University of Toronto

MIE Y8888 Research Project

Implementation of a Multi-agent Coordination and Motion Planning Protocol

Author:
John Pineros

Course Coordinators:
Hugh Liu

July 25, 2020

Contents

List of Figures

List of Tables

1 Introduction

1.1 Objective

The goal of this project was to implement the multi-agent motion planning protocol presented in [1] in both MATLAB and ROS. The results from the MATLAB simulation were used as a baseline and compared to the performance of the same protocol implemented within ROS which is acting in place of actual hardware. Based on the result of these two simulations recommendations for future work were made.

1.2 Scope

For this project the focus for the implementation was on 2D motion planning. The motion model for the agent used was the unicycle robot with linear and angular velocity as the control inputs. For state disturbance we looked at wind in the x and y axis. A focus was made on ensuring the simulation in MATLAB could simulate as many agents as possible and to ensure the critical inter-agent distance d'_m was maintained. The focus of the ROS implementation was not to have a large number of agents but to use a successful configuration from the MATLAB results and implement it on simulated hardware. By implementing the simulation in ROS we could determine what potential limitations could be found which were not present in MATLAB. The main paper used as reference for this project was [3].

2 Simulation Implementation

2.1 Modeling

The following section focuses on all the assumptions and mathematical models used to set up the multi-agent motion planning protocol. The specific details of how the motion planning protocol was implemented, depending on the simulation environment, are described in further detail in subsection 2.2.

2.1.1 Unicycle model

The agent state definition used was the unicycle model for both the state configuration and motion model. The 2D unicycle model was defined as:

$$q = [x, y, \theta]^T \tag{1}$$

With a motion profile having the following control inputs u_1 and u_2:

$$\dot{q_i} = [\cos\theta_i, \sin\theta_i, 0]^T u_1 + [0, 0, 1]^T u_2 \tag{2}$$

The definition used for the control inputs u_1 and u_2 was defined in section 2.1.5. The wind disturbance was modeled using the definition in [2].

$$w(x, y) = [w_x, w_y, 0]^T \tag{3}$$

The noise term has the properties of a Gaussian model, white noise with a known mean. The noise covariance P_w, $\in \mathbb{R}^{2x2}$ was constant and bounded. Having a bounded noise covariance was important for proving the stability of the EKF based observer model [2]. Since the noise term is described by a Gaussian white noise process we assumed that for every time step noise term was treated as a constant. The perturbed reference field uses the average noise term at a given point in space, given this information we assumed the partial derivatives used for $\dot{\varphi_i^p}$ are only a function of the linear velocity and the normalized reference field.
Along with the vehicle model we needed to define the class of agent we used in the simulation. From [1] we used the definition for class A agents since agent i needed to have access to the state information q_j of

any agent which is within its' sensing region C_i. Neighbour state information is necessary because of the bump function σ_{ij} defined in section 2.1.4. Another class of agent was also defined in [1] called class B agents. Class B agents do not have access to the state information of agents around them are also not considered in this report.

2.1.2 Reference frame definition

For mobile robots we also have the body rates $[p, q, r]$ which are body frame aligned [6]. They represent the angular rotations for each of the principal axis. For the simulation implementation the inertial frame and body frame have the axis definition shown in Figure 1. We are studying motion in the xy plane with the orientation angle θ, and an angular rate $\dot{\theta}$ which we assume is equal to the body rate r. This simplification is done because of the lack of rotation around the x and y axis.

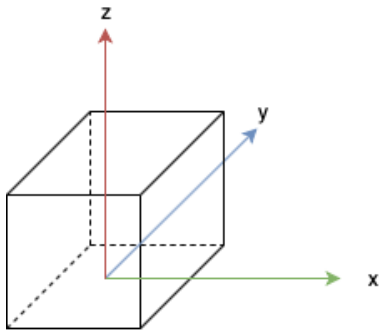


Figure 1: Mobile robot body frame definition.

2.1.3 Measurement model

For each agent the state measurement model used was defined as follows:

$$z_i = q_i + v_i \tag{4}$$

With $v_i(t)$ representing the measurement noise with properties of White noise process, zero mean, and a bounded covariance term P_{vi} [3].

2.1.4 Reference field derivation

The reference field equations used to attract or repel an agent from a given location were derived from the concept of electric dipoles. Electric dipoles are defined as a pair of point charges with equal charge but opposite sign. We define Q as the value of the point charge and the vector r as the 2D location of the center for the point charge. With this information we can define an ideal electric dipole as:

$$F(\mathbf{r}) = Q(\mathbf{r}_{Q_1} - \mathbf{r}_{Q_2}) \tag{5}$$

The derivation of the electric dipole flow fields was described in detail in [5]. The motivation for using electric dipoles is based on the partial derivatives of the flow lines which converge to the origin of the electric dipole [5]. The properties of a dipole like function where used to derive the general reference equation:

$$\mathbf{p} = \lambda(p^T\mathbf{r})\mathbf{r} - p(\mathbf{r}^T\mathbf{r}) \tag{6}$$

The parameter λ was introduced in [4] and it was used to describe if the reference field equation described an attractive field, for which $\lambda = 2$, or repulsive for which $\lambda = 0$. A modification to the repulsive field was done in [1] and it's the definition used in this simulation. The repulsive field equation was redefined as:

$$\mathbf{F}^i_{xoj} = \frac{(x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{7a}$$

$$\mathbf{F}^i_{yoj} = \frac{(y_i - y_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \tag{7b}$$

Where $[x_j, y_j]$ defines the global location of the object j being repelled. The attractive field definition is the one presented in [2] and has the following definition:

$$\mathbf{F}^r_{gix} = \frac{-(x_i - x_{gi})}{(x_i - x_{gi})^2 + (y_i - y_{gi})^2} \tag{8a}$$

$$\mathbf{F}^r_{gix} = \frac{-(y_i - y_{gi})}{(x_i - x_{gi})^2 + (y_i - y_{gi})^2} \tag{8b}$$

The goal and starting locations for agent i must also be selected such that the agent to agent distance must not be smaller than d_m [3]. The final parameter is the bump function σ_j which is dependant on a set of critical distances for class A agents. σ_j has the definition presented in [1]:

$$\sigma_{ij} = \begin{cases} 1 & \text{if } d_m \leq d_{ik} < d_r \\ a{d_{ij}}^3 + b{d_{ij}}^2 + cd_{ij} + d & \text{if } d_r \leq d_{ik} \leq d_c \\ 0 & \text{if } d_{ik} > d_r \end{cases}$$

where the value of d_{ik} is defined as the global distance between agent i and agent k. Agent k is defined as a critical agent which is inside of the sensing radius R_c but also meets the condition $k \in \{N_i | J_k < 0\}$. In [2] J_k was defined as:

$$J_k = r^T_{ki}\eta_i \tag{9}$$

The coefficients which make up condition 2 for σ_{ij} are defined as follows:

$$a = -\frac{2}{(d_r - dc)^3} \tag{10a}$$

$$b = -\frac{3(d_r + d_c)}{(d_r - dc)^3} \tag{10b}$$

$$c = -\frac{6d_r d_c}{(d_r - dc)^3} \tag{10c}$$

$$d = -\frac{d_c^2(3d_r - d_c)}{(d_r - dc)^3} \tag{10d}$$

Where d_c has the relationship $d_c \leq R_c$ and $d_m < d_r < d_c$. The parameter d_m is a safe minimum distance which relates radius ρ with a safety parameter ρ_ϵ which is a safety boundary that agent i must keep w.r.t an obstacle in space [1].

Having defined the attractive field, repulsive field and bump function the last thing is the modification to the global reference field equation F^*_i presented in [1] to the new definition in [3] which was used in the MATLAB and ROS implementation.

$$\boldsymbol{F_i} = \prod_{\boldsymbol{j \epsilon N_i}} (1 - \sigma_{ij}) \boldsymbol{F^r_{ij}} + \sum_{\boldsymbol{j \epsilon N_i}} \sigma_{ij} \boldsymbol{F^i_{oj}} \tag{11}$$

2.1.5 Observer model and input velocity calculation

One of the components of the multi-agent motion planning protocol is the ability to compensate for an external state disturbance defined in the unicycle model section. The EKF based observer model, first presented in [2], is used to predict the future state of the current agent based on the previous control input u_i and ω_i. From the EKF based observer model the important value for the system control input was the predicted pose rate $\dot{\hat{q}}_i$ which needed to be integrated to $\hat{q}_i$. With the integrated predicted pose $\hat{q}_i$ the linear velocity coordination [2] was defined as:

$$u_i = \begin{cases} -\frac{1}{\mu}\log(\sum_{j\epsilon N_i | \hat{J}_k < -\delta_J} e^{-\mu u_{i|k}}) & \text{if } d'_m \leq \hat{d_{ik}} < d_\epsilon \\ u_{ic} & \text{if } d_\epsilon \leq \hat{d_{ik}} \end{cases} \tag{12}$$

The key component of the linear velocity control law is the distance $\hat{d_{ik}}$. In global coordinates $\hat{d_{ik}}$ is the distance between agent i and a critical agent k. If there is an erroneous pose estimate from the EKF based observer then it's possible for the linear velocity to be calculated incorrectly causing agent i and agent k to potentially collide. A visual representation of the distance $\hat{d_{ik}}$ is shown in Figure 2. For agents which are outside of the critical distance d_ϵ the linear velocity was calculated by scaling the gain K_{ui} as follows:

$$u_i^n = K_{ui} \tanh\left(\|\hat{\boldsymbol{r}}_{\boldsymbol{i}} - \boldsymbol{r}_{\boldsymbol{gi}}\|\right) \tag{13}$$

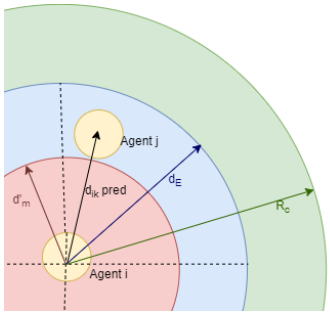


Figure 2: Agent i sensing radius definition for linear velocity calculation.

The angular velocity control input was calculated using:

$$\omega_i = -k_{\omega i}(\hat{\theta}_i - \varphi_i^p) + \dot{\varphi}_i^p \tag{14}$$

It is important to note that the variables φ_i^p and $\dot{\varphi}_i^p$ are calculated using the predicted state $(\hat{x}, \hat{y})$. This meant the reference field was calculated twice, once with the actual state of agent i used for the linear velocity and once with the predicted state $\hat{q}_i$ used for the angular velocity. The reference field was only affected by the average wind disturbance, $\overline{\boldsymbol{w_i}}$ in the definition of the perturbed reference field. The orientation of the perturbed reference field:

$$\varphi_i^p = \arctan(\frac{F_{niy}^p}{F_{nix}^p}) \tag{15}$$

with the perturbed reference field, calculated at the predicted state $\hat{q}_i$, given as:

$$F_i^p = \frac{u_i^n \boldsymbol{F_i}}{\|\boldsymbol{F_i}\|} - \overline{\boldsymbol{w_i}} \tag{16}$$

and the normalized perturbed reference field given as $F_{ni}^p = \frac{F_i^p}{\|F_i^p\|}$. The time derivative for the orientation of the perturbed reference field has the following:

$$\dot{\varphi}_i^p = (A_y^T.F_{nix}^p - A_x^T.F_{niy}^p)u_i \tag{17}$$

Where $\boldsymbol{A}$ is the gradient of the perturbed reference field:

$$A_x = \begin{bmatrix} \frac{\partial F_{nix}^p}{\partial \hat{x}} c\hat{\theta}_i \\ \\ \frac{\partial F_{nix}^p}{\partial \hat{y}} s\hat{\theta}_i \end{bmatrix} \tag{18a}$$

$$A_y = \begin{bmatrix} \frac{\partial F_{niy}^p}{\partial \hat{x}} c\hat{\theta}_i \\ \\ \frac{\partial F_{niy}^p}{\partial \hat{y}} s\hat{\theta}_i \end{bmatrix} \tag{18b}$$

The analytical calculation for the normalized perturbed reference field was calculated as:

$$\frac{\partial F_{nix}^p}{\partial \hat{x}} = \frac{\frac{\partial F_{ix}^p}{\partial x}(F_{ix}^{p\,2} + F_{iy}^{p\,2})^{1/2} - \frac{1}{2}F_{ix}^p(F_{ix}^{p\,2} + F_{iy}^{p\,2})^{-1/2}(2F_{ix}^p \frac{\partial F_{ix}^p}{\partial x} + 2F_{iy}^p \frac{\partial F_{iy}^p}{\partial x})}{F_{ix}^{p\,2} + F_{iy}^{p\,2}} \tag{19a}$$

$$\frac{\partial F_{nix}^p}{\partial \hat{y}} = \frac{\frac{\partial F_{ix}^p}{\partial y}(F_{ix}^{p\,2} + F_{iy}^{p\,2})^{1/2} - \frac{1}{2}F_{ix}^p(F_{ix}^{p\,2} + F_{iy}^{p\,2})^{-1/2}(2F_{ix}^p \frac{\partial F_{ix}^p}{\partial y} + 2F_{iy}^p \frac{\partial F_{iy}^p}{\partial y})}{F_{ix}^{p\,2} + F_{iy}^{p\,2}} \tag{19b}$$

$$\frac{\partial F_{niy}^p}{\partial \hat{x}} = \frac{\frac{\partial F_{iy}^p}{\partial x}(F_{ix}^{p\,2} + F_{iy}^{p\,2})^{1/2} - \frac{1}{2}F_{ix}^p(F_{ix}^{p\,2} + F_{iy}^{p\,2})^{-1/2}(2F_{ix}^p \frac{\partial F_{ix}^p}{\partial x} + 2F_{iy}^p \frac{\partial F_{ix}^p}{\partial x})}{F_{ix}^{p\,2} + F_{iy}^{p\,2}} \tag{19c}$$

$$\frac{\partial F_{niy}^p}{\partial \hat{y}} = \frac{\frac{\partial F_{iy}^p}{\partial y}(F_{ix}^{p\,2} + F_{iy}^{p\,2})^{1/2} - \frac{1}{2}F_{ix}^p(F_{ix}^{p\,2} + F_{iy}^{p\,2})^{-1/2}(2F_{ix}^p \frac{\partial F_{ix}^p}{\partial y} + 2F_{iy}^p \frac{\partial F_{iy}^p}{\partial y})}{F_{ix}^{p\,2} + F_{iy}^{p\,2}} \tag{19d}$$

The derivation of the partial derivatives for the perturbed reference field, and by extension the reference field, are summarized in Appendix A.

2.2 Simulation Setup

For both simulations we are interested in meeting the safe inter-agent distance condition $d_{ik} > d'_m$ and the safe convergence to the desired goal location q_{gi}. During an initial implementation of the simulation in MATLAB the symbolic toolbox was used to solve the partial derivatives in equation 18. This posed a problem due to the significant increase in computation time caused by using a symbolic solution, specially when the simulation used more than 1 agent. Using a symbolic solver also meant that when the multi-agent protocol was transferred onto actual hardware an analytical solution would still need to be calculated. Due to both of these constraints an analytical solution was derived and implemented.

For both simulations it was assumed that agents could not slowdown to a $0(m/s)$ velocity. This assumption implies that agents who approach the goal location will always have a small linear velocity command. This condition had to be implemented due to the smooth approximation of condition 1 in equation 12. Depending on how close agent i is to the goal location it could slow down to a halt while other agents are navigating around it. In the case where agent i stops within the sensing radius of agent j, which is trying to get to its' goal location, the smooth approximation of equation 12 could cause agent j to slow down and potentially stop which is not desirable. The other factor was the state disturbance from wind. When agent i reached a goal location it still needed to counteract the random wind disturbance pushing it in a random direction. The value of wind was always a positive number which meant the

direction of the wind disturbance was always in the positve x, y direction. A small state command ensures the agent will always push towards the goal location.

The structure of the functions used in the simulation code follow the flow diagram in Figure 3. The splitting the reference field into sub-functions permits for rapid changes to the reference field calculations or to the reference field gradient while maintaining the main simulation the same. This structure was adopted for both the MATLAB and ROS simulations.

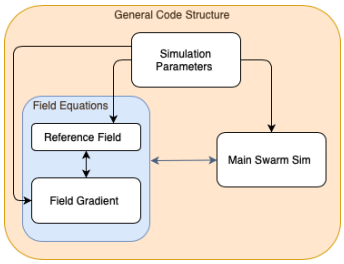


Figure 3: Simulation code function structure.

2.2.1 MATLAB

MATLAB was used as the first development environment to test out the multi-agent motion planning protocol. The structure for the simulation has the flow diagram in Figure 4. This structure was designed to study the computation time required per agent while leaving the pose update separate. A similar computation format would eventually be used for the ROS implementation. Once the main simulation was implemented the only script which needed to be changed defined the simulation parameters. This flexibility permitted a rapid testing of various numbers of agents with their own desired goal locations.

2.2.2 ROS

The ROS simulation was set up to emulate implementation on a real life agent. For the actual agent pose q_i ROS uses the location of the agent within the Gazebo environment. If this protocol was implemented on an actual robot the actual pose would need to be calculated using an external sensor to estimate the ground truth, such sensors could be a motion capture system or a GPS for outdoor use. For the purposes of this ROS implementation the measurement model in equation 4 was used. This assumption was used to maintain the same covariance value as the MATLAB simulation. The modification between the actual Gazebo pose and the wind disturbance used in the simulation is presented in Figure 5.

Having implemented the wind disturbance in this method does have one limitation. A physical wind disturbance would imply that if the agent did receive a linear velocity command of $0(m/s)$ the agent should still be physically moved by the wind based on the direction agent i is headed. This is not the case in this implementation and it is a simplification which was done in order to maintain parity with what the MATLAB simulation was set up to do. Once the pose measurement has been parsed and available for each of the agents the actual state and the measured state was passed onto a separate function which acted as the agent model. The agent model is presented as the top blue region in Figure 6. Decoupling the pose measurement adjustment from the agent motion model gives the flexibility for the measurement to come from a different source other than the modified pose from Gazebo. Similar to the MATLAB simulation the agent control input calculation is done within the first blue region while publishing the agent command is done in a separate loop. Splitting up the control calculation from the command publishing was done to ensure that each agent had the linear and angular velocities commanded for a roughly equal amount of time. In an early iteration of the simulation the command inputs for agent i where calculated and published within the same loop. This initial iteration caused unequal publishing time for the system command which meant the pose measurements between agents would be outdated.

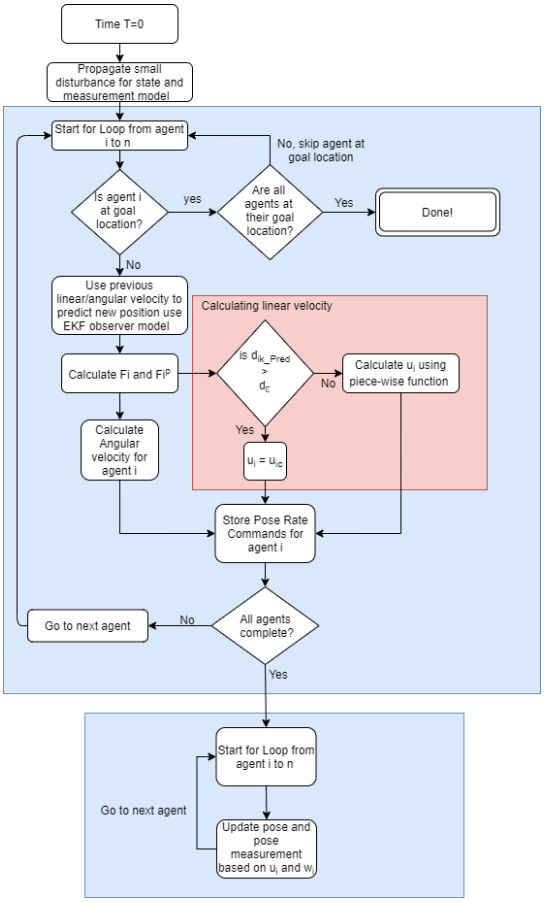


Figure 4: Simulation flow diagram for MATLAB.

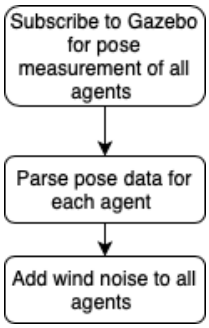


Figure 5: Gazebo pose measurement adjustment.

2.3 Results

In this section simulation results for both MATLAB and ROS are studied. MATLAB simulations allow us to quickly identify the conditions which lead agents to have difficulty converging to a desired goal location. ROS allow us to compare an ideal simulation predicted by MATLAB to what may occur with an actual robot. To maintain parity between the two simulation environments the linear and angular velocities of the agents where saturated to match the maximum limits allowed by the turtlebot2 mobile robotics platform [8] summarized in table 1.

When comparing results for both the ROS and MATLAB simulations the starting and ending locations summarized in table 2.

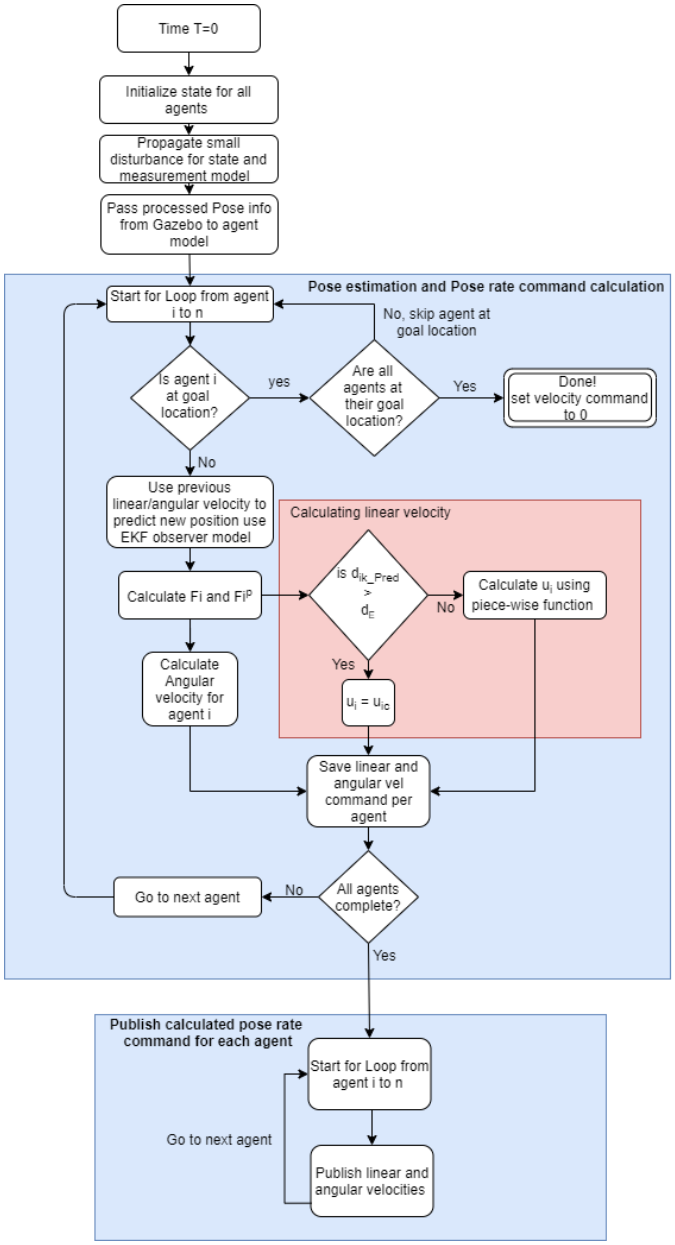


Figure 6: Simulation flow diagram for ROS.

2.3.1 MATLAB Results

For the MATLAB simulation the interest was in studying how effective the motion protocol is with large numbers of agents. Figure 7 shows the combined path taken by all 4 agents as well as the relative distance between agent 1 and the other 3 agents. From Figure 7d the critical distance d'_m was conserved by every agent which allowed them to reach the goal location without any collision.

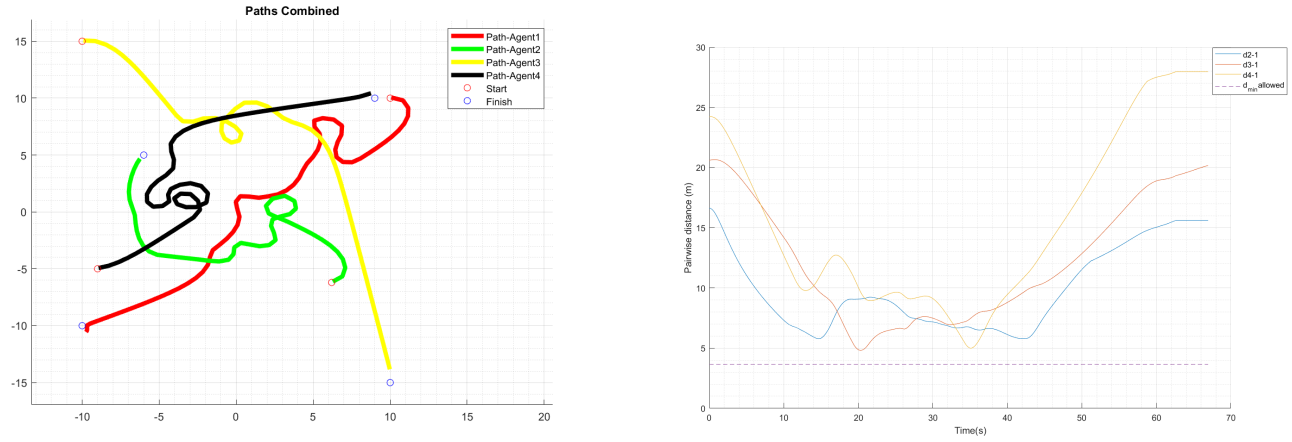
Once the 4 agent case was developed and shown to be successful the next step was to identify the maximum number of agents before the pose estimation error would cause the simulation to become unstable. Computation time was the parameter which we could monitor and give an indicator of which number of agents caused a simulation to fail. Computation time for a 1,2, 4 and 7 agent simulation was done. In table 3 the starting and ending location of all agents was summarized. The starting and ending pose was appended by the properties of the new agents. This ensured that when a working configuration stopped being successful the only change made would be the addition of newer agents.

	Maximum	Minimum (m)
Linear Velocity	0.7 m/s	-0.7 m/s
Angular Velocity	π rad/s	$-\pi$ rad/s

Table 1: Turtlebot2 velocity constraints

Agent index	x_0 (m)	y_0 (m)	x_g (m)	y_g (m)
1	10	10	-10	-10
2	6.2	-6.2	-6	5
3	-10	15	10	-15
4	-9	-5	9	10

Table 2: Start and end locations for all 4 agents



(a) Combined path for 4 agents from MATLAB simulation. (b) Relative distance between agent 1 and agents 2,3,4.

Figure 7: MATLAB simulation results for a 4 agent scenario

Agent index	x_0 (m)	y_0 (m)	x_g (m)	y_g (m)
1	10	10	-10	-10
2	6.2	-6.2	-6	5
3	-10	15	10	-15
4	-9	-5	9	10
5	-20	17	-10	-7
6	-20	25	-10	-15
7	-20	32	-10	-22

Table 3: Start and end locations for 7 agents

Figure 8 shows the impact on computation time as the number of agents in the simulation increases. With the current implementation going beyond 7 agents the computation time is already higher than the desired step size of $0.01(s)$. A configuration of more than 7 agents always resulted in at least 1 agent not being able to reach the goal location due to instability in the observer model pose prediction error. From experimentation when the magnitude of the prediction error is consistently greater than $0.1m$ the likelihood of the error covariance going unstable Was very high.

The limitation on the number of agents is due to the set up of the simulation. Recalling Figure 4 the linear and angular command calculation and pose update are done in separate loops. However, the

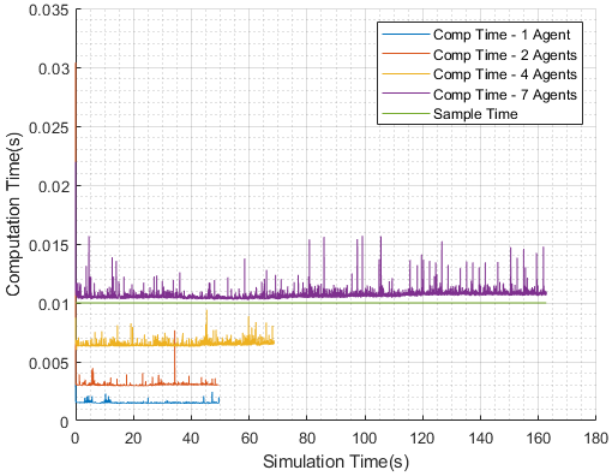


Figure 8: Computation time for 7 agent swarm.

calculation of the agent input commands is done inside of a for loop which means that the more agents where added the computation time also began to increase linearly. A trend like this can be seen in 8 where 2 agents have an average computation time of $0.003s$, and 7 agents have an average computation time of $0.011s$. The ideal set up has command calculation for all agents done simultaneously to increase efficiency. From Figure 8 the computation time for a single agent is roughly $0.0015s$ which is well below the desired step size.

As mentioned in the derivation section, an analytical solution for the reference field gradient was used in part to calculate the angular rate for each of the agents. We can see from Figure 9 that the angular rate is fairly low except for situations where the agent is really close to the goal location or the agent is trying to avoid collisions with its' neighbour. Having a low value for angular rate means the steady state assumption for lemma 1 in [1] is met. It's also important to notice the switching behaviour of the angular rate of the reference field, in particular the closer the agent gets to the desired location. To avoid the reference field functions to become undefined a pose error check $\boldsymbol{d_{error}} = \|[x_i - x_g, y_i - y_g]\|$ was used to check when $\boldsymbol{d_{error}} < 1.5m$. When this condition was true the agent was considered to be at the goal location. If this check wasn't implemented we can see in Figure 9 that the angular rate starts to go unstable causing the simulation to fail. The linear velocity is also not set to $0(m/s)$ at the goal location since there is still a wind state disturbance which would push the agent away.

2.3.2 ROS Results

The results of the ROS simulation show improvements in the computation time for each agent as well as a path which is not as sharp as the MATLAB simulation. The only difference not being simulated in MATLAB are mechanical properties of the TurtleBot2 such as moment of inertia about the yaw axis. It was also observed when a linear and angular velocity was published there was a small delay in the TurtleBot2 actually moving. This delay can also cause the simulation to be smoother since the TurtleBot2 does not react instantly to rapid changes in input commands. Similar to the MATLAB simulation the safe inter-agent distance d'_m was also maintained and the agents safely converged to a desired goal pose. Both of these results are summarized in Figure 10.

One interesting point to compare between the ROS and MATLAB simulations are the computation times taken by the 4 agent case. The ROS clock was used to generate a tik and tok counter to check how long the computation of input commands for all 4 agents as seen is Figure 6.Within the ROS environment the computation time for all 4 agents was around $0.002(s)$ which is roughly a third of the computation time taken by the MATLAB simulation. The downside of the simulation in ROS is the environment is

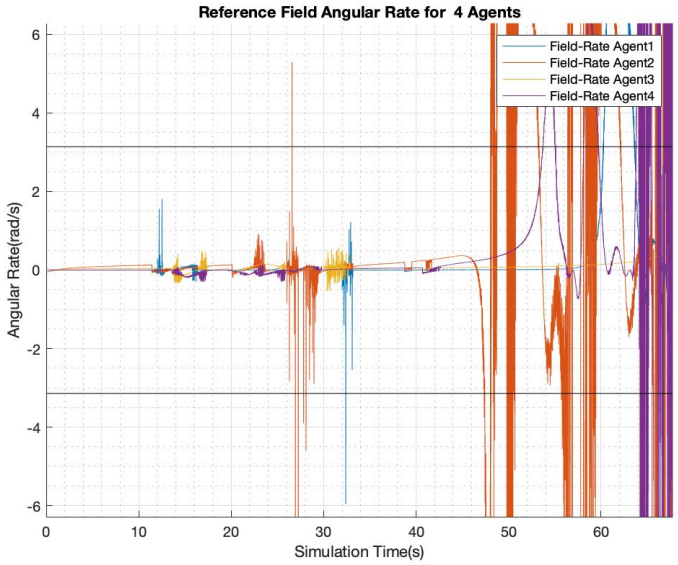
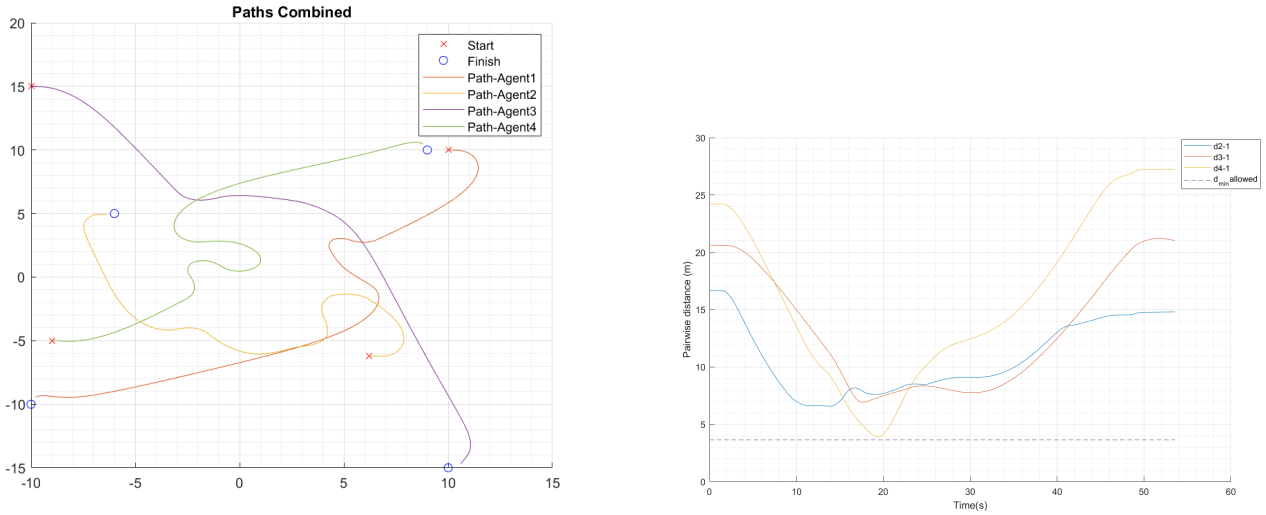


Figure 9: Reference field angular rate calculation 4 agent scenario



(a) Combined path for 4 agents from ROS simulation. (b) Relative distance between agent 1 and agents 2,3,4.

Figure 10: ROS simulation results for a 4 agent scenario

not as flexible when it comes to rapidly modifying the number of agents in the simulation. One area to be explored would be to compare the 7 agent case to see if there is a similar ceiling for computing the command for all agents.

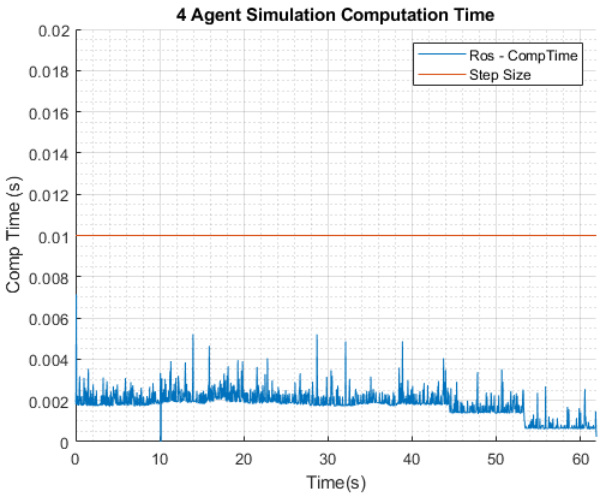


Figure 11: Computation time for 4 agent swarm in ROS.

3 Conclusion

3.1 Summary

In this project we studied the performance and scale-ability of the motion planning protocol presented in [1] by implementing it in both ROS and MATLAB environments. The upper limit on the number of agents for the MATLAB simulation was found experimentally while the ROS simulation was used to represent the algorithm implementation on actual hardware. Several simplifications such as agent yaw rate definition, reference field gradient calculation, and wind disturbance implementation were done in order to generate a baseline result of the motion planning protocol.

3.2 Future Works

From the experimental results the following areas for future work were identified. The concept of a connectivity radius was presented as an ideal region where agent i always has the state information for agent j, which is considered a critical if the condition $J_k < 0$ is met. Currently the work presented in [7] focuses on characterizing the noise from measurements between agents. In actuality connectivity has a noise component but a signal reliability as well which can be affected by the environment or failure in the hardware. A further improvement would be to add inter-agent connectivity loss as a potential disturbance and implement a fault tolerance algorithm to compensate for loss in connectivity. A potential refinement can be made to the repulsive component of the reference field by defining a new constant which quantifies what the status of a connection between agent i and agent k is.

A scenario not looked at was the extension of the reference field to 3 dimensions. A potential starting point is presented in [9] where a refinement can be proposed to their equation of the potential function which defines the shape of the gradient as a cone. One limitation with their proposed field function is the inability for the the agent to move above or bellow the field, only around it. By combining the reference field function in [9] with the properties presented in [5] a new reference field definition can be implemented.

Lastly, as mentioned in the ROS simulation setup portion of the report, the wind disturbance was added to the pose of the Gaszebo simulation. A true wind disturbance would act as a separate physical force on the mobile robot. To improve the quality of the simulation an actual wind model should be added which is constantly applying a disturbance on the ground robot.

4 Acknowledgement

Thank you Dr.Hugh Liu and the team at the Flight Systems and Control Labs for their feedback and recommendations during the project.

A Reference field equations

A.1 Partial Derivatives for Reference Field

The reference field was made up of two components, the attractive and the repulsive component. The partial derivative for the attractive field was:

$$\frac{\partial F_{gix}}{\partial x} = \frac{(x_i - x_{gi})^2 - (y_i - y_{gi})^2}{((x_i - x_{gi})^2 + (y_i - y_{gi})^2)^2} \tag{20a}$$

$$\frac{\partial F_{gix}}{\partial y} = \frac{2(x_i - x_{gi})(y_i - y_{gi})}{((x_i - x_{gi})^2 + (y_i - y_{gi})^2)^2} \tag{20b}$$

$$\frac{\partial F_{giy}}{\partial x} = \frac{2(x_i - x_{gi})(y_i - y_{gi})}{((x_i - x_{gi})^2 + (y_i - y_{gi})^2)^2} \tag{20c}$$

$$\frac{\partial F_{giy}}{\partial y} = \frac{(x_i - x_{gi})^2 - (y_i - y_{gi})^2}{((x_i - x_{gi})^2 + (y_i - y_{gi})^2)^2} \tag{20d}$$

The partial derivative for the repulsive field was:

$$\frac{\partial F^i_{xoj}}{\partial x} = ((x_i - x_j)^2 + (y_i - y_j)^2)^{-1/2}(1 - (x_i - x_j)^2((x_i - x_j)^2 + (y_i - y_j)^2)^{-1}) \tag{21a}$$

$$\frac{\partial F^i_{xoj}}{\partial y} = \frac{-1}{2((x_i - x_j)^2 + (y_i - y_j)^2)^{3/2}} \tag{21b}$$

$$\frac{\partial F^i_{yoj}}{\partial x} = \frac{-1}{2((x_i - x_j)^2 + (y_i - y_j)^2)^{3/2}} \tag{21c}$$

$$\frac{\partial F^i_{xoj}}{\partial x} = ((x_i - x_j)^2 + (y_i - y_j)^2)^{-1/2}(1 - (y_i - y_j)^2((x_i - x_j)^2 + (y_i - y_j)^2)^{-1}) \tag{21d}$$

A.2 Partial Derivatives for perturbed Reference Field

The perturbed reference field has the following partial derivatives:

$$\frac{\partial F^p_{ix}}{\partial x} = \frac{u^n_i(\frac{\partial F_{ix}}{\partial x}(F^2_{ix} + F^2_{iy})^{1/2} - F_{ix}\frac{1}{2}(F^2_{ix} + F^2_{iy})^{-1/2}(2F_{ix}\frac{\partial F_{ix}}{\partial x} + 2F_{iy}\frac{\partial F_{iy}}{\partial x}))}{F^2_{ix} + F^2_{iy}} \tag{22a}$$

$$\frac{\partial F^p_{ix}}{\partial y} = \frac{u^n_i(\frac{\partial F_{ix}}{\partial y}(F^2_{ix} + F^2_{iy})^{1/2} - F_{ix}\frac{1}{2}(F^2_{ix} + F^2_{iy})^{-1/2}(2F_{ix}\frac{\partial F_{ix}}{\partial y} + 2F_{iy}\frac{\partial F_{iy}}{\partial y}))}{F^2_{ix} + F^2_{iy}} \tag{22b}$$

$$\frac{\partial F^p_{iy}}{\partial x} = \frac{u^n_i(\frac{\partial F_{iy}}{\partial x}(F^2_{ix} + F^2_{iy})^{1/2} - F_{iy}\frac{1}{2}(F^2_{ix} + F^2_{iy})^{-1/2}(2F_{ix}\frac{\partial F_{ix}}{\partial x} + 2F_{iy}\frac{\partial F_{iy}}{\partial x}))}{F^2_{ix} + F^2_{iy}} \tag{22c}$$

$$\frac{\partial F^p_{iy}}{\partial y} = \frac{u^n_i(\frac{\partial F_{iy}}{\partial y}(F^2_{ix} + F^2_{iy})^{1/2} - F_{iy}\frac{1}{2}(F^2_{ix} + F^2_{iy})^{-1/2}(2F_{ix}\frac{\partial F_{ix}}{\partial y} + 2F_{iy}\frac{\partial F_{iy}}{\partial y}))}{F^2_{ix} + F^2_{iy}} \tag{22d}$$

References

[1] Dimitra Panagou, *A Distributed Feedback Motion Planning Protocol for Multiple Unicycle Agents of Different Classes.*, IEEE Transactions On Automatic Control, Vol 62. March 2017

[2] Dimitra Panagou, *Robust Semi-Cooperative Multi-Agent Coordination in the Presence of Stochastic Disturbances*, IEEE Transactions On Automatic Control, Vol 62. March 2017

[3] Dimitra Panagou, Kunal Garg, *A Robust Coordination Protocol for Safe multi-agent motion planning.*, University of Michigan Ann Arbour. Guidance, Navigation and Control Conference. January, 2018.

[4] Dimitra Panagou, Herbert Tanner, Kostas Kyriakopolous *Control of Nonholonomic Systems Using Reference Vector Fields*, IEEE Conference on Decision and Control and European Control Conference, December 2011.

[5] Dimitra Panagou, Herbert G, et.al, *Dipole-like fields for Stabilization of systems with Pfaffian Constraints*, IEEE International Conference on Robotics and Automation. May 2010.

[6] Hugh H.T.Liu , Peter Grant, etc.*AER1217 Lecture Notes*, University of Toronto Institute for Aerospace Studies, 2020

[7] Huang, M. and Manton, J.H. *Coordination and consensus of networked agents with noisy measurements: Stochastic algorithms and asymptotic behaviour*, SIAM Journal on Control and Optimization. Vol. 48, No.1, Feb 2009.

[8] Thomas L.Harman *ROS Robotics By Example*, Packt Publishing. 2016

[9] Xia Chen, Jin Zhang *The Three-dimention Path Planning of UAV Based on Improved Artificial Potential field in Dynamic Environment*, Fifth International Conference on Intelligent Human-Machine Systems and Cybernetics. 2013.