ROB 550 BotLab Report

Trushant Adeshara, Ruben Orsolle, Federico Seghizzi
{trushant, rorsolle, seghizzi}@umich.edu

***Abstract*—The goal of the Botlab project is to modify a provided codebase to control a 3 Degrees of Freedom (DoF) differential-drive mobile robot to autonomously move through a maze-like environment and detect and lift tag-labelled blocks. To achieve this, several PID loops were developed and tuned to control the motion of the wheels and the robot's ability to track a trajectory. To localize the robot and guide its navigation, dead-reckoning odometry was augmented with IMU readings, and combined with a particle-filter-based Simultaneous Localization and Mapping (SLAM) algorithm. This, combined with an exploration algorithm, provided occupancy grids used by an A* algorithm to generate trajectories to tag-labelled blocks detected via computer vision algorithms. A custom-designed forklift would then lift these and enable their transportation to predetermined locations.**

I. Introduction

The focus of the project lies in the development of an autonomous navigation infrastructure for a differential-drive mobile robot forklift. The robot's servo-motors provide it with three DoF and feature magnetic incremental encoders; the lidar instead provides a 2D scan of the surrounding environment; the camera finally provides an RGB image of the environment in front of the robot.

The autonomous navigation infrastructure should localize the robot in an unknown environment, autonomously navigate through it to explore and generate a map, and ultimately detect and navigate to tag-labeled blocks without colliding with obstacles. These blocks should then be transferred to their designated locations.

The implementation of the above behaviour can be split into four key components:

1) Low- and High-Level PID motion control through IMU-augmented odometry feedback.
2) Robot localization and environment mapping through particle-filter-based SLAM.
3) Trajectory planning via A* and environment exploration via frontiers detection.
4) Block detection and retrieval via april-tag recognition and forklift operation.

The different sections of the report will mirror this conceptual subdivision.

II. Methodology

A. Augmented Odometry and Motion Control

The robot's motion control consists of a high-level trajectory tracking algorithm that relies on augmented odometry to compute the forward and turning velocity commands. These are then passed through the wheel velocity controller that instead relies on the encoder readings to generate a PWM signal for the motors.

1) Augmented Odometry: The robot can estimate its position and consequently control its motion via simple dead-reckoning odometry. This consists of estimating the robot's $X - Y - \theta$ by comparing the readings from the encoder-derived velocity readings of the two wheels.

Preliminary results (Section III.A.1) revealed this method to be inadequate at estimating the robot's position and bearing correctly, with the latter being particularly problematic and causing error propagation throughout system operation. Consequently, these encoder-based readings were augmented with processed data from the IMU. Specifically, the odometry-based orientation θ of the bot was entirely replaced with the refined value generated from the IMU without any modifications.

2) Motor Control: In order to control the body velocity, we use two loops: a loop controlling the body velocities $\hat{v}$ and $\hat{\omega}$, estimated from the IMU-augmented odometry; an inner loop controlling the wheel velocities $\hat{\omega}_{R/L}$ obtained from the encoder. The conditional PID (or Cond PID) is a normal PD with a conditional integrator: this integrates only when the error is lower than 5% of the final value. Subsequently, the overshoot and the settling time are reduced while maintaining the static error null. Parameters were manually tuned until no static error (integral term) was present, whilst maintaining no or low overshoot and settling time around 0.5s. These are reported in Table I.

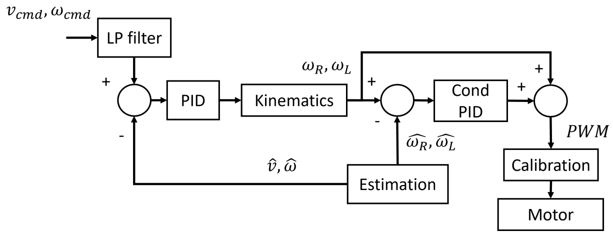

Fig. 1: Block Diagram of the Velocity Controllers

3) Trajectory Following: The implemented trajectory-following algorithm is a Rotation-Translation-Rotation (RTR) algorithm with PID control and capped output.

TABLE I: Parameters for the LP filter and PID controllers.

Filter/Controller	Parameters		
	K	T_i	
Low-Pass Filter	1.0	0.2	
	P	**I**	**D**
v	1.0	1.0	0
ω	1.0	1.0	0
$v_{L/R}$	1.0	0.2	0

Considering the RTR algorithm first, for each point this commands the bot to perform three motions:

1) Initial rotation of α to align with the direction of the next waypoint.
2) Translation of Δs to move from the current position to the next waypoint;
3) Final rotation of $\Delta\theta - \alpha$ to align the robot with the final waypoint' specified bearing.

To control the turn and forward velocities through these maneuvers, two proportional controllers were implemented: one with $K_p = 1$ for the forward movement, and one with $K_p = 3$ for the rotational movement.

To compensate for this simplistic control strategy, the velocity output was restricted to be within application-specific values. This prevented the output of excessively large commands from the P-controller and enabled enforcement of a speed limit through operation.

B. Simultaneous Localization and Mapping (SLAM)

The proposed SLAM algorithm is shown in Figure 2.

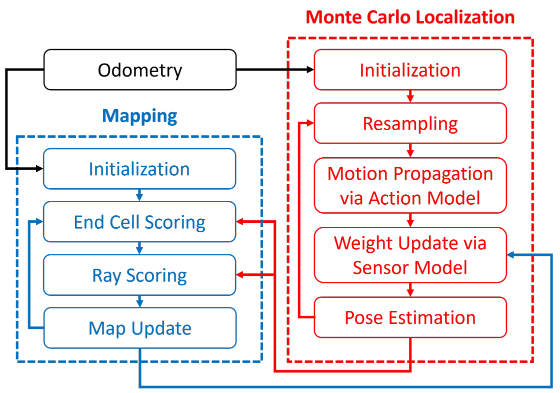

Fig. 2: High-level overview of the SLAM algorithm. Elements in blue relate to Mapping; Elements in red relate to Localization.

1) Mapping: The mapping algorithm employed by the robot builds a probabilistic occupancy map based on the logOdds of each cell being occupied or empty. Specifically, for each ray with a range lower than 5m (i.e. for rays that bounce off an obstacle and do not endlessly proceed into the void), the algorithm adds 7.5 to the logOdd value of the end cell, increasing its likelihood of being occupied. The algorithm then executes Bresenham's algorithm to determine the cells between the ray's source and the end cell, and subtracts 2 from each logOdd, reducing their occupancy likelihood.

To determine the source cell and end cell, the algorithm does need the robot's location. Initially, this is assumed to match the odometry's estimated location to create an initial map. Subsequently, the estimate from the localization algorithm is used.

Additionally, since the robot moves whilst the ray emitter spins, not all rays generated within a single rotation originate at the same location. Furthermore, the sensor readings are not processed at the same rate as the odometry ones. To account for this, interpolation is used to align all ray readings together with the odometry, enabling the correct formation of the map from the scans.

2) Monte Carlo Localization via Particle Filtering: Monte Carlo localization of the robot is achieved through a particle filter [1] . This is initialized by generating a specified number of equal-weighted particles either at a known start position, or uniformly distributed in the environment. The particles represent potential locations of the robot, with their weights indicating the likelihood of representing the actual location. Following initialization, the filter loops through the following steps:

a) Resampling: Via low-variance resampling [1].

b) Motion Propagation via Action Model: The model employed is an RTR model matching that described in Section II.A.3. To capture uncertainty in the motion performed, noise from a zero-meaned normal distribution is added to the RTR motion variables. Equation 1 encapsulates how the action model updates position estimates.

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} (\Delta s + \epsilon_2) \cdot c(\Delta\theta_t + \epsilon_1 + \alpha) \\ (\Delta s + \epsilon_2) \cdot s(\Delta\theta_t + \epsilon_1 + \alpha) \\ \Delta\theta_t + \epsilon_1 + \epsilon_3 \end{bmatrix} \quad (1)$$

The values ϵ_1, ϵ_2, and ϵ_3 are obtained via Equation 2.

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} \sim \begin{bmatrix} \mathcal{N}(0, k_1|\alpha|) \\ \mathcal{N}(0, k_2|\Delta\theta|) \\ \mathcal{N}(0, k_1|\Delta\theta - \alpha|) \end{bmatrix} \quad (2)$$

The values of the uncertainty parameters were determined by varying both k_1 and k_1 were varied between 0.01 and 0.05 in increments of 0.01. Each possible combination of values was evaluated by commanding the robot to perform a simple 1mx1m L-shaped motion. The selected values of $k_1 = 0.01$ and $k_2 = 0.3$ guaranteed a consistent distribution of particles that did not converge nor diverge excessively throughout the entire motion. This was measured by the maximum particle spread, defined as the maximum distance between two particles in the distribution.

c) Weight Update via Sensor Model: The weights of the propagated particles are updated via the sensor model to reflect the likelihood of being at the particle's pose given the lidar's reading and current map. Specifically, the weight is determined from the sum of each ray's score, assuming that the rays originate at the particle's position. Each ray is scored as follows:

- $score = 0$ if the ray's range is greater than 5m. This ensures that rays that do not intersect any obstacles and endlessly proceed into the void are not used for weighting the particle;
- $score = 0.1$ if the ray's end cell on the last-updated map is believed to be free;
- $score = 0.5$ if the ray's end cell on the last-updated map is believed to be occupied.

The weights of all the particles are finally normalized.

d) Pose Estimation: The re-weighted particles are used to estimate the robot's $X - Y - \theta$ by computing the weighted average the top 25% of particles' poses.

C. Planning and Exploration

1) Path Planning: Path planning through the SLAM-generated map is achieved via an A* algorithm, selected due to its good balance between efficient computation and optimal path generation. It achieves this by combining the cost functions of Dijkstra's Algorithm and Greedy Breadth-First Search. The former's cost, termed g cost, represents the "distance" from the start to the current node; the latter's cost, termed h cost, represents the distance from the current node to the goal. The two are defined in Equations 3 and 4. The last term of Equation 3 was added to account for the distance from the child node to the nearest obstacle after preliminary results reported in Section III.C.I showed the planner generating trajectories excessively close to obstacles.

$$g(\text{child}) = g(\text{parent}) + 1 - \text{dist}(\text{child}; \text{map}) \quad (3)$$

$$h(x_s, x_g, y_s, y_g) = dx + dy + (\sqrt{2} - 2)\min(dx, dy) \quad (4)$$

2) Exploration: The exploration algorithm is designed to move the robot to frontier regions of the map until none remain and the environment is fully captured.

Considering frontiers detection, this is performed after every map update via a connected-components search. Specifically, when a map cell has a neutral logOdd value of 0 but one or more of its neighbors do not, this is marked as a frontier cell; connected cells sharing this property are then gathered to form a single frontier.

The algorithm then loops through the list of frontiers to detect the closest one. The path planner described above is then used to generate a safe path in proximity to the frontier's start cell. As the robot approaches the frontier, the lidar data is automatically fed through the mapping algorithm to update the map and close the specific frontier under evaluation. When no frontiers remain, the exploration algorithm will automatically return the robot to its start location.

D. Forklift Operation

1) Forklift Design: Two forklift designs were developed. The first, shown in Figure 3 employs a rack and pinion mechanism driven by two Dynamixel XL320 servo motors. The use of primarily laser-cut components (only the motor holders were 3D printed) resulted in a rapidly manufacturable design; the reduced weight of the components enabled fast pallet picking and stacking, even at a distance.

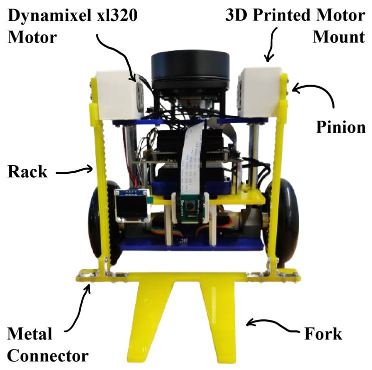

Fig. 3: Rack and Pinion Based Forklift Design

The second design, shown in Figure 4, consists of a compact linear gantry operated by a single Dynamixel XL320 servo motor. This relied on a 2020 aluminum extrusion as a guide for v-wheels moved by a GT2 belt. This design, though heavier, is sturdier and offered a superior control resolution, whilst also improving stability throughout motion thanks to the presence of a counter-castor wheel.

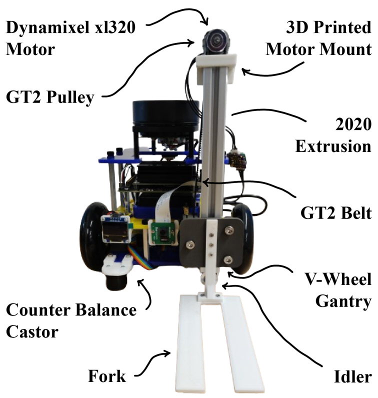

Fig. 4: V-Slot Gantry Based Forklift Design

2) Pallet Detection and Retrieval: Pallets, as shown in Figure 5, feature nested April tags on all sides, with odd ID tags located on the sides where the forklift openings are. The outer tags have an ID that is 10 times the smaller nested tags. These are used to establish a reliability zone: a zone where both small and large April tags are detected which ensures reliable and accurate depth estimation. This zone is used to prevent the robot from colliding with pallets.

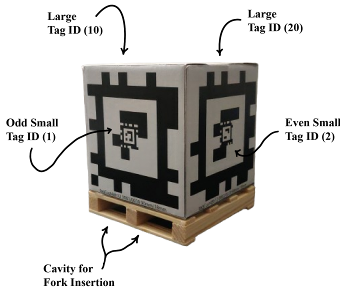

Fig. 5: Pallet Configuration

To locate and retrieve the pallets, two new LCM channels are introduced for the camera and forklift. In fully autonomous mode, the robot captures an image every 0.1 seconds to detect the presence of a pallet in the environment. Upon sensing the pallet, the reliability zone is assessed, and PnP extrinsics are calculated to determine the homogeneous transformation used to guide the robot to a retrieval position.

This is specified via at least two waypoints to locate the robot within 0.2m in front of the pallet insertion region. Once aligned, the bot moves forward until the large April tag detection is lost, triggering the forklift subroutine. This features hard-coded motor settings to ensure the forks are elevated or lowered as appropriate.

III. Results

A. Augmented Odometry and Motion Control

1) Augmented Odometry: To evaluate the performance of the augmented odometry, the robot was commanded to drive a 1m square at increasing translational and rotational speeds: 0.1m/s and $\pi/2$ rad/s; 0.5m/s and π rad/s; 1m/s and 2π rad/s. The average position and orientation errors from the IMU-augmented odometry are compared to the errors from the encoder-based odometry. Results are reported in Figure 6.

2) Motor Calibration and Control: Prior to evaluating the motor controller's performance, a mapping needs to be established between the PWM signals it would originate, and the wheel velocities produced. This is achieved through calibration whilst assuming an affine

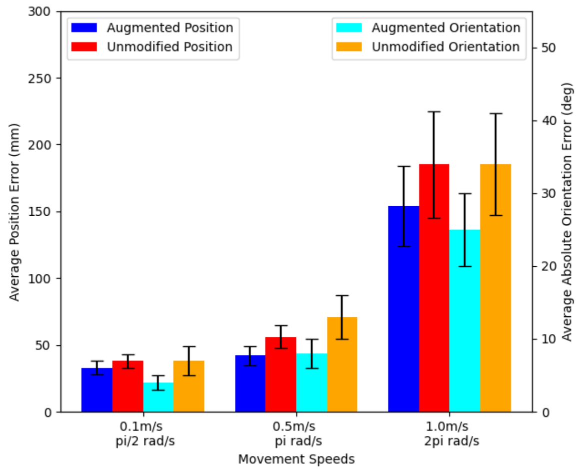

Fig. 6: Average position and orientation errors across different speeds for the augmented and unmodified odometry. Error bars represent the standard deviation.

relationship for this mapping in both the positive and negative domains, although the fits in the two may differ. Table II gathers the fundamental calibration data.

Parameters	Mean	Std Dev
Positive Slope 1	6.7e-2	3.4e-3
Positive Slope 2	7.2e-2	3.4e-3
Positive Intercept 1	6.6e-2	4.1e-3
Positive Intercept 2	5.6e-2	6.5e-3
Negative Slope 1	7.2e-2	3.4e-3
Negative Slope 2	6.5e-2	3.5e-3
Negative Intercept 1	-5.8e-2	4.2e-3
Negative Intercept 2	-7.0e-2	5.0e-3

TABLE II: Motor Calibration Data

Considering the motor controller, this is evaluated by commanding step inputs for both linear velocity v and angular velocity ω. The responses are illustrated in Figures 7 and 8. Time-wise, the controller successfully settles at the specified commands within 1% of the commanded value in 0.5 seconds.

3) Trajectory Following: To evaluate the ability of the robot to track a trajectory accurately, it was commanded to drive a 1m square for 4 times both clockwise and counter-clockwise at the speeds reported in Section III.A.1. The robot's location through the process is compared against the ground truth and visualized in Figure 9. The commanded velocities associated with the trajectory are presented in Figure 10.

B. Simultaneous Localization and Mapping (SLAM)

1) Mapping: To evaluate the performance of the mapping algorithm, the robot was fed simulated lidar data to generate a map of two separate application-relevant environments. To prevent faults in the localization algorithm

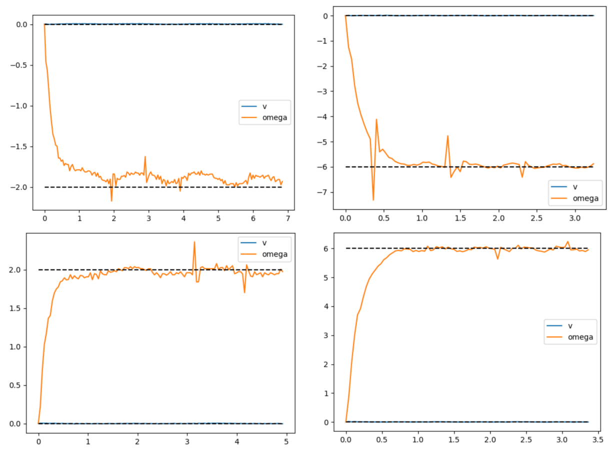

Fig. 7: Performances for ω. **Rows**: negative/positive inputs; **Columns**: low/high magnitude.

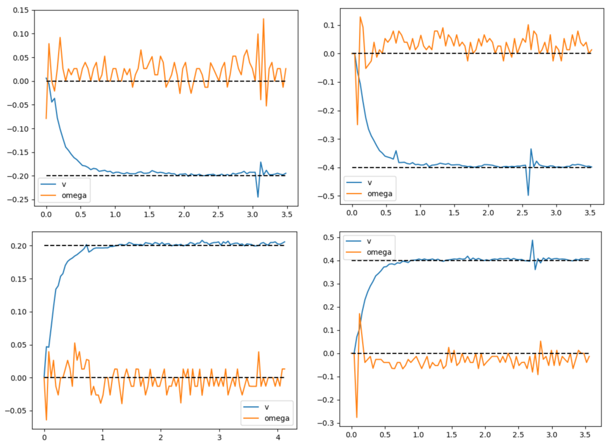

Fig. 8: Performances for v. **Rows**: negative/positive inputs; **Columns**: low/high magnitude.

from altering the results, the ground-truth data was used during this experiment. Given the evident difficulties in comparing the produced maps against their ground truth counterparts quantitatively, these are instead compared qualitatively in Figure 11.

2) Monte Carlo Localization: Two aspects were considered when evaluating the particle filter's performance.

Firstly, the filter's update rate had to be evaluated to ensure that it was sufficiently high to meet the application requirements of 10Hz. To test this, the filter was run with 100, 300, 500, and 1000 particles in two different simulated environments of increasing complexity. To prevent faults in the mapping algorithm from altering the results, the ground truth map was used in this experiment. Performance is reported in Table III.

Averaging the results for each test across environments and plotting frequency against particle number, the relation between the two was found to follow a power curve.

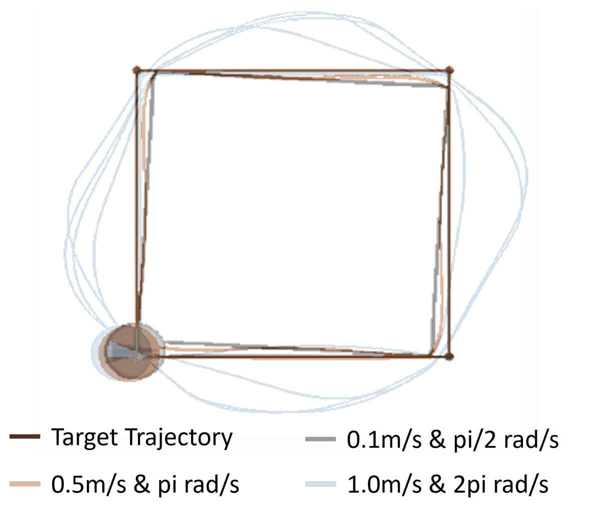

Fig. 9: Augmented-Odometry pose estimate over four loops at varying velocities.

TABLE III: Particle filter update rate for environments of increasing complexity and particle number (prts).

Map	100 prts	300 prts	500 prts	1000 prts
Simple	∼ 830 Hz	∼ 280 Hz	∼ 170 Hz	∼ 90 Hz
Complex	∼ 770 Hz	∼ 270 Hz	∼ 170 Hz	∼ 90 Hz

Based on it, the system can be estimated to support up to 10000 particles when running at the required 10Hz for the considered application.

Secondly, the particle's behavior throughout the motion had to be monitored to ensure that the uncertainty parameters were effectively tuned. To achieve this, the robot was moved through the same two simulated environments (once again using ground truth maps), whilst measuring 300 particles' spread (i.e. the maximum distance between any two particles). Results are reported in Table IV, with Figure 12 visualizing data for one of the complex environments.

TABLE IV: 300 Particles' Spread Statistics

Map	Mean	Max	Min	St. Dev.
Simple	17.7 mm	26.4 mm	0.0 mm	3.4 mm
Complex	19.2 mm	32.5 mm	0.0 mm	3.9 mm

3) Combined Implementation: The overall performance of the SLAM system was evaluated by running the algorithm with 300 particles on the same two simulated environments, albeit without providing ground truth maps or poses. Statistics for the error between the estimated pose and the ground-truth pose are reported in Table V. A visual comparison of the two, and of the produced maps against the ground truth map is produced in Figure 13 for the most complex map considered.

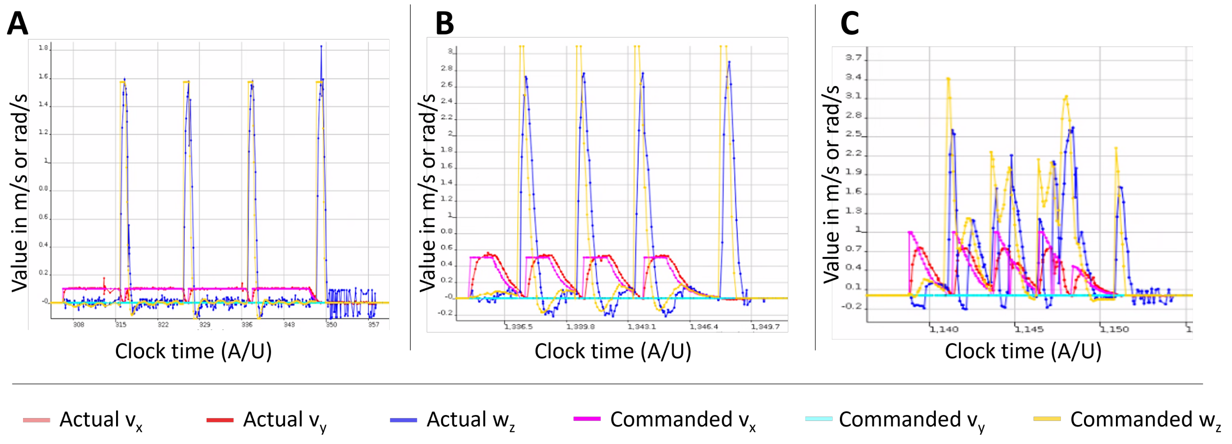

Fig. 10: Commanded and actual velocities for: **A** 0.1m/s and $\pi/2$ rad/s; **B** 0.5m/s and π rad/s; **C** 1m/s and 2π rad/s.

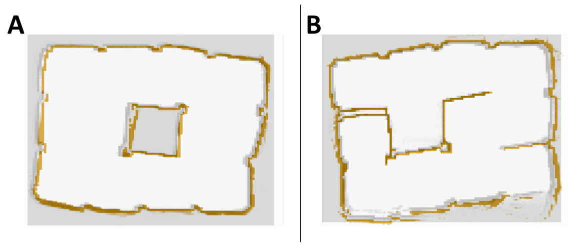

Fig. 11: Comparison of the algorithm-generated map with the ground truth map (yellow overlay) for: **A** low-; **B** high-complexity environments.

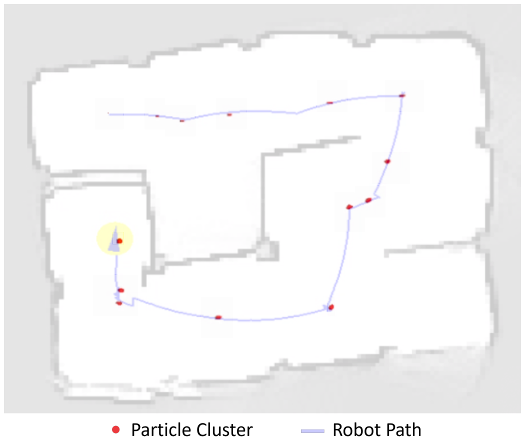

Fig. 12: Particle spread evolution for every 5s of operation. The semi-transparent path and map are overlaid.

C. Planning and Exploration

1) Path Planning: Several evaluations were performed to assess the implemented path planner's performance. Firstly, to address the safety of the path generated, the algorithm was run in four simulated environments of increasing complexity, with and without the obstacle proximity term of Equation 3. The minimum distance between the generated trajectory and the map's obstacles is reported in Table VI.

TABLE V: Error between the SLAM-derived and ground truth pose. Values for position (rows 1 and 2) are separated from those for orientation (rows 3 and 4).

Map	RMS	Max	Min	St. Dev.
Simple	94.3 mm	186.8 mm	0.0 mm	42.5 mm
Complex	128.4 mm	251.9 mm	0.0 mm	53.3 mm
Simple	5.7 deg	12.8 deg	0.0 deg	2.7 deg
Complex	6.3 deg	10.7 deg	0.0 deg	2.5 deg

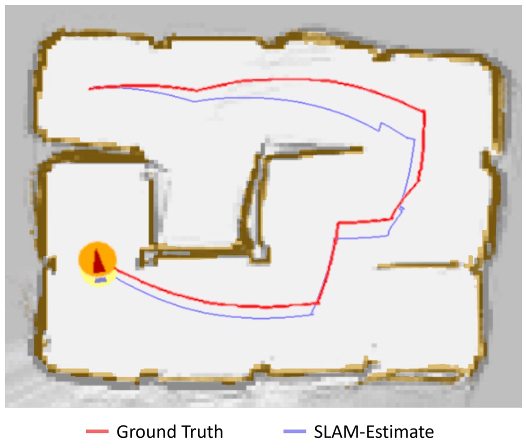

Fig. 13: Comparison of SLAM-estimated map and trajectory against the ground truth (yellow-red overlay).

Secondly, the efficiency of the algorithm implementation is rigorously evaluated through a series of planning tests in the same four environments considered above, whilst tracking the associated computational efficiency. Statistics for this evaluation are reported in Table VII.

TABLE VI: Minimum distance between the generated path and the map's obstacles.

Test	w/o Penalization	w/ Penalization
Convex	5.2cm	7.3cm
Maze	4.4cm	6.9cm
Narrow constr.	4.8cm	7.7cm
Wide constr.	6.8cm	12.5cm

TABLE VII: A* performances in the different test environments. All values in μs.

Test	Min	Mean	Max	Median	Std dev
Convex	46	72.5	99	70.5	26.5
Maze	726	27959	68626	8818	26431
Narrow constr.	3004	3621	4146	3713	470
Wide constr.	3205	3510	3727	3727	221

Finally, the viability of the generated trajectory was evaluated by running the A* algorithm in a real-life environment on the actual robot. A comparison between the planned trajectory and the executed one is reported in Figure 14.

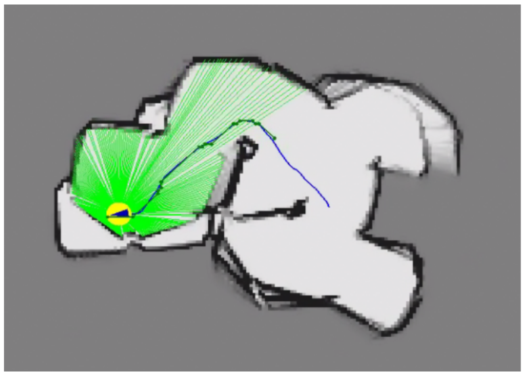

Fig. 14: Comparison of the A*-generated trajectory (green) against the robot's performed one (blue).

2) Exploration: Since the exploration algorithm heavily relies on the algorithms described above, instead of analyzing the quality of the exploration (dependent on the mapping algorithm and captured in Section III.B.1), this section will exclusively report its success rate. Specifically, the system was run ten times in the two simulated environments considered for the SLAM evaluation. Exploration was successful 70% of the time in the low-complexity environment and 40% in the high-complexity environment. The main failure modes were the system's inability to return home within the designated tolerance, and its inability to complete the exploration due to faults in the state machine.

D. Forklift Operation

Several aspects were considered to evaluate the performance of the developed forklifts and the associated code. Firstly, the algorithm's ability to detect the tags was evaluated. This was achieved by recording when the small and large tags would be detected. Throughout this assessment, the robot was driven in straight lines from a distance of 5m from the block, with the lines diverging from the block's normal at 0deg, 15deg, 45deg, and 60deg. Reliable detection ranges for the small and large April tags were found to be 0.1 m to 0.6 m and 0.2 m to 5 m, respectively. The reliability zone was thus set to be between 0.2 m and 0.6m, with the lower limit set to account for the fork's length.

Secondly, the system's ability to autonomously pick up blocks was evaluated. This was achieved by placing the robot at a 0.5m distance from the pallet along the same directions reported above and testing the block-retrieval subroutine. Whilst the robot moved to the correct retrieval location without fault, it was noticed that both designs did not elevate the block sufficiently to enable stacking, due to a lack of stiffness in the forks which caused them to sag under the pallets' weight. This was found to occur in 5 of the attempted 7 trials performed for each design.

Considering the operation of the specific designs, the rack and pinion one was found to be fast but jittery, sometimes shaking the pallet slightly forward in an unsecured position. Additionally, the two motors could sometimes go out of sync, causing the forks to move at an angle. As for the gantry-based design, this was found to be significantly smoother to operate, without facing any of the issues of its simpler counterpart.

E. Competition Performance

The competition provided an opportunity to evaluate the performance of the system holistically. Table VIII reports performance for each task, with the "Success %" metric being based out of 10 attempts.

TABLE VIII: Competition Performance Summary

Task	Autonomy	Success %	Main Limiting Factor
1	Full	70%	Home-Return Issues
2	None	70%	Forklift Design
3	Full	20%	Home-Return Issues
4	None	20%	Forklift Design

IV. Discussion

A. Odometry and Motion Control

Considering firstly the IMU-augmented odometry, this appears to tangibly improve on the simple encoder-based odometry only at high speeds, suggesting that the implemented modifications could be further improved upon, with potential benefits for the entire architecture.

Addressing secondly the variation in the calibration data, data in Table II indicates that the standard deviation falls within the range of 5% to 10% of the mean value. This variation could stem from multiple factors:

- Possible inherent noise in motor velocity measurements introduced by the encoders;
- Possible variations in the ground surface might cause wheel slippage during calibration;
- Variations in how undercarriage wires rest on the ground, altering friction.

Irrespective of the above, the data variation is small enough to be considered acceptable.

The performance of the wheel controllers also appears adequate, guaranteeing very low settling time, overshoot, and steady-state errors.

As for the performance of the trajectory tracking algorithm, data in Figures 9 and 10 suggests that this performs as desired only at low velocities, with performance tangibly deteriorating as the motion speed increases. Given the results gathered for the augmented odometry, it is likely that this behavior results from the sub-optimal performance of this estimation.

B. Simultaneous Localization and Mapping (SLAM)

The implemented SLAM system appears highly capable of generating environment maps, as visualized in Figures 11 and 13. The latter however also highlights non-negligible errors in the localization-derived pose estimate, confirmed by the data reported in Table V.

The most likely explanation for the above can be inferred from Figure 12, where the particles remain too clustered together and do not adequately capture motion- and measurement-derived uncertainty, preventing better estimates. Given the extremely streamlined measurement model used, increasing the number of particles could be a viable option to alleviate this issue without sacrificing performance, as inferred from Table III. More effectively perhaps, a more sophisticated measurement model could yield better estimates. In both cases, the action model could also be re-tuned to enable greater particle spread.

Irrespective of this limitation, the system's performance appears acceptable, with the maximum error from Table V being at most a third of the smallest dimension of the space the robot is navigating, and the RMS value being only slightly larger than the robot's base radius.

C. Planning and Exploration

The developed planning algorithm is shown to work efficiently whilst generating viable trajectories that the physical robot can follow. It does not however appear to generate sufficiently safe trajectories: given the error associated with the SLAM-derived localization, the generated paths should feature much larger distances from obstacles than those reported in Table VI.

The planning algorithm's performance also appears to be sub-optimal. The second failure mode recorded indicates that the state-machine's implementation needs improvements to account for edge cases that cause the robot to stop exploring. The first instead is a consequence of the shortcomings of the A* algorithm.

D. Forklift

The main fault in both proposed designs lies in the sagging of the forks due to the use of an insufficiently stiff material. This could be overcome by either increasing the height of the racks or guide extrusions or employing a stiffer material.

Considering the operation of the forklift, meaningful difficulties were encountered in integrating their logic within the system's autonomous control algorithm.

E. Competition Performance

The system was capable of tackling all competition tasks at least once, albeit with some limitations. Specifically, in navigation-focused tasks 1 and 3, the system was not capable of reliably returning to its start pose within the designated tolerance. Considering the data reported in Section III.A and III.B, this is likely caused by the combined effects of insufficiently precise SLAM and augmented odometry, rather than by issues in the trajectory controller or path planner. In the case of task 3, these problems were aggravated by issues within the exploration state machine, which would frequently fail.

In the forklift-focused tasks 2 and 4, the robot encountered issues in recovering blocks autonomously. This is mostly due to improper integration of the autonomy logic within the overall system, something that could only be overcome by performing the tasks in teleoperated mode. A future solution could be the development of a dedicated state machine to manage the transition between different actions more reliably. Additionally, performance was affected by the design issues documented in Section III.D, which prevented reliable block stacking.

V. Conclusion

The system overall appears well versed in tackling all tasks required, albeit highly unreliable. Iimprovements are required in the IMU-augmented odometry, localization, and forklift design and autonomy integration to tangibly augment the systems' performance.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2006. [Online]. Available: http://www.probabilistic-robotics.org/

[2] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2005. [Online]. Available: https://books.google.com/books?id=wGapQAAACAAJ