emptyset$
loop
  $c \leftarrow$ randomly chosen free configuration
  $N_c \leftarrow$ a set of candidate neighbours of $c$ chosen from $N$
  $N \leftarrow N \cup c$
  for all c $\leftarrow$ randomly chosen free configuration do
    if same connected component $\Delta(c, n)$ then
      E $\leftarrow$ E $\cup$ (c,n)
      update R's connected components
    end if
  end for
end loop
```
Algorithm 1: Probabilistic Path Planning algorithm [5]

```
$\bar{\mu_t}$ = $A_t \mu_{t-1}$ + $B_t.u_t$ {Prediction steps}
$\bar{\Sigma_t} = A_t \Sigma_{t-1} {A_t}^T + R_t$
$K_t = \bar{\Sigma_t} {C_t}^T (C_t \bar{\Sigma_t} {C_t}^T + Q_t)^{-1}$ {Update steps}
$\mu_t = \bar{\mu_t} + K_t(z_t - C_t \mu_t)$
$\Sigma_t = (I - K_t C_t)\bar{\Sigma_t}$
```
Algorithm 2: Kalman Filter algorithm [6]

to map a state μ_t to an observation z_t. The noise in the system is assumed to be gaussian nature.

V. Proposed solution and Numerical Validations

In this section before presenting the solutions and numerical validations, we would like to discuss on various trials and experiments that we conducted before reaching to the final solution. We present the details of few notable experiments.

A. Formation control

One of our primary objectives was to maintain formation. In our project, we decided to go with two basic formation (Diamond and V). The following steps are followed to bring the robots in formation and maintaining them,

- Number of robots and choice of formation are received from the user
- Depending on the choice of formation, the robots are indexed
- The robot with index 1 takes the lead and other robots come into formation at a particular angle(Diamond - 30 °for two robots adjacent to leader, V -shape -30 °in both direction with respect to index 1 robot) and distance(1 unit - diamond formation, 0.7 unit - V -shape)
- With these indexes, higher indexed robots holds the formation by following the lower indexed robots by maintaining the above mentioned parameters.

B. Experiment 1

This experiment involves building a path planning robot. In this we use the system that is mentioned in III. The differential drive robot model was designed in *MATLAB* using the Mobile robotics toolbox. As mentioned in IV, the path is planned as graph, and panned out as the algorithm (Dijkstra) progresses in search of the goal. We present here a short video VII-A which shows the probabilistic path planning and formation control.

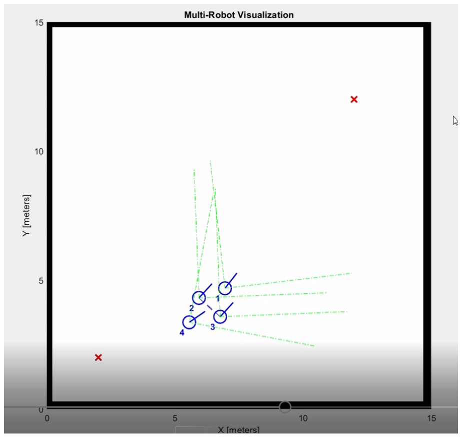


Fig. 3. Diamond formation

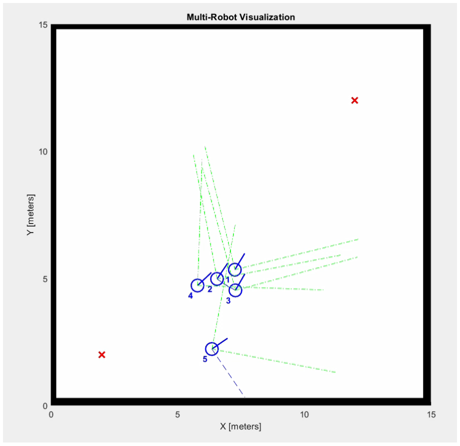


Fig. 4. V-shape formation

C. Experiment 2

This is an experiment carried out to track a moving object using probabilistic path planning. Our findings include increased computation time in following and keep track on moving object. In this experiment, we changed the target of the follower robot from a fixed target mentioned in the earlier experiment and fed it with the position of the randomly moving object VII-A. In 6 presents the model of object tracking which tracks the moving object, where the object tracking logic follows the algorithm by,

1) Getting the label to be tracked from user (color of robot)
2) Look for the received label through the sensors and perceive the distance
3) If the tracked robot(the robot that is being tracked) is away by more than 1 unit away from the tracking robot (the robot that is in user's control), move towards the robot at 1.5 units per second

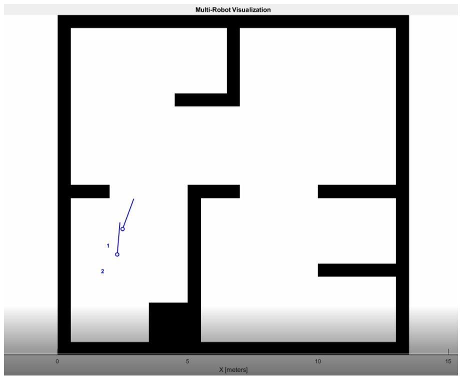


Fig. 5. Moving object Tracking

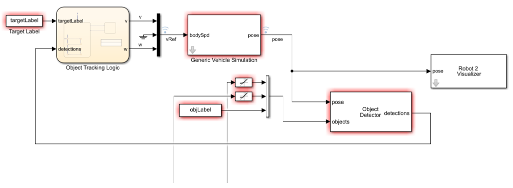


Fig. 6. Moving object Tracking model [7]

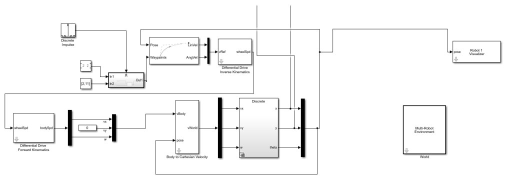
4) If the tracking robot is facing other way from the robot then robot performs rotational motion at an angular speed of 1.5 °per second
5) Once the robot finds the tracked robot, move to step 3.

D. Experiment 3

In this experiment, we simulated an almost real world scenario, by adding obstacles and noise to the system. In this as seen in VII-A the robots came into its formation, and the measurement gaussian noise was too high with a mean of 0.7. Due to this the robot indexed 2 could not overcome the obstacle placed in its path, and tries to move along the contour of the obstacle. From these experiments, we were able to understand the concept of swarm robotics and how they work together and coordinate themselves to complete the ask. In deployment of these solutions in real world, they can be implemented with further improvement and study in moving object tracking.

E. Stability

In any control system study, stability of the system is a major concern before being deployed into the real world


Differential Drive Inverse Kinematics
Differential Drive Forward Kinematics
Body to Cartesian Velocity
Discrete
Robot 1 Visualizer
Multi-Robot Environment
World
Fig. 7. Moving object Tracking model contd. [7]

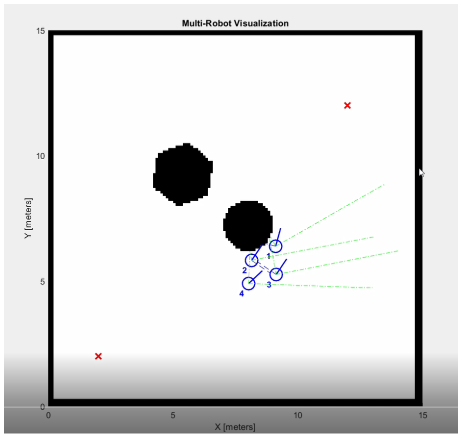


Fig. 8. Experiment with obstacle in the environment

environment. The algorithm (probabilistic path planning) implemented to control the path followed by the robots is based on probability which sometimes can lead to instability by moving/planning a path close to an obstacle which can endanger the robot into a collision course. Regarding the formation stability of the system, the instability of the system can be caused by the sensor measurements, which might result in breaking of formation, collision with partner robots. In case of internal stability, from III the robot position and velocity are determined. This can get into a wobbling state when the sensor noise is extreme which continuously changes the the values of x, y, θ.

F. Robustness

Robustness is expected to be implemented in any type of system. Here, such robustness can be seen by the ability of the swarm to avoid an obstacle and continue/ re-route the robots in a different path and continue with the formation after passing through the obstacles. Internal robustness can be defined as the ability of the robot to understand its environment even in a noisy environment/sensor and perceive its surroundings to achieve its goals. In our project, the swarm robustness was maintained to some extent by operating in formation even in noisy environment 4.

G. Non-linearity

In this group system, the non-linearity can be caused due to the sensor noise, communication error. When non-linearity is introduced into the system through limiting sensor range, noisy sensors, swarm showed different behaviour by non-cooperative swarm behaviour, unpredictable path following and breaking from formations. Such non-linearity was tackled to some extent using Kalman filter was not able to achieve complete control over the non-linearity scenario.

VI. Future work and Conclusion

In conclusion, The designed formation algorithm can be implemented in limited obstacle and predictable environment(warehouse, logistics) this project can be extended to

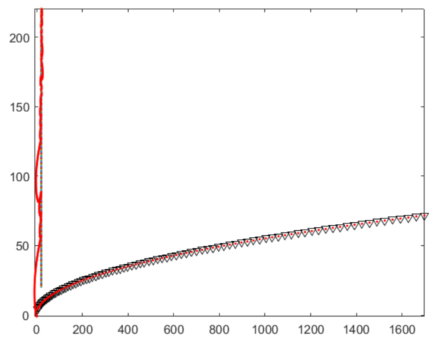


Fig. 9. Implementation of Kalman filter in noisy environment

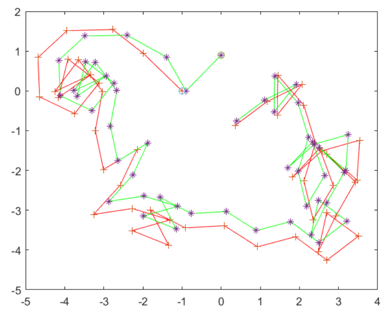


Fig. 10. Implementation of Kalman filter

being implemented in automated cars which will require further investigation into the problem in bringing in rules, regulations and other safety features into the controller design of the cars as implemented during the ENGN 6224 course.

VII. Appendix

A. *Videos*

The videos are arranged as experiments 1,2 and 3 in a chronological order

References

[1] J. Kennedy and R. Eberhart, ”Particle swarm optimization,” Proceedings of ICNN’95 - International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4.

[2] S. Sengupta, S. Basak, and R. Peters, Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives, Machine Learning and Knowledge Extraction, vol. 1, no. 1, pp. 157191, Oct. 2018.

[3] Dang, A.D., La, H.M., Nguyen, T., Horn, J., Distributed Formation Control for Autonomous Robots in Dynamic environments, extracted from: https://arxiv.org/ftp/arxiv/papers/1705/1705.02017.pdf

[4] Macaulay, T., Chapter 12 - Threats and Impacts to the IoT, RIoT Control, Morgan Kaufmann, 2017, Pages 221-278, ISBN 9780124199712, https://doi.org/10.1016/B978-0-12-419971-2.00012-1 (http://www.sciencedirect.com/science/article/pii/B9780124199712000121)

[5] Kavraki, L. E., Svestka, P., Latombe, J. C., Overmars, M. H. . *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. IEEE transactions on Robotics and Automation, 12(4), 566-580, (1996).

[6] R. Mahony, V.Ila , *ENGN 6627 Robotics*, Lecture slides, Semester 2, 2018.

[7] Mathworks Inc., *MATLab Documentation*, 2019.