Adaptive Point-Tracking Control of Cooperative Multi-Robots Box Pushing

2291911 2563570 2596788 2584999

***Abstract*—This paper addresses a cooperative multi-robot box-pushing problem in a straight-line trajectory by proposing an Adaptive Point-Tracking Control (APTC). The APTC algorithm is designed to ensure straight-line movement and correct the box's orientation deviations. By combining Point-Tracking Control (PTC) with directional bump detection feedback in a PID controller, the algorithm adjusts the PTC gain to regulate the linear velocity of the pushing robots. Experiments with Pololu 3pi+ robots demonstrate that the APTC algorithm effectively minimizes box deviation and enhances trajectory precision and consistency. The results indicate that the APTC algorithm outperforms traditional PTC and PID controllers in maintaining low Yaw deviations and is robust up to initial box orientations of 400 ms delays, proving its applicability in various controlled environments. We open-source our implementation on GitHub.**[1]

I. Introduction

Mobile robots have become essential tools for object manipulation and transportation. Among the various techniques available, pushing is notable for its simplicity in moving objects to designated locations [1]. Multi-robot systems provide significant advantages over single-robot systems for box-pushing tasks, as they can distribute workload and handle objects that would be too large or heavy for individual robots [2]. However, this approach introduces additional challenges in coordination and control. To address these challenges while maintaining simplicity, this study focuses on the cooperative control of multi-robots to translate a box in a straight-line trajectory, specifically targeting scenarios where the box exceeds the manipulation capabilities of a single robot and aiming to minimize deviations from the intended trajectory.

The box-pushing problem involves altering the position and orientation of a stationary object on a 2D surface by applying point contact forces [1]. To achieve straight-line box translation, this problem presents two primary challenges: (1) ensuring that the robots maintain a globally straight trajectory of themselves during the pushing process while being able to correct the box orientation, and (2) ensuring that mobile robots don't lose control or lose contact from the box throughout the motion.

To achieve a straight trajectory for mobile robots, the resist rotation behaviour could be used by activating an inertial measurement unit (IMU) to calculate the current heading of the robots and try to fix its heading to the initial heading [3]. However, the sensor needs calibration and a complex sensor fusion algorithm to compensate for noise and accumulative drift. To reduce the use of external sensors and the complexity, we adopt a Point-Tracking Control (PTC) algorithm that relies on the kinematics and odometry of differential wheeled mobile robots to track a point in the 2D plane [4]. By generating a virtual reference point in the same direction as the robot's heading, PTC will determine the necessary linear and rotational velocities for straight-line motion. Robot position and orientation are estimated using wheel encoders that measure rotational angles and directions, commonly called odometry.

As the PTC algorithm focuses on tracking a point without perceiving the box orientation, the box may lose contact with the robots. We utilize front-mounted bumpers to measure the contact force direction. This measurement serves as feedback in a PID control system, where the set-point of zero represents a perfectly balanced contact force. The PID controller outputs the PTC parameter to minimize directional error, helping maintain balanced contact while following a straight-line trajectory.

In this paper, we propose a cooperative control algorithm called an Adaptive Point-Tracking Control (APTC) algorithm that combines the PTC algorithm with bumpers-based perception to solve the multi-robots box-pushing problem in a straight trajectory while minimizing the deviation of the box. Our experimental results compare APTC algorithm performance against the PID Speed Controller and PTC algorithm.

A. Hypothesis Statement

Building upon the PTC algorithm's stability in point tracking [4] if two robots are initiated simultaneously and maintain straight-line trajectories individually, the pushing box may also follow a straight path due to the balanced forces, leading to the first hypothesis:

> ***H1****: Implementing a Point-Tracking Control (PTC) algorithm in mobile robots ensures they follow a straight path, making the pushed box move in a straight line too. This is expected to minimize the orientation error of the box and improve precision in controlled environments.*

As APTC algorithm incorporates bump sensor feedback, it may be robust in scenarios where initial box orientations are different. This leads to our second hypothesis:

> ***H2****: The initial box orientation might have a greater impact on baseline control compared to Adaptive Point Tracking Control (APTC). Utilizing bump sensor readings, mobile robots under APTC are expected to correct the box's orientation error and ensure stability during the pushing operation while remaining precise and accurate.*

The rest of this report proceeds as follows: Section II describes the control algorithms we have adopted and further improved. Section III describes the setup procedure of the experiment method and metrics for measuring the performance of each scenario. Section IV shows the

[1] https://github.com/nongmelt/box_pushing.git

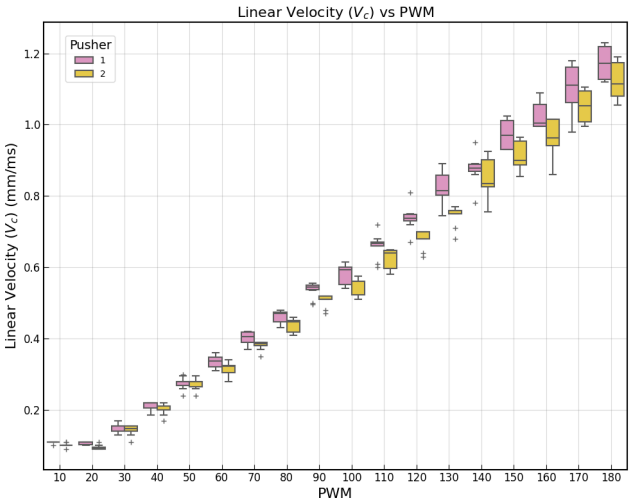


Fig. 1: Relationship between linear velocity and PWM of pushers

findings and quantitative results. Section V discusses the findings along with the presented hypotheses and suggests future works for improvement.

II. Implementation

In this section, we describe all implementations of our baseline, which is PI for speed controller, PTC algorithm, and APTC algorithm.

As shown in figure 4, we are using 3 Pololu Pi+ robots, two for pushing the box called *pushers*, and another robot will be kept in the center of the box, represented as the box orientation called *observer*.

A. Speed Controller

Since a constant PWM signal does not guarantee a constant speed and each robot has a different PWM-to-speed characteristic as shown in figure 1, we implement a Speed Controller using a PID controller for the right and left wheels of each robot. PID controllers adjust PWM value to ensure the robots maintain a constant speed. This makes a baseline for our experiment and a building block for a straight-line control strategy.

B. Point Tracking Control Algorithm

This section derives the stability of the control algorithm using *Lyapunov Stability Theory*, and the essential equations to determine linear velocity V_{c_p} and rotational velocity ω_{c_p} for differential mobile robots to track a point. The kinematics of a differential-wheeled robot are given by:

$$\dot{x}_p = V_{c_p} cos(\theta_p) \tag{1}$$

$$\dot{y}_p = V_{c_p} sin(\theta_p) \tag{2}$$

$$\dot{\theta}_p = \omega_{c_p} \tag{3}$$

where x_p is the position to the destination of the *pusher*, y_p is the lateral position of the *pusher*, and θ_p is the heading angle of the *pusher*. Given V_{c_p} and ω_{c_p}, we can calculate the left velocity V_{l_p} and right velocity V_{r_p} as following:

$$V_{c_p} = \frac{V_{r_p} + V_{l_p}}{2} \tag{4}$$

$$\omega_{c_p} = \frac{V_{r_p} - V_{l_p}}{D_p} \tag{5}$$

where V_{r_p}, V_{l_p} are right wheel speed and left wheel speed respectively, and D_p is the diameter of the *pusher*.

To achieve the PTC algorithm, let's assume that the tracking target is at origin $(0, 0)$ and *pusher* is located in any arbitrary position (x_p, y_p) as shown in the figure 2 where r_p is the Euclidean distance from the *pusher* to the tracking point or *distance target*, ϕ_p is the relative angle between the current heading and the tracking point or *approaching angle*.

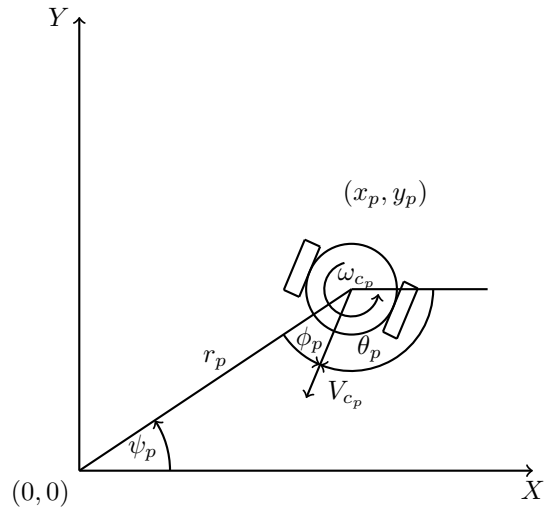


Fig. 2: Representation of state variables for PTC [4]

From Figure 2, we obtain $r_p = \sqrt{x_p^2 + y_p^2}$ and $\phi_p = \pi_p - \theta_p - \psi_p$. We want to design a control law that maintains the value of *the distance target* while converging *approaching angle* to zero [4]. We consider the equations given above and the Lyapunov function as derived in [5], and obtain obtain control law by using positive K_1 for V_{c_p} and K_2 for ω_{c_p} to track the desired point as the following equation:

$$V_{c_p} = K_1 r cos(\phi_p) \tag{6}$$

$$\omega_{c_p} = -K_1 sin(\phi_p) cos(\phi_p) - K_2 \phi_p \tag{7}$$

To adopt the PTC control algorithm, we assume that we have a virtual reference point in the direction along the x-axis. While *pushers* manipulate the box, the reference point remains constant in a global coordinate frame. If there is a heading error during moving, the PTC control algorithm will determine the values of V_{c_p} and ω_{c_p} using equations in (6, 7) to adjust the left V_{r_p} and right wheel speed V_{l_p} accordingly to align with the given reference point using equations in (4, 5). This enables *pushers* to move with decreasing linear velocity while reducing the heading error and going straight.

C. Adaptive Point Tracking Control Algorithm

We know that PTC could not perceive the box orientation while pushing; however, with two frontal bumpers on *pushers*, we can design a control law to balance the reading from two bumpers by calculating the error of two bumpers values, called *bumper balance*. We implement the P controller with *bumper balance* as an input to the controller and results as a K_1 gain for our PTC control algorithm. With this method, *pushers* can take into account *bumper balance* and adjust their linear velocity (V_{c_p}) accordingly while maintaining the straight path.

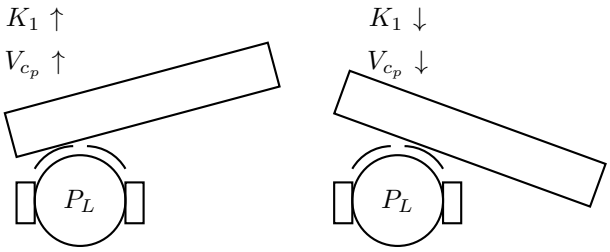


Fig. 3: *Left pusher* scenario when the left bumper is pressed more than the right one (left), and when the right bumper is pressed more than the left one (right).

As shown in Figure 3, when considering the left *pusher*, if the left bumper value is higher than the right bumper reading, it indicates that the box on the left side is lagging behind the right side. Consequently, the left pusher should increase its linear velocity. However, this adjustment must not disrupt the PTC control algorithm. One method to perform linear velocity adjustment is to modify the linear velocity gain of the PTC control algorithm. Since K_1 is proportional to linear velocity, increasing K_1 will result in an overall increase in linear velocity. Conversely, if the right bumper reading of the left pusher is higher than the left bumper, the overall linear velocity should be decreased, which correspondingly reduces the K_1 gain. For the right pusher, the gain adjustment process follows the same principle, but with an inverse logic.

III. Experiment Methodology

A. Overview of Method

In this section, we detail the experimental setup for evaluating our proposed hypotheses. As shown in figure 4, we established a linear track with a start line at $x = 0$ and a finish line at $x = 1000$ mm. The box is positioned such that its lower border edge aligns with the start line, with the middle of this edge serving as the origin point of our setup.

We deploy three Pololu Pi+ robots, two robots as *pushers* are placed symmetrically, 240 mm apart, and another robot as *observer* will be kept in the center of the box, and enables a gyroscope along with high-pass filter to estimate yaw angle θ_o, represented as the box deviation. We use a groove between two pieces of 3D printed part to verify *pushers* placement.

Before each trail set, we perform surface cleaning to ensure constant friction, and test the Speed Controller, PTC algorithm, and APTC algorithm in sequence.

First, the *observer* is activated, allowing its gyroscope sensor to initialize and complete its calibration routine.

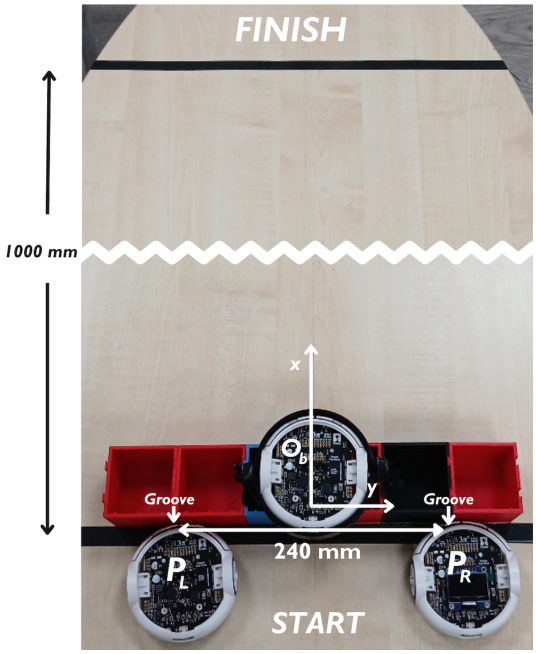


Fig. 4: The Experiment setup used for all the different control systems

When a beep sounds, the operator simultaneously activates the *pushers*. Since we assume the ambient light environment remains relatively stable, we ignore bumper calibration routine during start. We perform a single bumper calibration where we do the calibration 5 times, storing both the mean minimum values and mean ranges for each robot's left and right bumpers in memory.

Upon activation, the *pushers* become motorized and continue until their x_p positions reach 1000 mm. At the trial's conclusion, another beep from the *observer* indicates completion, prompting data collection. We collected 65 data points in total for each *pusher*. We collected this amount of data to maximize memory usage while ensuring predictable robot behaviour (staying below 90% of RAM utilization). Additionally, we record 250 data points measuring the *observer*'s estimated yaw (θ_o) at 50 Hz over 5 seconds, a total duration for each trial.

- **Speed Controller**: For both hypotheses, two pushers use the PID speed controller by setting the desired linear velocity to 0.35 m/s. Two left and right wheels are subjected to different PID controllers with the same K_p, and K_i values.
- **PTC algorithm**: Two pushers will be given virtual reference tracking points ahead of them in the same direction as their initial headings. We tune K_1, and K_2 gains by experiments with a single robot monitor δy error, and adjust the gain until the error is less than ± 1 mm. We notice that while *pushers* move toward the finish line, the linear velocity decreases until the distance some motor doesn't function. Therefore, we calculate K_1 gain with an assumption that the pushers will go straight, and want *pushers* to reach the finish line at a minimum linear velocity of 0.14 m/s. Hence, we obtain the set *distance target* of 1220 mm. For K_2 gain, we ran the experiments and tuned accordingly to minimize the deviation of the box until we got the

proper value before the control law became unstable.

- **APTC algorithm**: With the values of K_1 and K_2 from PTC, we add an additional one P controller to calculate the *bumper balance* error, δB, and this results in an adaptive K_1 gain. Each left pusher and right pusher has a different direction of δB. To tune the P controller, we limit the linear velocity of the pushers to [0.14, 0.85] m/s. As the *bumper balance* value lies between [0.0, 1.0], we can calculate the bounding limit of K_p gain. By conducting experiments, we use the bisection method to adjust P gain to get the best result.

1) Hypothesis 1: We compare three different control systems given the same setup and initial linear velocity, which is 0.35 m/s. With this linear velocity, the single pusher can not manipulate the pushing box. The experiment is conducted over 10 trials for each control strategy.

2) Hypothesis 2: We compare two control strategies, the Speed Controller and APTC, as we expect that the PTC and APTC will share the same trend. We use the delay function on the left *pusher* starting by 100 ms to 500 ms with 100 ms intervals to imitate the different initial box orientations. This range was chosen based on average human reaction time (approximately 200-250 ms) to cover both faster and slower response scenarios.

B. Discussion of Variables

- **Independent Variables**: For *H1*, We compare three different control systems of two pushers—(1) PID Speed Controller, (2) PTC, and (3) APTC—and measure the performance of each control system. For *H2*, We compare (1) PID Speed Controller and (2) APTC.
- **Dependent Variables**: For collecting data points, we observe the odometry of each pusher (x_p, y_p, θ_p), linear velocities (V_{l_p}, V_{r_p}), and bumper readings (B_{l_p}, B_{r_p}). We observe the deviation of the box, which is represented by the estimated yaw of the observer (θ_o)
- **Controlled Variables**: The 3D-printed PLA box measuring 360 mm x 60 mm and weighing 320 g, including the 146 g observer unit with batteries. All experiments use the same surface with an assumed uniform friction coefficient. To ensure reliable robot motor control and prevent slipping, we constrain the speed between 0.14 m/s and 0.85 m/s. The speed controller's PI gains remain constant across all robots and control strategies. The left and right pusher configurations are consistent throughout all experiments.

C. Discussion of Metric(s)

1) Hypothesis 1:

- **Trajectory Deviation**: We measure the deviation between each *pusher's* observed position $y_p(x)$ and the target position $y_{target} = 0$ mm, defined as the *translation error*. We quantify this deviation using the Root Mean Square Error (RMSE), which indicates how well the PTC algorithm helps *pushers* maintain their intended straight-line trajectories compared to our baseline. We also calculate the RMSE between the observer's yaw angle estimation $\theta_o(x)$ and the desired value $\theta_o = 0°$ for each control strategy to justify the effect of *pushers*'s trajectory on the box's trajectory.
- **Controller Yaw Stability**: To assess the control system's ability to maintain the box's orientation, we calculate the Mean Square Error (MSE) $\theta_o(x)$ for each control strategy. This metric evaluates how effectively different control strategies correct box orientation errors.
- **Precision and Accuracy of Controller**: We measure the final box position $(x_{box,dest}, y_{box,dest})$ to evaluate the performance of each control strategy using two metrics. The precision is quantified by the Standard Deviation (SD) of final box positions. The accuracy is measured by calculating Euclidean distances between the actual final box positions and the target position $(x_{box,target}, y_{box,target}) = (1000, 0)$ mm, defined as the *position error from target*.

2) Hypothesis 2:

- **APTC Performance Across Initial Box Orientations**: We evaluate the APTC method's robustness by calculating two metrics for each initial delay time: the *position error* between final and target positions, and the Mean Squared Error (MSE) of $\theta_o(x)$. These measurements quantify how effectively the APTC method performs under different initial conditions.

IV. Results

A. Hypothesis 1

1) Trajectory Deviation: As shown in Figure 5a, the median value of the *translation error* under Speed Controller is approximately 4 times greater for the *right pusher* and about 10.5 times greater for the *left pusher*, compared to other control strategies, around 1-2 mm for PTC and APTC algorithms. This significant deviation highlights the challenges PID Speed Controller faces in maintaining a straight-line trajectory.

Figure 5b shows that the PTC control strategy exhibits a higher median value (4.4°) for box trajectory deviation than the PID control strategy (3.6°). While PTC makes *pushers* move straight, the box does not move straight as expected.

2) Controller Yaw Stability: The PID Speed Controller control strategy has the advantage of yaw error correction because of PID gains, which let it maintain speed and correct deviations even when the pressure from the box causes it to slow down. Although PID Speed Controller-control exhibits yaw deviations in the middle of the path, it manages to correct these deviations towards the end position to some extent, as shown in figure 5c.

From the figure 5b, the ranges highlight that APTC demonstrates the lowest deviation, followed by PID and then PTC. This indicates that the APTC maintains the smallest median deviation in RMSE of 2.288°, while the PID controller has a median deviation of 3.645°, and the PTC algorithm exhibits larger median deviations of 4.393°.

Additionally, the standard deviation values for different control strategies indicate that PTC maintains more consistent distances from the mean, whereas PID exhibits significantly higher variability, which further supports the observation that PID control faces challenges in maintaining a stable trajectory.

PID Speed Controller can minimize the Yaw error of the box trajectory near the end, but it has trouble staying stable in the middle of the journey. Even though PTC isn't as

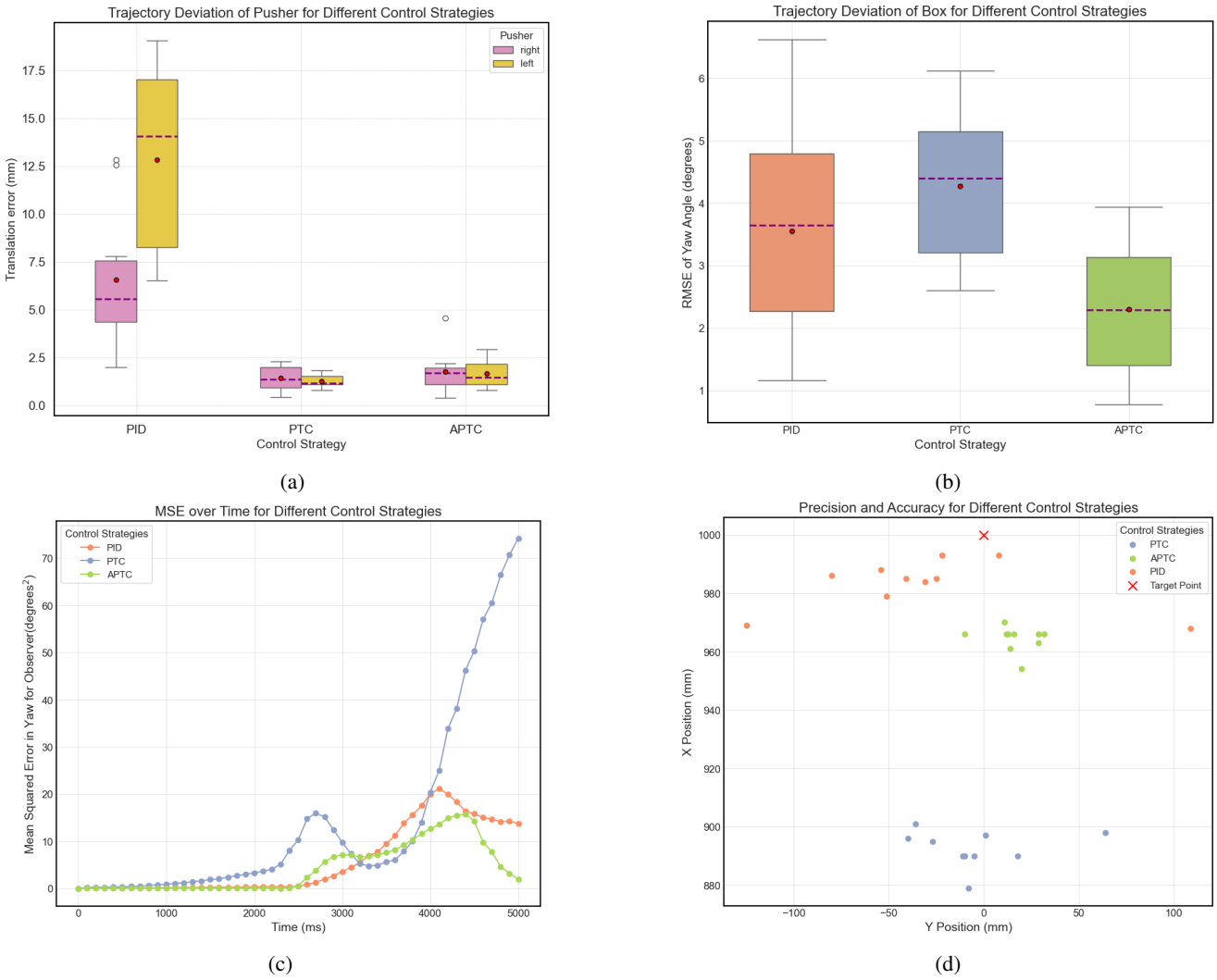


Fig. 5: Metrics of Hypothesis 1 for three different control strategies: (a) translation error RMSE of *pushers* and (b) yaw error RMSE of *observer* (c) MSE of yaw of *observer* (d) Euclidean distances of final box positions

consistent as APTC, it's better than PID Speed Controller when it comes to steadiness. This analysis demonstrates that even with a steady trajectory, PTC does not correct yaw error as much as PID.

3) Precision and Accuracy of Controller: The figure in 5d shows that while PID Speed Controller exhibits high accuracy in reaching near the target it suffers from lower precision, as evidenced by its relatively high standard deviation of **39.000**. In contrast, the PTC provides high precision, demonstrated by a significantly lower standard deviation of **5.751**. Nevertheless, the accuracy of the PTC could potentially be higher if not for the stopping limitations imposed by the *pushers* as we cannot reach the low PWM, which causes them to halt before fully reaching the target.

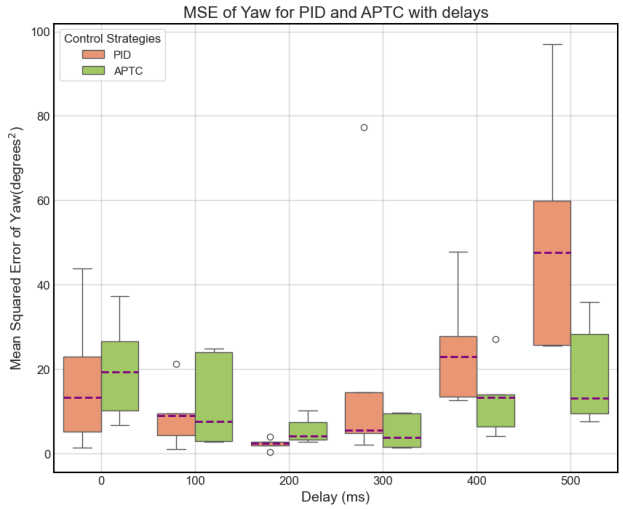
Based on these observations, we conclude that Hypothesis 1 is **falsified**. Contrary to initial expectations, the PTC does not outperform the PID Speed Controller in terms of maintaining a stable trajectory for the box. The feedback error in PID control contributes significantly to its relative effectiveness in managing trajectory deviations over time.

B. Hypothesis 2

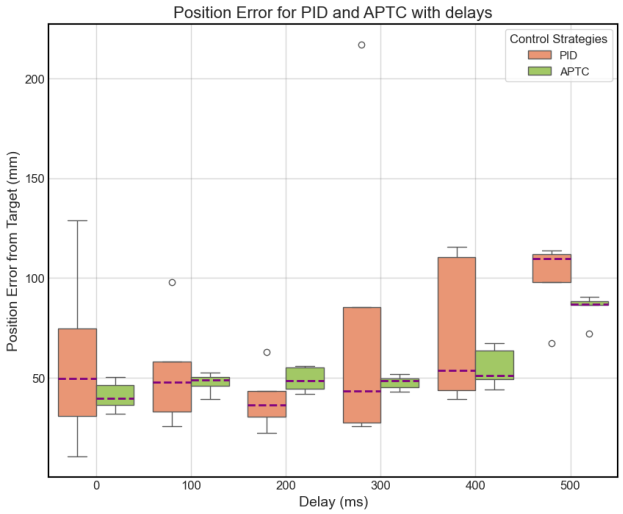
1) MSE of the Observer's Yaw: The figure 6a is analyzed to assess stability and precision. The results indicate that the PID control strategy becomes increasingly unstable with higher delays, in other words, high initial box orientation, as evidenced by larger box plots, reflecting lower precision and accuracy. This instability is caused by the PID controller not detecting the box's orientation as accurately as the bump sensor. With the PID controller, we might assume that the speed drop is due to the collision or losing contact with the box, resulting in an uncoordinated speed increase without proper alignment with the other *pusher*. This lack of coordination between the *pushers* leads to significant trajectory deviation, as the *pushers* are unaware of each other's orientation and fail to adjust their speed accordingly.

In contrast, the APTC measures the directional contact force with the box accurately, providing information about the box's orientation, which helps *pushers* balance their bumper values and match their speed. This coordination significantly reduces trajectory deviation.

As shown in figure 6a, the APTC values remained constant with the median staying around 13 deg^2, while



(a)



(b)

Fig. 6: Metrics of Hypothesis 2 for two different control strategies with varying delayed of: (a) yaw error RMSE of *observer* and (b) translation error RMSE of *pushers*

PID's median jumps to about 47 deg^2 at a 500 ms delay, indicating stable performance for APTC despite delays.

2) Position Error from Target: This stability is further supported in figure 6b, where the PID controller shows greater data dispersion compared to the APTC across all delay conditions. Even with increasing delays, the APTC demonstrates consistent performance with lesser dispersion, highlighting its robustness in managing trajectory and positional accuracy.

From figure 6b we see that the range of values is more for the PID which indicates the instability compared to APTC.

Based on these results, Hypothesis 2 is supported. The APTC controller, with the integration of a bump sensor, proves to be more robust and consistent in performance, especially under delay conditions, compared to the PID Speed Controller.

V. Discussion and Conclusion

This study provides valuable insights into the effectiveness of different control strategies for managing the trajectory and heading of a box-pushing robot. While the PID Speed Controller helps manage trajectory deviations through error adjustments, it exhibits instability. PTC, despite having fewer median trajectory deviations for pushers, fails to effectively reduce box heading errors over time.

Figure 5b illustrates that PTC exhibits a higher median value for box trajectory deviation compared to PID control. Although PTC makes pushers move straight, the box does not follow the expected path due to environmental factors like slip between pushers and the box. This slip creates a tangential force affecting the box's trajectory. Misalignment of pushers with the box further amplifies the error as pushers continue exerting forward force without adjusting for the box's position, resulting in deviation.

The APTC strategy, particularly with bump sensor integration, demonstrates greater robustness and consistency, even under delay conditions. It achieves higher precision and accuracy, although its stopping mechanism requires refinement. Data reveal that APTC outperforms PID in both delayed and non-delayed scenarios, with a lower median delayed position error (86.83mm for APTC vs. 109.65mm for PID at 500ms delay). Although APTC struggles under high delays, it results in less yaw error, potentially due to high bumper gains preventing deviation correction. Adjusting bumper gains can help the box maintain its initial orientation and promptly correct deviations.

Overall, this study highlights the necessity of customised control strategies to optimize performance in box-pushing tasks, with APTC emerging as a more reliable solution in scenarios involving delays and the need for high precision. Future work could explore moving the box along arbitrary trajectories and maintaining a constant speed. PTC focuses on stable low-speed operation, which has limitations as pushers struggle at very low PWM levels. In contrast, APTC uses adaptive gains, reducing position errors effectively but relies on PTC's speed-reduction approach, constraining its ability to fully reach the target. Enhancements like adaptive bumper balance for correcting initial orientation errors and box pushing for non-linear paths can be explored.

The APTC strategy emerges as a more reliable solution for scenarios requiring precision and delay handling. Further optimization of speed control, adaptive gain adjustments, and trajectory tracking could significantly enhance its effectiveness in autonomous object manipulation tasks.

References

[1] A. Verma, B. Jung, and G. Sukhatme, "Robot box-pushing with environment-embedded sensors," in *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515)*, pp. 212–217, 2001.

[2] E. F. Parra-González and J. G. Ramírez-Torres, "Cooperative multi-robot box-pushing in a cluttered environment," in *2008 Electronics, Robotics and Automotive Mechanics Conference (CERMA '08)*, pp. 514–519, 2008.

[3] P. O'Dowd, "Supplementary labsheet 3: Inertial sensors and magnetometer," 2024. Accessed: 28-10-2024.

[4] S. Moon, D. Kwak, and H. J. Kim, "Cooperative control of differential wheeled mobile robots for box pushing problem," in *2012 12th International Conference on Control, Automation and Systems*, pp. 140–144, 2012.

[5] S.-O. Lee, Y.-J. Cho, M. Hwang-Bo, B.-J. You, and S.-R. Oh, "A stable target-tracking control for unicycle mobile robots," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, vol. 3, pp. 1822–1827 vol.3, 2000.