AI34 - CPS PROJECT FINAL REPORT

FINAL REPORT

Mostafa ABOUGHALIA
UTC Computer Engineering Student
Optimization apprentice engineer at Airbus D&S
mostafa.aboughalia@etu.utc.fr

Alexandre TAESCH
UTC Computer Engineering Student
Data scientist apprentice at Airbus Helicopters
alexandre.taesch@etu.utc.fr

January 05, 2024

ABSTRACT

In this project, we presents a detailed MATLAB code designed for controlling a fleet of mobile robots. The focus is primarily on three key aspects: navigation, localization, and communication. The code demonstrates how these components are integrated to achieve coordinated movement, accurate positioning, and effective communication among robots in a dynamic environment.

Keywords Autonomous Robotics · Multi-Robot Systems · Path Planning · Formation Control · Localization Techniques · Communication Protocols · Obstacle Avoidance · Quality of Service · Sensor Fusion · Adaptive Algorithms

Contents

List of Figures

Project Overview

The project focuses on the cooperative control of mobile robots and is split into two major parts:

1. **Navigation**: Involves the autonomous navigation of a robot leader between waypoints, designing a symmetric diamond formation for follower robots, and implementing an obstacle avoidance strategy.
2. **Localization**: Focuses on various robot localization methods, including odometric models and collaborative localization, with an emphasis on studying their accuracy and impact.
3. **Communication**: Examines real-time communication quality in terms of delay, reliability, and packet loss, addressing challenges like broken links and obstacles affecting communication range.
4. **Integration and Objectives**: Aims to integrate navigation, localization, and communication to develop a system capable of maintaining formation and navigating effectively despite various challenges.

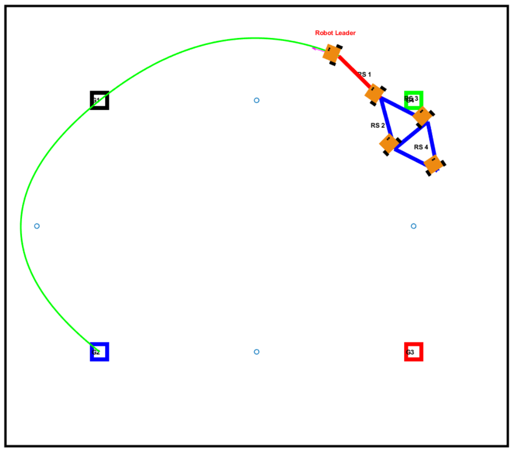

Figure 1: Final Trajectory and Diamond Formation Screenshot

1 Planning, control and decision-making for autonomous Multi-Robot Systems (MRS)

1.1 Smoother Navigation Between Waypoints

To achieve smoother navigation, the leader robot can implement a spline-based path planning algorithm. This method ensures continuous motion without stopping at each waypoint. The Catmull-Rom spline is suitable for this purpose.

```
% Calculate Catmull-Rom spline for smooth navigation
control_points = [previous_waypoint; current_waypoint; next_waypoint];
curvePoints = calculateCatmullRomCurve(control_points, numPoints);
% Navigate along the curve
for point = curvePoints'
    navigateTo(point);
end
```

This code snippet demonstrates how to generate a smoother path for the robot leader using Catmull-Rom splines. By interpolating between a series of waypoints, the leader can navigate with a more fluid motion, avoiding abrupt stops or sharp turns. This technique is particularly useful in environments where maintaining momentum is important, and it contributes to the efficiency and safety of the robotic system.

Several strategies were considered initially for smoother navigation. These included a Speed Limitation Strategy, Pre-calculating Turns with Centrifugal Force Formula, and Trajectory Pre-calculation for Two Waypoints. However, after careful analysis and consideration, the third option, which involves the use of a spline-based algorithm, was chosen.

The Catmull-Rom spline was selected for its simplicity and effectiveness in generating smooth trajectories. It utilizes a set of control points to form a curve that passes through each point. The mathematical foundation of this spline is based on the matrix multiplication of T, M, and G, where the matrix M is defined as follows:

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -2 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \tag{1}$$

This matrix is crucial in the calculation of the Catmull-Rom spline. It defines the curve's characteristics, ensuring a smooth transition between waypoints. The implementation of this spline algorithm significantly improved the leader's path navigation, providing a seamless and efficient route between waypoints.

1.2 Formation Control Implementation

In addition to defining the topology for the diamond formation, we also utilize Laplacian matrices to enforce constraints and manage the distances between robots. This approach not only ensures a precise formation but also provides flexibility in changing formations dynamically. Two primary formations have been implemented: diamond and line.

(i) Laplacian Matrices and Weighted Distances: For each formation, a Laplacian matrix and a corresponding weight matrix are defined. The Laplacian matrix represents the connections between robots, indicating whether the connections are uni- or bi-directional. The weight matrix, on the other hand, specifies the desired distances between the robots. This dual-matrix approach allows us to precisely control the formation while maintaining a predefined distance between each robot.

The Laplacian matrix for the diamond formation is given by:

$$L_{\text{diamond}} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 3 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix} \tag{2}$$

The corresponding weights for the diamond formation are:

$$W_{\text{diamond}} = \begin{bmatrix} 0 & d & 0 & 0 & 0 \\ d & 0 & 0 & 0 & 0 \\ 0 & d & 0 & d & d \\ 0 & d & d & 0 & d \\ 0 & 2\sqrt{2}d & d & d & 0 \end{bmatrix} \tag{3}$$

For the line formation, the Laplacian matrix is defined as:

$$L_{\text{line}} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{4}$$

And the weights for the line formation are:

$$W_{\text{line}} = \begin{bmatrix} 0 & d & 2d & 3d & 4d \\ d & 0 & d & 2d & 3d \\ 2d & d & 0 & d & 2d \\ 3d & 2d & d & 0 & d \\ 4d & 3d & 2d & d & 0 \end{bmatrix} \tag{5}$$

Using these matrices, we are able to exhibit not just the functional aspects of formation control, but also its versatility. The implementation allows for seamless transitions between the line and diamond formations, demonstrating the potential of our control algorithms in real-time scenarios. This flexibility is particularly useful for showcasing the capabilities of our robotic system during demonstrations, where the adaptability and precision of the formation control are evident (as seen in the attached video demonstrations).

1.3 Navigation Code Snippet (main.m)

This code snippet from `main.m` represents the core execution loop of the simulation for autonomous multi-robot systems. Within this loop, three primary operations are performed iteratively for each time step: updating the positions and orientations of the robots, managing the formation control using Laplacian matrices to maintain a predefined geometric structure, and navigating the robots towards designated waypoints. This loop forms the backbone of the simulation, orchestrating the movement and coordination of the robots throughout the process.

In order to enhance the smoothness of the trajectory, particularly for the leader robot, we have integrated the Catmull-Rom spline calculation, initially discussed in the smoother navigation section. This spline-based approach allows the leader robot to navigate fluidly through the waypoints, ensuring that the trajectory is not only precise but also exhibits a smooth and continuous motion. Such a smooth trajectory is crucial for maintaining the momentum and efficiency of the le

```matlab
% Main execution loop
for t = 1:iterations
    % Update robot positions and orientations
    % ... (position update logic)

    % Formation control using Laplacian matrices
    % ... (formation control logic)

    % Navigate towards waypoints
    % ... (waypoint navigation logic)
end
```

ader robot, especially in dynamic environments.

Furthermore, the followers are programmed to adhere to the formation defined by the Laplacian matrices. They continuously adjust their positions relative to the leader, ensuring the formation's integrity is preserved. This follower logic is critical for maintaining the geometric structure of the formation during the entire navigation process.

Additionally, the entire system is designed with obstacle avoidance capabilities. While the leader robot focuses on smooth navigation and reaching the waypoints, the followers are tasked with following the leader's path while simultaneously maintaining the formation and avoiding any encountered obstacles. This aspect of the simulation showcases the robustness and adaptability of our multi-robot system in complex environments.

In the next section, we will delve deeper into the obstacle avoidance strategy and how it is seamlessly integrated into our formation control and waypoint navigation framework.

1.4 Obstacle Avoidance Strategy

A critical component of our robotic system is the efficient and safe avoidance of obstacles. This is achieved through a hybrid approach that combines reactive and cognitive strategies, ensuring both immediate response to unforeseen obstacles and strategic path planning.

Reactive Obstacle Avoidance: The code snippet below demonstrates the reactive part of the obstacle avoidance strategy. This approach is immediate and reflexive, allowing the robots to quickly adjust their paths in response to detected obstacles. It is particularly useful for scenarios where unexpected obstacles appear suddenly in the robot's path.

```
% Update robots' states and control inputs
for i = 2:N
    robot_pos = x(1:2, i); % Current position of robot i
    robot_angle = x(3, i); % Current orientation of robot i

    [d_obs, d_obs_dot, nearest_obstacle_idx] = compute_obstacle_distance_and_derivative(robot_pos,
    ↪  obstacles);

    V = lyapunov_function(d_obs);
    V_dot = lyapunov_derivative(d_obs, d_obs_dot);

    if d_obs < 0.25
        % Robot is approaching an obstacle, steer away from it
        obstacle_pos = obstacles(:, nearest_obstacle_idx); % Get position of nearest obstacle
        angle_to_obstacle = atan2(obstacle_pos(2) - robot_pos(2), obstacle_pos(1) - robot_pos(1));
        turn_direction = sign(sin(robot_angle - angle_to_obstacle)); % Determine turn direction

        avoidance_speed = 0.5; % Set a speed for avoidance
        avoidance_turn_rate = 0.3; % Set a turn rate for avoidance

        % Update control input to steer away from the obstacle
        dxi(:, i) = [cos(avoidance_turn_rate); sin(avoidance_turn_rate)] * avoidance_speed;
    else
        % Maintain the formation
        % ... (formation control logic)
    end
end
```

This section of the code dynamically computes the distance to the nearest obstacle and, if the obstacle is within a threshold distance, calculates an avoidance maneuver. The robot then adjusts its trajectory accordingly, ensuring safe navigation. This method is based on a Lyapunov function, ensuring stability and robustness in the control strategy.

Alternative Approaches: While reactive obstacle avoidance is effective for immediate responses, other strategies could also be employed. One such alternative is predictive obstacle avoidance, where potential collisions are anticipated in advance based on the robot's velocity and trajectory, as well as the movement patterns of the obstacles. This approach requires more complex computations but can result in smoother trajectories and more efficient path planning in dynamic environments with moving obstacles.

1.5 Integrating Obstacle Avoidance into Catmull-Rom Spline for Leader Robot

In an effort to further enhance the autonomy and efficiency of our leader robot, we attempted to integrate obstacle avoidance directly into the Catmull-Rom spline calculation. This approach aimed to enable the leader robot to dynamically adjust its path in response to obstacles while following a smooth trajectory.

Modified Catmull-Rom Spline Algorithm: The core idea was to modify the traditional Catmull-Rom spline algorithm to consider obstacles in the environment. The following MATLAB code snippet demonstrates this approach:

```
function [curvePoints, waypointsIntermediaires, new_intermediate_waypoints] =
↪  getCatmullCurve(points, numPoints, obstacles, new_intermediate_waypoints)
    % ... (initial setup and calculations)

    while collisionDetectee
        % Loop to check and adjust for collisions
        % ... (collision detection and handling logic)

            if collisionDetectee
                % Recalculate the Catmull-Rom curve with adjusted waypoints
                [curvePoints, curve_x, curve_y] = calculateCatmullRomCurve(pointsTemp, numPoints);
                break;
            end
    end
end
```

This modification involved checking for potential collisions along the spline curve and dynamically inserting intermediate waypoints to avoid obstacles. The algorithm then recalculates the Catmull-Rom spline to include these adjustments, ensuring a collision-free path.

Advantages and Partial Success: The primary advantage of this approach is the enhanced fluidity and responsiveness of the leader robot's trajectory. It allows the robot to maintain a smooth and continuous motion while effectively navigating around obstacles. However, this modification achieved only partial success in our implementation. While it demonstrated improved obstacle avoidance in some scenarios, the complexity of dynamically adjusting the spline curve in real-time posed challenges.

Demonstration Video: To illustrate the functionality and limitations of this modified approach, we have included a demonstration video. The video showcases the leader robot navigating through an environment with obstacles, using the modified Catmull-Rom spline algorithm. It provides a visual understanding of how the robot adjusts its path in response to obstacles and the scenarios where this approach succeeds or faces challenges.

Through this ongoing research and development, we aim to refine this method further, enhancing its reliability and applicability in more complex environments. This exploration represents a significant step towards fully autonomous and adaptable robotic navigation.

1.6 Importance of Precise Localization and Communication

Accurate localization and reliable communication are crucial for maintaining the formation integrity and ensuring the efficiency and safety of the multi-robot system.

```
% Check localization and communication status
if isLocalizationAccurate() && isCommunicationReliable()
    maintainFormation();
else
    adjustStrategy();
end
```

The code snippet above illustrates the system's dependency on accurate localization and reliable communication. Precise localization is fundamental for each robot to understand its exact position and orientation within the group. This precision is vital for synchronizing movements and ensuring that the robots do not deviate from their intended paths, which could compromise the formation's structure or lead to collisions.

Reliable communication is equally important. It allows for the coordination of collective movements and the sharing of critical information among robots. This includes data on positioning, velocity, environmental sensing, and any adjustments required to maintain formation integrity. In scenarios where communication is compromised, the robots must be capable of adapting their strategies autonomously to prevent disarray or dysfunction within the system.

Implications of Localization and Communication Failures: Inaccuracies in localization or failures in communication can have significant consequences. They can lead to misalignments in formation, inefficient path following, increased risk of collisions, and overall reduced effectiveness of the robotic swarm. Therefore, the system is designed to continuously monitor these parameters and adjust its strategy accordingly. This adaptability ensures that the system remains functional and effective, even in the face of unforeseen challenges.

Illustrating the Impact: To illustrate the impact of precise localization and reliable communication, Figures 2 and 3 show the linear speed and acceleration of the leader robot under different conditions. These graphs highlight how the leader's movement characteristics can be affected by localization and communication factors, demonstrating their importance in real-time operation.

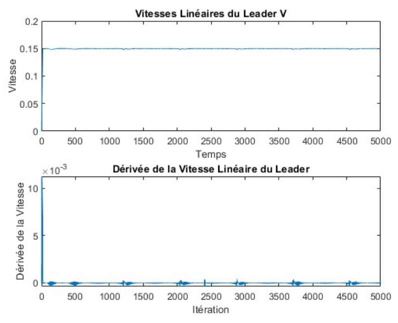

Figure 2: Leader Robot Linear Speed

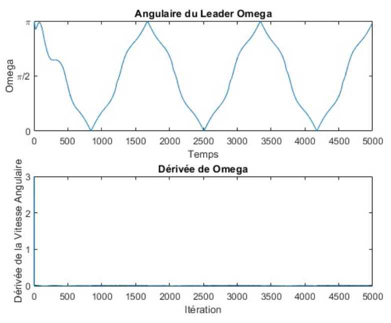

Figure 3: Leader Robot Acceleration

These visual representations underscore the direct relationship between the performance of the leader robot and the system's localization and communication capabilities. They serve as practical examples of how these factors are essential for the smooth and coordinated operation of autonomous multi-robot systems.

2 Localization of Autonomous Mono and MRS

- **Odometry and Prediction:** Initially, the robots' positions and orientations are updated based on their velocities and previous states. This is achieved through a prediction step using an odometry model.

```
A = [cos(odometer(3, t - 1, i)) 0; sin(odometer(3, t - 1, i)) 0; 0 1];
odometer(:, t, i) = odometer(:, t - 1, i) + A * [r.time_step * velocities(1, i);
↪  r.time_step * velocities(2, i)];
```

- **Sensor Update with Fault Detection:** The code then integrates sensor measurements (simulating LIDAR) to refine the localization. It also accounts for potential sensor faults by adjusting measurements based on predefined fault times.

```
z = [d1 + normrnd(0, 0.001); d2 + normrnd(0, 0.001); d3 + normrnd(0, 0.001); d4 +
↪  normrnd(0, 0.001)];
if find(tfault1 == t) z(1) = z(1) + 0.05; end
```

- **Information Fusion and State Update:** Finally, the code fuses the predicted state and sensor measurements to update the robots' estimated states. This involves calculating the information matrix and vector, and applying fault detection logic to isolate and exclude erroneous sensor data.

```
[r_global(t), detect] = detection(sgi, sgI, yp, Yp, X);
if detect ... % Exclude erroneous measurements
odometer(:, t, i) = podo * yu; % State update
```

This localization approach effectively combines odometry and sensor data, enhancing the accuracy of robot positioning. The inclusion of fault detection and isolation mechanisms ensures robustness against sensor anomalies, making it suitable for dynamic and potentially unpredictable environments.

2.1 Accuracy and Uncertainty in Localization Procedures

```
% Odometer model example for robot localization
previous_pose = [x; y; theta];
current_velocity = [v; omega];
delta_t = 1; % Time interval
new_pose = previous_pose + delta_t * [v * cos(theta); v * sin(theta); omega];
```

In the scenario where robots rely solely on their odometric models, localization accuracy decreases over time due to error accumulation, known as drift. This method, although straightforward, becomes less reliable in prolonged missions or complex trajectories. Conversely, when robots use external landmarks for localization, the accuracy typically improves. This method compensates for the drift in odometry by referencing fixed points in the environment, offering a more stable and reliable localization, especially in structured environments.

2.2 Handling Observational Errors

```
% Simulating observational error
landmark_position = [x_landmark; y_landmark];
observed_distance = norm(current_pose - landmark_position) + noise;
```

Observational errors, such as those from imprecise sensors or environmental factors, can lead to significant deviations in a robot's perceived position. To manage these inaccuracies, advanced filtering techniques like Kalman filters are often used. These filters process noisy sensor data and predictions from the robot's motion model to produce a more accurate estimate of the robot's location. By continuously updating this estimate with new observations, the system can maintain a high level of accuracy despite inherent sensor errors.

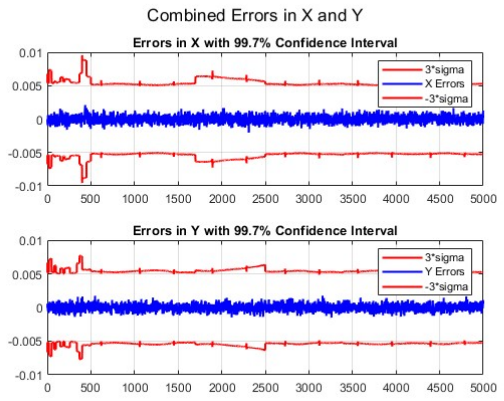

Figure 4: Combined Errors in X and Y

We illustrates that cumulative defaults, even in a brief time frame, can have a severe impact on the localization certainty, but can be contained thanks to the implementation of Kalman Filters and Informational Filters.

Analysis of the `detection` Function The `detection` function in MATLAB is crucial for anomaly detection in system monitoring. It operates by calculating a global residual and comparing it against a predefined statistical threshold.

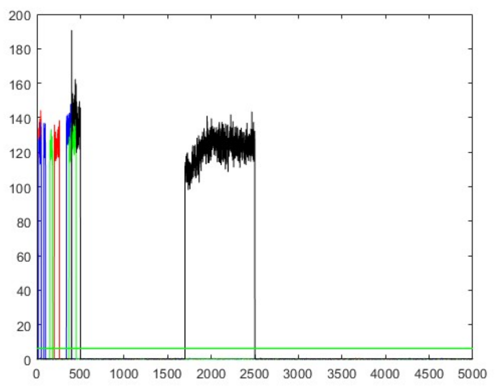

Figure 5: Residue per landmark

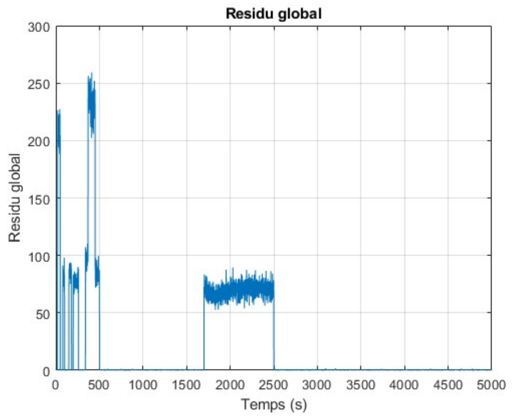

Figure 6: Global Residue

- **Residual Calculation:** The function takes inputs `sgi`, `sgI`, `yp`, `Yp`, and `X`, using them to compute the global residual `r_global`. This involves updating the prediction (`yu`) and the information matrix (`Yu`), followed by calculating an updated state estimate `Xkk`. The residual is essentially the weighted squared difference between the updated estimate and the previous state `Xkp`.
- **Threshold Definition:** A detection threshold is defined based on the chi-squared distribution with a 90% confidence interval. This statistical approach is essential for determining whether the residual indicates an anomaly.
- **Anomaly Detection:** The function assesses if `r_global` exceeds the threshold. If it does, an anomaly is detected (`detect = true`); otherwise, it is concluded that there is no anomaly (`detect = false`).

This methodology is pivotal in real-time system monitoring, particularly in scenarios like robotics or process control, where accurate state estimation is vital. It illustrates a statistical method for anomaly detection, using probability distributions to inform about the system's state and health.

3 Real-time Communication and IoT of Embedded Systems

3.1 Implementation of QoS Criteria

Quality of Service (QoS) criteria are essential in robotic systems for ensuring reliable communication and efficient task execution. The implementation of QoS criteria involves dynamically adjusting system parameters in response to varying environmental conditions and network states.

```
% Function to calculate communication delay
function delay = calculateCommunicationDelay(obstaclePresent, linkBroken)
baseDelay = 0.1; % Base delay without obstacles
if linkBroken
    delay = 3; % Delay for broken link
elseif obstaclePresent
    delta = 0.5; % Additional delay due to obstacle
    delay = baseDelay + delta;
else
    delay = baseDelay;
end
end
```

In the function above, communication delay is quantified based on the current network and environmental conditions. The `baseDelay` is a standard latency in a clear communication channel, while `delta` accounts for additional delay caused by obstacles in the communication path. A significant delay is introduced when a link is broken, reflecting the severe impact on the communication network.

Significance of QoS in Multi-Robot Systems: The QoS criteria, particularly in the context of communication delay, play a crucial role in multi-robot systems. They directly influence the coordination and synchronization among robots, especially in tasks that require precise timing and spatial arrangements. By quantifying and adapting to communication delays, the system can adjust its strategies for formation control, obstacle avoidance, and path planning, ensuring optimal performance even under suboptimal conditions.

Adaptive Strategy Based on QoS Metrics: The system's ability to adapt based on QoS metrics enhances its resilience and reliability. For instance, in scenarios where communication delays are high, the system might switch to more conservative strategies, such as reducing speed or simplifying formations, to maintain coordination. Conversely, when communication is fluid and uninterrupted, more complex maneuvers and faster movements can be executed.

This dynamic adjustment, governed by QoS criteria, is key to the robustness of our robotic system. It enables the robots to operate efficiently in a wide range of environments, from structured indoor settings to unpredictable outdoor terrains, making the system versatile and dependable for various applications.

3.2 Reliability and Packet Loss Rate Study

```matlab
% Function to calculate packet loss
function packetLossRate = calculatePacketLoss(totalPackets, lostPackets)
packetLossRate = (lostPackets / totalPackets) * 100;
end
```

The reliability of the communication system is assessed through the packet loss rate, calculated as the percentage of lost packets over total packets sent. This measure is vital for evaluating the robustness of the MRS communication system, particularly in challenging environments or scenarios where maintaining a constant stream of communication is crucial for operational success.

$$\text{Transmission Speed} = \left(\frac{\text{Distance R1–R2}}{\text{MaxRange}}\right) \times 10 \times \text{Transmission Time} \times \text{Energy}$$

This equation represents the transmission speed between two robots (R1 and R2), taking into account the distance between them, the maximum range of communication (MaxRange, fixed by us), the transmission time, and the energy consumption. The scaling factor '10' adjusts the transmission time and energy to align with real-world units.

In MATLAB, we tested this communication model by simulating different scenarios. We varied the distances between the robots (R1 and R2), adjusted the maximum communication range, and changed the energy parameters. By doing this, we could observe how changes in these variables affected the overall transmission speed, thus validating our model in different simulated conditions.

3.3 Impact of MRS Geometry Change on Communication

In multi-robot systems, the geometric arrangement of the robots can significantly affect communication efficacy. As the distances between robots vary, so too does the likelihood of communication challenges, such as increased delays and the potential for broken links.

```matlab
% Adjusting MRS geometry
desiredDistance = 3 * actualDistance;
% Recalculate communication parameters
updateCommunicationParameters(desiredDistance);
```

This MATLAB code snippet illustrates the process of adjusting the geometry of the MRS, in this case by increasing the distance between robots. The `updateCommunicationParameters` function recalculates the necessary communication parameters to accommodate this change. Such adjustments are crucial, as increasing the inter-robot distance can introduce new communication challenges.

Communication Challenges with Increased Distances: As the distance between robots in an MRS increases, the risk of communication disruptions also rises. This is particularly true in environments with obstacles, where the likelihood of line-of-sight disruptions and signal attenuation is higher. An increased distance can result in weaker signals and a higher chance of packet loss, as depicted in Figure 7.

Adaptive Communication Strategies: To mitigate these risks, adaptive communication strategies become essential. These strategies may include dynamically adjusting transmission power, employing more robust communication protocols, or utilizing relay robots to maintain an effective communication chain between distant robots.

Packet Loss Consideration: In our simulations, we have modeled a 5% chance of packet loss for each communication attempt between robots, as shown in Figure 8. This realistic approach helps us to understand and plan for the communication challenges that may arise in real-world applications of MRS.

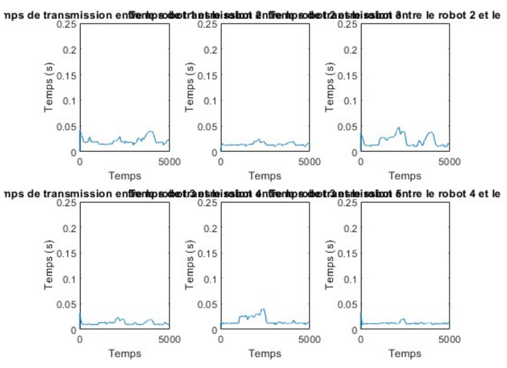

Figure 7: Transmission Time between robots

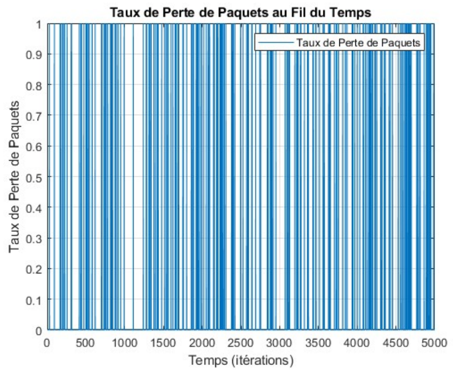

Figure 8: Packet Loss over time

These figures demonstrate the practical implications of geometric changes on communication within an MRS. Understanding and addressing these challenges is crucial for maintaining the reliability and efficiency of multi-robot systems, especially in complex environments where precise coordination is essential.

Prospective for the Communication Part

In our project, we have chosen Bluetooth Low Energy (BLE) as the primary communication technology for our small robots. This decision is based on BLE's ability to meet the diverse requirements of our robotic systems.

Advantages of Bluetooth Low Energy: BLE offers several benefits, crucial for the effective operation of our robots:

- **Low-Cost and Low-Power Consumption:** BLE is designed for minimal energy usage, essential for small robots with limited battery capacity.
- **Short-Range Communication:** With a range of approximately 10 meters, BLE is well-suited for operations within confined spaces or close formations.
- **Bandwidth:** Offering a bandwidth of up to 2 Mbps, BLE provides sufficient data transfer rates for the communication needs of our robotic systems.
- **Security Concerns:** BLE includes features to address security concerns, although continuous improvement in this area is crucial.

Challenges and Future Perspectives: While Bluetooth Low Energy (BLE) offers significant advantages, addressing its challenges is crucial for enhancing the Quality of Service (QoS) in our robotic communication systems. Our project has already made strides in this area by computing and plotting the packet loss rate and the distances between robots, providing valuable insights into the current communication performance.

- **Compute New Metrics for QoS:**
  - **Refine Packet Loss Analysis:** Building on our computed packet loss rate, we aim to delve deeper into minimizing packet loss to ensure even more reliable data transmission.
  - **Energy Consumption:** Continuously assessing and optimizing energy consumption is crucial. Our analysis could continue and include correlating energy usage with transmission patterns and distances between robots.
  - **Transmission Speed:** Ensuring adequate transmission speeds is vital. We could also leverage our distance plots to study how transmission speed varies with robot spacing and develop strategies to maintain optimal speeds.

In a next study, our focus would be on refining these aspects of communication, ensuring that our multi-robot systems can operate efficiently and effectively in a variety of environments and applications.

Efficiency

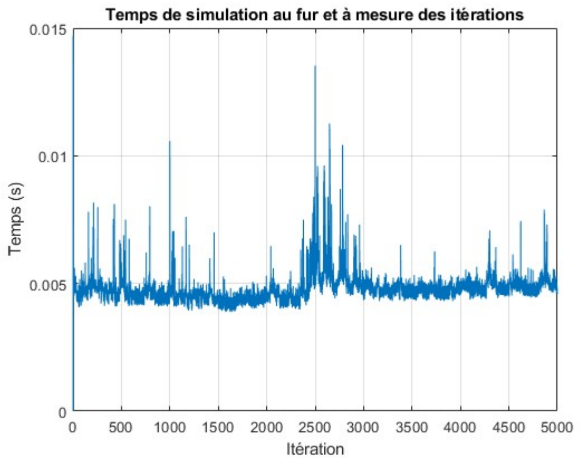

Figure 9: Simulation Time

Overall, our architecture demonstrates an efficient use of resources, which is advantageous considering that, in real-world applications, robots are often limited in their onboard computational capabilities. This efficiency enables our system to be effectively deployed in real-time scenarios where resource optimization is crucial.

Conclusion

In concluding our project on autonomous multi-robot systems (MRS), we've made significant strides in addressing the challenges of planning, control, localization, and communication. Our methods have enhanced the smooth navigation between waypoints, robust formation control, and precise localization techniques, incorporating both odometric models and landmark referencing. Moreover, our focus on quality of service in communication has underlined the importance of reliable and timely data exchange among robots, especially in obstacle-laden environments.

However, challenges remain, particularly in optimizing the balance between reactive and cognitive strategies for obstacle avoidance and further improving the reliability of communication under varying geometric formations. Future work could involve integrating machine learning algorithms for adaptive behavior in unpredictable environments, enhancing sensor fusion techniques for more accurate localization, and exploring advanced communication protocols to handle diverse operational scenarios. The goal remains to develop a MRS that is not only efficient and robust but also capable of operating autonomously in a wide range of complex and dynamic environments.