

Regulated Pure Pursuit for Robot Path Tracking

Steve Macenski^{1*}, Shrijit Singh², Francisco Martín³ and Jonatan Ginés³

¹R&D Innovations, Samsung Research America, Clyde Ave, Mountain View, 94043, CA, United States.

²Department of Computer Science, Manipal Institute of Technology, Udipi - Karkala Rd, Eshwar Nagar, Manipal, 576104, Karnataka, India.

³Intelligent Robotics Lab, Rey Juan Carlos University, Camino del Molino, Fuenlabrada, 28943, Madrid, Spain.

*Corresponding author(s). E-mail(s): s.macenski@samsung.com;

Contributing authors: shrijit.singh@learner.manipal.edu; francisco.rico@urjc.es; jonatan.gines@urjc.es;

Abstract

The accelerated deployment of service robots have spawned a number of algorithm variations to better handle real-world conditions. Many local trajectory planning techniques have been deployed on practical robot systems successfully. While most formulations of Dynamic Window Approach and Model Predictive Control can progress along paths and optimize for additional criteria, the use of pure path tracking algorithms is still commonplace. Decades later, Pure Pursuit and its variants continues to be one of the most commonly utilized classes of local trajectory planners. However, few Pure Pursuit variants have been proposed with schema for variable linear velocities - they either assume a constant velocity or fails to address the point at all. This paper presents a variant of Pure Pursuit designed with additional heuristics to regulate linear velocities, built atop the existing Adaptive variant. The *Regulated Pure Pursuit algorithm* makes incremental improvements on state of the art by adjusting linear velocities with particular focus on safety in constrained and partially observable spaces commonly negotiated by deployed robots. We present experiments with the Regulated Pure Pursuit algorithm on industrial-grade service robots. We also provide a high-quality reference implementation that is freely included ROS 2 Nav2 framework at <https://github.com/ros-planning/navigation2> for fast evaluation.

Keywords: Service Robots, Mobile Robots, Motion Planning, Path Planning

1 Introduction

Dynamic Window Approach (DWA) [1], Pure Pursuit (PP) [6], and Model Predictive Control (MPC) [2] are by far the most commonly deployed path trackers. They all have a strong heritage for reliability in a wide range of environmental conditions. DWA and MPC are often, but not

always, formulated as multi-objective trajectory generation problems to maximize criteria such as avoiding dynamic obstacle collisions on top of path tracking. This has made them particularly well suited for many robotics applications where dynamic robot behaviors are rewarded. A great deal of work has been conducted on these allowing

them to be found on many commercially available robots today.

However, there still exist many applications of deployed robot systems where this can be considered a detractor. Surveyed among high-end research and service robot navigation systems, pure path tracking continues to be a common theme in many robots [3, 4]. Among single-objective path trackers, a simple and reliable method continues to be exploited decades after its development: Pure Pursuit.

Pure Pursuit uses simple geometry to find the curvature of a path required to drive a robot towards a given point on the path. The algorithm itself does not place any stability restrictions on the translational velocities during operation, however it also lacks any schema for selecting them. Near-universally, implementations use a fixed speed. Many variations of Pure Pursuit exist, however most address the more obvious area of the selection of lookahead points, which largely aids in stabilizing convergence behaviors towards the path at a wider range of velocities. The Pure Pursuit algorithm was not developed with service and industrial robots in mind, which have additional safety requirements making it further unrealistic to move at a fixed velocity.

This work proposes an incremental improvement on the Pure Pursuit path tracking algorithm by describing a reference method of adjusting translational and rotational velocities to improve safety and operability in a broad range of common deployed robot applications. We improve the Adaptive Pure Pursuit (APP) algorithm [7] by regulating velocities via penalizing sharp changes in path curvature and proximity to obstacles - two of the most common events requiring conscientious navigation behaviors. The *Regulated Pure Pursuit* (RPP) algorithm slows sharp turns into partially observable dynamic environments (aisles, hallways, intersections) to reduce the likelihood and impact of potential collisions. It also reduces linear velocity in close proximity to obstacles such as people and fixed infrastructure to reduce likelihood of collision in constrained indoor environments. Finally, it additionally includes preemptive collision detection missing from other variations.

Another main contribution of this work is in describing and providing free and high quality implementations of PP, APP, and RPP for evaluation. It is integrated into the Nav2 mobile robot

navigation system commonly used by researchers and adds a new capability to the framework [5]. This work is well documented, tested, and is in use on robots deployed today.

This paper is organized as follows. First, in section 2, we will describe the conventional Pure Pursuit Algorithm, providing a mathematical formulation that will allow us to describe related works that provide variations to the original algorithm. In section 3, we describe the Regulated Pure Pursuit Algorithm, our main contribution to this paper. Section 4 will describe its implementation in a real robotic system, and in section 5, we will experimentally validate the improvement introduced with our contribution. Finally, we will provide the conclusions in section 7.

2 Related Work

Pure Pursuit [7] is a widely used algorithm for path tracking. It is simple but effective. Considering a path \mathcal{P} as an ordered list of points $\mathcal{P} = \{p_0, p_1, \dots, p_n\}$ where $p_i = (x_i, y_i) \in \mathcal{P}$. A local trajectory planner is a function f that determines the linear and angular velocity to track a reference path \mathcal{P}_t at time t (Eq. 1).

$$(v_t, \omega_t) = f(\mathcal{P}_t) \quad (1)$$

Figure 1 shows visual geometric representations of Pure Pursuit. First, it determines the closest point p_r on \mathcal{P}_t to the robot position. Using a given lookahead distance, L , the lookahead point p_l is determined as the first p_i at least L distance away from p_r in Eq. 2.

$$\begin{aligned} dist(p_i) &= \sqrt{(x_r - x_i)^2 + (y_r - y_i)^2} \\ p_l &= p_i \in \mathcal{P}_t, \begin{cases} dist(p_{i-1}) < L \\ dist(p_i) \geq L \end{cases} \end{aligned} \quad (2)$$

With a known p_l , we can determine curvature of the circle (recall, $R = 1/\kappa$) using simple geometry. If \mathcal{P}_t is represented in vehicle base coordinates, \mathcal{P}'_t , where the robot position is the origin, then the curvature can be represented as

$$\kappa = \frac{2 y'_l}{L^2} \quad (3)$$

where κ is the path curvature required to drive the robot from its starting position to the lookahead point, y'_l is the lateral coordinate of the lookahead point p'_l , and L is the desired distance between p_r and p_l .

Figure 1 shows this visually, where L is represented geometrically as the circle's chord. With curvature to drive towards the lookahead point, the commands can be sent to a robot controller. This process is then updated at the desired rate.

The primary parameters of the Pure Pursuit path tracker are simply the velocity of travel v_t and the distance L along the path used to select the lookahead point p_l . In the standard formulation, this lookahead point p_l is a distance L from the robot tuned to achieve an acceptable trade-off between oscillations centered around the path (shorter distances) and slower convergence (longer distances). There exists a broad range of admissible lookahead distances [6].

PP and its variants are applicable to Ackermann and differential-drive robots due to PP's formulation supporting dynamics with longitudinal motion and a turning rate in body-fixed frame. For Ackermann robots to use PP, the path must be feasible given the kinematic-constraints of the platform - while differential-drive robots may follow any holonomic path using PP. While omni-directional robots may utilize PP variants, they will be restricted from performing lateral movements.

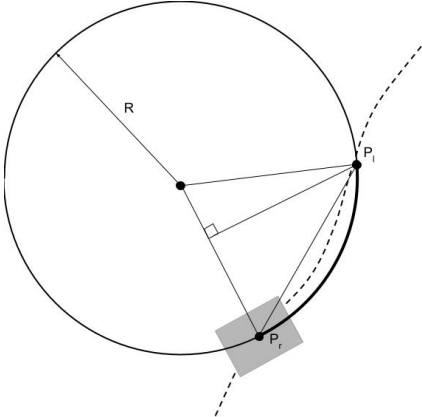


Fig. 1 Geometry of finding the path curvature.

There are several known downsides of this approach. In high curvature situations, Pure Pursuit is known to have overshoot or undershoot

behaviors resulting in path deviations, even in a well tuned system [6]. This is typically not a major concern for autonomous driving applications which naturally has a minimum turning radius limit, but a more substantive issue for smaller-scale applications like industrial and consumer robots. It also does not specify any translational velocity specification criteria during execution. While this can be beneficial as it allows for a great deal of flexibility with different linear velocity profiles, in practice, without described methods, nearly all known variants use a constant translational velocity profile. This is an unsafe defacto-standard.

Variants of this algorithm have been proposed to increase path tracking stability by varying computations of the lookahead point. MIT's entry into the DARPA Urban Challenge implemented the *Adaptive Pure Pursuit* (APP) algorithm for lane following while varying the lookahead distances proportionally to the translational velocity [7]. For velocities in the operating range, a mapping of lookahead distances to velocities is required such that they have an acceptable trade-off between oscillation and slower convergence to the path. A common formulation for this is in Eq. 4, where L_t is the lookahead distance, v_t is the translational velocity, and l_t is a lookahead gain representing the amount of time to project v_t forward [8].

$$L_t = v_t l_t \quad (4)$$

A recent variant has been created to significantly improve path tracking accuracy of Pure Pursuit [?]. This variant also modifies the process of determining the lookahead point, but instead adjusts this point off the path to address the long-standing edge case of short-cutting during path tracking in substantial turns. While it also contains a heuristic change to the translational velocity during that edge-case, its policy is derived from on-road properties and ackermann kinematics that does not generalize to other applications outside of on-road driving of Ackermann vehicles. It does not solve the problems facing deployment of PP techniques to general mobile robotics applications for which this work focuses (e.g. practicable translational velocity control); though its improvements in path tracking accuracies are notable.

Pure Pursuit nor its variations account for dynamic effects of the vehicle. As PP is geometrically derived, vehicle dynamics are not modeled in this style of path tracking algorithm. There are other types of path trackers that do account for vehicle dynamics [9]. This paper uses PP and its variants for comparison to offer a candid contrast with the same or directly analog parameters to clearly indicate the differences in performance due to the contribution ¹.

3 Regulated Pure Pursuit

The Regulated Pure Pursuit algorithm is designed for service and industrial mobile robots in real-world constrained and partially observable environments. It provides methods for adapting the robot's translational velocity to current conditions; robot systems cannot simply barrel through aisles and facilities at full speed without regard. These methods are linear regulation cost functions that provide high-quality behavior across a variety of practical mobile robot environments derived from commonplace requirements on reducing robot velocities in the presence of sharp turns or when operating in confined regions.

The first phase of the Regulated Pure Pursuit algorithm is to transform the input path \mathcal{P}_t $\{p_0, \dots, p_j, p_r, \dots, p_n\}$ into the robot's base coordinate frame \mathcal{P}'_t and prune it. In doing so, determining the curvature of the path is reduced to the simple algebraic expression shown in Eq. 3. Before transformation, p_r is determined and all prior points $\{p_0, \dots, p_j\}$ are permanently pruned from the stored path to prevent unnecessary future transformation of obsolete data. The transformed path $(\{p'_r, \dots, p'_n\})$ is also pruned for all p_i points where $dist(p'_r, p'_i) \gg L_t$ as they are sufficiently far away that they will never need to be considered at t . These far path points continue to be held in the stored path for future iterations, until a new path is received, as the robot progresses along the path.

Next, RPP will utilize the same lookahead selection mechanics as Adaptive Pure Pursuit described in Eq. 4. The lookahead distance L_t is

thusly proportional to the speed v_t and a lookahead gain l_t such that longer distances are used while moving faster. This stabilizes the path tracking behavior over larger ranges of translational velocities [10]. This distance is used to select the lookahead point p_l . While interpolating between path points was found to demonstrably improve smoothness on sparse paths at autonomous vehicle speeds, empirically this did not contribute much benefit at the service and industrial robot speeds using typical grid map planning resolutions (0.025 m - 0.1 m) [8]. However, interpolation is beneficial for use in sparser path resolutions (0.1 - 1.0m).

The desired linear velocity, v_t , is next further processed by the curvature and proximity heuristics. Both heuristics are applied to linear velocity and we take the maximum of the two.

The purpose of the **curvature heuristic** is to slow the robot to v'_t during sharp turns into partially observable environments, such as when entering or exiting hallways and aisles commonly found in retail, warehouses, factories, schools, and shopping malls. This allows for significantly safer traversal when making blind turns. This heuristic is applied to the linear velocity v_t when the change in curvature θ is above a minimum threshold T_θ . This minimum radius restricts velocity scaling in minor turns or path variations that do not require slowing. The curvature velocity v'_t returned by this heuristic is selected as:

$$v'_t = \begin{cases} v_t & \kappa > T_\kappa, \\ \frac{v_t}{r_{min} \kappa} & \kappa \leq T_\kappa \end{cases} \quad (5)$$

Where r_{min} is the minimum radius to apply the heuristic. This formulation is a mathematical reduction of a simple error calculation between the minimum radius and the radius of a circle represented by κ and can be trivially derived.

The **proximity heuristic** is applied to the linear velocity v_t when the robot becomes in close proximity to dynamic obstacles or fixed infrastructure. The purpose of this is to slow the robot when in constrained environments where the potential for collision is particularly high. Reducing the speed near fixed infrastructure lowers the likelihood of collision by lessening the impact of small path variations in tight spaces. Lowering the speed of industrial and service robots in close proximity to dynamic agents, such as humans, is a common safety requirement- allowing a robot to reactively

¹Not leaving the opportunity for misleading results by comparing unrelated families of methods with unknown degrees of tuning

stop faster to prevent potential injury. The linear formulation of this heuristic reduces the speed by the ratio of d_O/d_{prox} with a gain, α , to adjust the response for individual systems. The other formulations tested, such as exponential and quadratic, far too significantly penalized proximity to objects and contained only a narrow band of gains which would result in an acceptable trade-off between proximity to obstacles and velocities to accomplish the robotic task. This linear heuristic in Eq. 6 has a broad range of α which may be finely tuned by a system designer and is derivative of the well-used Adaptive Pure Pursuit’s formulation [10].

$$v'_t = \begin{cases} v_t \frac{\alpha d_O}{d_{prox}} & d_O \leq d_{prox} \\ v_t & d_O > d_{prox} \end{cases} \quad (6)$$

Where d_{prox} is the proximity distance to obstacles to apply the heuristic, d_O is the current distance to an obstacle, and α is a gain to scale the heuristic function for aggressive behavior, with the requirement that $\alpha \leq 1.0$. A higher α lowers the velocity of the robot in proximity to obstacles more expeditiously. The value of d_{prox} should be established based on the system requirements of a robot’s application for how close an obstacle can be before the robot begins slowing its maximum velocity.

After velocity regulation, the algorithm then determines the path curvature using Eq. 1. The angular velocity is computed using the regulated velocity, not desired linear velocity, which prevents consequential undershoot behavior relative to target curvatures [6]. Finally, the angular velocity, ω_t , is then trivially found as Eq. 7.

$$\omega_t = v'_t \kappa \quad (7)$$

The final step of the algorithm is to check our path tracking command for current or imminent collisions, new to RPP. A given angular velocity ω_t and regulated linear velocity v'_t can be projected forward in time, resulting in a circular arc. Points on the arc are sampled at the grid map cell resolution forward for a set duration. Collision checking is done based on a duration to collision rather to the lookahead point such that the robot is always, at minimum, a set duration from collision. At slow speeds, it may not be sensible to collision check to a lookahead point tens of meters, or hundreds of seconds, away. Rather, a temporal schema allows

for fine maneuvers in confined spaces where the current velocity commands may not be admissible a short distance - but long time - away.

This controller is known to converge from the stability analysis completed in [11]. Ollera’s work on the general Pure Pursuit algorithm applies to Regulated Pure Pursuit as well, showing that straight paths and constant curvature paths are stable. Pure Pursuit itself is independent of velocity in its formulation. It is only used in the final step to convert the path curvature into a rotational velocity for a base to track. The regulation heuristics proposed in this variant do not change the basis of stability of the root algorithm.

4 Implementation

Another contribution of this work is a high-quality reference implementation of Regulated Pure Pursuit. This reference implementation is available as one of the default path tracking algorithm plugins in the new and improved ROS 2 Navigation System, known as Nav2. Nav2 is a scalable navigation framework with multiple algorithm implementations, documentation, and support for building modern and reliable research and commercial navigation systems [5]. This optimized C++ implementation has 92% unit test coverage and is used in the experiments in Section 5. It also contained a few additional features that are highlighted in this section.

Rather than creating distinct implementations of Pure Pursuit, Adaptive Pure Pursuit, and Regulated Pure Pursuit, all three have been built into a single reference implementation parameterized to enable each specific behavior. This allows us to test these algorithms easily and analyze their run-time performance more closely knowing that they share the vast majority of their computations. This also allows researchers and developers to quickly evaluate and tune these features by simply changing a handful of parameters from the available binaries.

Additionally, there is a setting to allow the robot to slow its speed when approaching the target goal pose. This feature allows the robot to come to a more gradual stop. When the robot is within close proximity to the goal, the translational velocity is lowered proportional to the remaining distance, up to a minimum viable velocity to make progress.



Fig. 2 The Pal Robotics Tiago used in the experiments [12].

This work is specifically targeted at industrial and service robots, many of which are differential drive. While this is of no technical concern to Pure Pursuit, many differential drive robots will utilize holonomic search-based path planners (such as Dijkstra’s or A*) rather than kinematically feasible planners frequently used by Ackermann vehicles [13]. This is of concern to the Regulated Pure Pursuit implementation as the relative heading of the path may not align with the robot’s starting heading. In this case, it is beneficial for the robot to be able to rotate to a rough starting heading before beginning to track the holonomic path. Further, rotating to the final heading when the robot is within the translational goal tolerances is useful when being deployed in orientation-sensitive applications. These are non-issues when using feasible planning algorithms that, no matter the drivetrain. However, it is useful to consider this situation since holonomic grid search algorithms are popular.

Further, it is a common approach for local trajectory algorithms to leverage a rolling environmental model with the robot at its center. The predictive collision detection algorithm will check for collisions some N seconds in the future given the robot’s current commanded speed. If a near-future collision is detected, the robot is stopped. When a robot is traveling at high speeds, this may actually lapse outside of the bounds of a rolling environmental model if not carefully tuned. The implementation offers protections from such situations and issues stern recommendations to prevent the serious safety violation.

Users are able to tune all of the parameters related to PP, APP, and RPP. In particular, the heuristics have parameters r_{min} , α , and d_{prox} . It

is possible for these to be poorly tuned such that a robot would drop to an impractically low speed when in close proximity to an obstacle or in a turn. To prevent the robot from traveling *too* slow, a minimum speed threshold is exposed such that the robot’s speed will never be set below a parameterized value. Note: the parameters and speeds in the experiment were set up such that this did not impact our experimental results in Section 5.

An open-source community contributed feature made available is support for reversing - not only driving forward. When a path contains a cusp or discontinuity in direction, the controllers will reverse from forward to back and vis-a-vis. When paired with a planner that can generate such paths, such as the Hybrid-A* planner in Nav2, the robot can track more complex paths.

5 Experiments and Analysis

This section describes the experiments to demonstrate the benefits of our contribution compared to other existing Pure Pursuit variants. One experiment was conducted in simulation and three others were conducted with a physical Tiago robot (Figure 2). Tiago is used in industrial and service applications and the test environment is shown in Figure 7, corresponding to a university campus building.

In all three hardware experiments below, the same navigation configurations are utilized. The maximum linear speed v_{max} is set to $0.8m/s$, the maximum acceleration a_{max} is $0.2m/s^2$, and the maximum angular speed ω_{max} is $3.2rad/s^2$. The lookahead distance L_t is set between $0.25m$ and $1.2m$ and lookahead time is $1.0s$ for Adaptive and Regulated Pure Pursuit; while Pure Pursuit uses the maximum of $1.2m$. In all four experiments, no replanning was utilized such that the distance to path data is regular and can be meaningfully analyzed. SLAM Toolbox was used to generate the 2D map and AMCL was used to localize within it [5, 14].

5.1 Path Tracking Experiment

A simulation experiment was conducted to analyze the improvements Regulated Pure Pursuit can offer during sharp turns in ideal conditions. This simulation was conducted using the Turtlebot 3 robot, the Gazebo simulator, and an empty

environment with ground truth information made available to remove the contribution of odometric and localization error. The robot followed the reference step function path indicated by the red piece-wise line in Figures 3 and 4 using the three variants of Pure Pursuit.

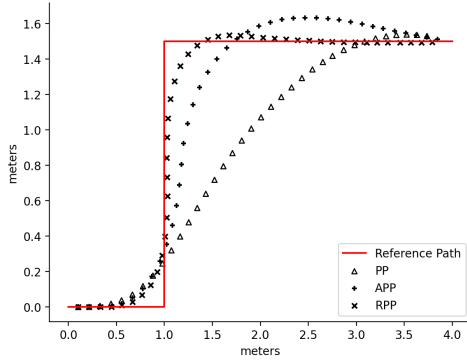


Fig. 3 Comparison of variants of Pure Pursuit.

Figure 3 displays the comparison with the different methods. The same parameters were utilized in this experiment as the hardware experiments, with the exception of the desired linear velocity, set to $1.0m/s$ and the curvature heuristic minimum radius, set to $1.5m$. The overshoot observed during sharp turns was notably lower in Regulated Pure Pursuit due to the controlled slow down during sharp changes in curvature allowing the robot to track the path more finely. The slow down also triggers the use of a closer adaptive lookahead point further increasing local stability. The mean tracking error from the reference path of Regulated Pure Pursuit was merely $0.03m$. The others experienced over an order of magnitude difference at $0.10m$ and $0.19m$ for APP and PP, respectively. While the main contribution of this work is *not* an improvement on path tracking, it is a convenient emergent property to further ensure safety. This experiment showcases that the algorithm can follow a computed collision-free path in confined spaces more closely and thusly more safely.

Using the ideal environment, the effects of varying the curvature heuristic’s minimum radius were also studied. From Figure 4, it can be observed that larger values of r_{min} allow a robot to follow the reference path more closely. This is

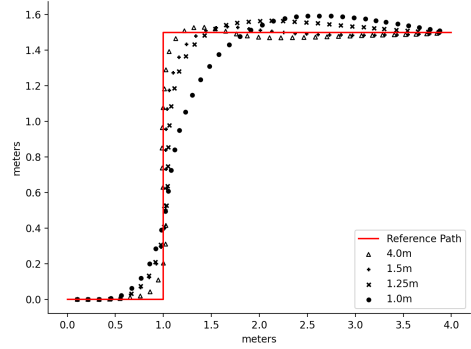


Fig. 4 Effects of varying r_{min} in Regulated Pure Pursuit.

due to the curvature triggering reactions quickly on approach to the discontinuity. When r_{min} was increased from $1.0m$ to $1.5m$, the mean tracking error decreases from $0.8m$ to $0.3m$. This reduction in tracking error does come at a cost: a reduction in the speed resulting in longer navigation times. For this experiment, when $r_{min} > 1.5m$, the improvement on tracking error became insignificant, but navigation times begin to spike as the speed during turns trends towards the minimum allowed speed.

5.2 Blind Turning Experiment

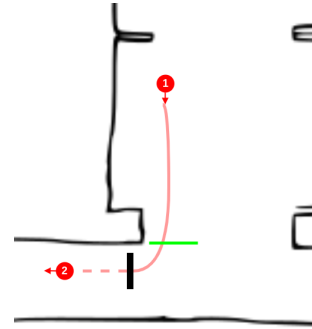


Fig. 5 A blind corner experiment with an obstacle just out of view. When the robot cross the green line, the obstacle is set.

This experiment evaluates the three algorithms’ performance when the robot encounters an unexpected obstacle that blocks its path around a blind corner. The experiment consists of the robot making a sharp 90-degree turn (Figure 5) behind which there is an obstacle. The obstacle is such that the robot cannot observe it before beginning

to execute the curve and has minimal time to react, placed after the robot crosses the green line.

For each algorithm, the experiment is repeated ten times. The purpose of this experiment is to test the reaction of each algorithm in a common but particularly dangerous situation. While PP and APP do not themselves contain collision detection capabilities, for a fair comparison, each are evaluated using the collision checking method described in Section 3.

Table 1 Result of the Blind Turning Experiment.

	PP	APP	RPP
Avg Stopped Distance (m)	0.15	0.16	0.24

Table 1 shows the results of this experiment; in all cases, none of the methods resulted in a collision. This shows that the collision detection features of RPP (applied to all methods) are sufficient to prevent blind collisions at service robot speeds. A similar experiment without RPP’s collision detection yielded collisions in nearly all instances. The RPP algorithm displayed a 33% increase in average stopped distance relative to APP. APP and PP, both which do not change their speeds in turns, had expectedly similar behavior. This is an improvement in reaction time for safe navigation in environments with many dynamic agents potentially cornering at the same time. While the 8cm increase does not seem like much, with many robots navigating independently for their own tasks, this additional precaution is the difference between robots scraping by and colliding with permanent damage. This highlights that RPP’s slowing of the robot in blind turns helps to reduce risk in partially observable settings.

5.3 Confined Corridor Experiment

The confined corridor experiment was designed to evaluate how well the proposed approach improves performance in narrow spaces at high speeds when making continuous changes in direction. Figure 6 shows the test environment map, creating a "slalom" pattern, starting from the left. The total width of the corridor is 1.5 meters and the obstacles each have dimensions of roughly 0.7 m. Each algorithm was tested five times and the data from this experiment is shown in Table 2.

Table 2 Result of the confined corridor experiment.

	PP	APP	RPP
Time (s)	13.0	12.7	13.3
Distance (m)	9.04	9.54	9.63
Collisions	0	0	0
Average Speed (m/s)	0.679	0.736	0.682
Average Distance Obstacle (m)	0.662	0.681	0.683
Average Distance to Path (m)	0.100	0.059	0.052

Figure 6 displays the robot position data overlaying the trajectory each algorithm was asked to follow. The most telling region of this experiment to analyze is the final and sharpest turn. RPP’s proximity and curvature heuristics contributed in this location and resulted in a reduced speed and 14% less path tracking error than APP. The other algorithms had path undershoot due to the sharp turn and came into much closer proximity to the obstacle and wouldn’t be considered functionally safe. This undershoot was corrected in RPP by reacting to the increased curvature partially due to the deviation from the path, compounded by its approaching proximity to the obstacle, slowing the robot down. This allowed RPP to better recover from a minor deviation off the path and navigated safer distance away from the obstacle. The impact of the regulation heuristics can be seen in blue as the robot weaves around the final obstacle after quickly changing direction.

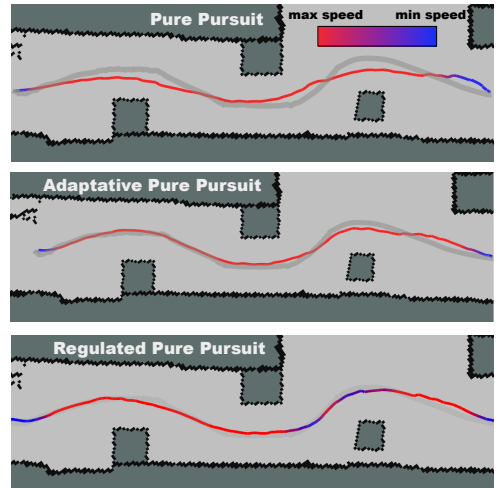


Fig. 6 Calculated path in gray and the robot’s traveled path, colored by speed, during the corridor experiment.

The distance traveled by PP is lower than both variants due to the short cutting it displayed

throughout the trajectory in Figure 6. This habitual undershoot also led to an decreased average distance to an obstacle - passing much closer to potential collisions.

5.4 Full-system experiment

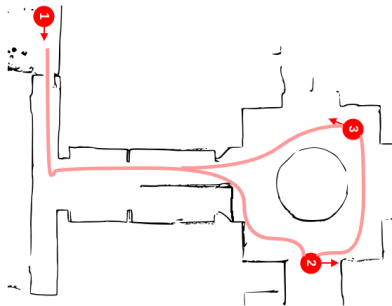


Fig. 7 Itinerary for full-system experiment. The total length is 70 meters.

In the final experiment, the robot will follow a route through the three control points shown in Figure 7 in a campus building. The route contains collision-free open space, confined halls, and blind sharp turn of approximately 70 m. The robot repeats the path three times with each of the algorithms, the data is provided in Table 3. The purpose of this experiment is to compare the RPP algorithm’s general behaviors at a system-level to the existing variants.

The data indicates little difference in overall system-level behavior - the average time to navigate was between 85-95s. Consistent with the previous experiment, PP does display slightly shorter distance traveled and an increased tracking error - corresponding to path short-cutting.

Table 3 Result of the full-system experiment.

	PP	APP	RPP
Time (s)	85.6	94.3	88.6
Distance (m)	58.24	59.26	58.64
Collisions	0	0	0
Average Speed (m/s)	0.661	0.675	0.646
Min Distance to Obstacle (m)	1.139	1.133	1.135
Average Distance to Path (m)	0.062	0.049	0.043

The most compelling attribute of this experiment is the lack of particularly unique outcomes between the algorithms. Although RPP slows the

robot in narrow corridors and while completing sharp turns, surprisingly, these maneuvers did not meaningfully impact the high-level navigation metrics. Time to task completion and average speeds were consistent across all three algorithms when considering the standard deviations between the trials. The system-level performance of RPP is so similar to that of APP, we conclude that the additional benefits of RPP come at little disadvantage to a system designer. In fact, it is possible for a solution architect to increase the maximum speed of the robot modestly when using RPP due to its slowing in turns and in proximity to obstacles (the common limiting factors of speed in robotics applications). This meaningfully improves the overall efficiency of a robot system while securing higher-quality safety features.²

6 Limitations

Regulated Pure Pursuit, and other Pure Pursuit variants, suffer from a lack of modeling of the vehicle. As this class of technique is purely geometric based on the path, it will not consider dynamic constraints while changing velocities to track the path. Further, the global path must be feasibly drivable for a given robot platform since these methods compute velocities in the absence of kinematic limitations. For differential-drive robots, this may be any path. However for Ackermann steering robots, this path must already be drivable considering the kinematic limitations on the minimum possible turning radius.

Regulated Pure Pursuit also continues to short-cut sections of paths with high curvature turns, though to a much reduced degree than PP or APP. The work introduced in [?] provides an alternative method for selecting the lookahead point with the goal of more accurately tracking the reference path. While this work also introduces a velocity heuristic that is not suitable for our class of applications, a variation on the method for selecting lookahead points has merit to be applied in mobile robotics.

²A video with the experiments with the real robot can be found at <https://youtu.be/LQAzzJ8GmS0>

7 Conclusion

Regulated Pure Pursuit builds incrementally on Adaptive Pure Pursuit with a focus on service robots. This method contains a schema for selecting velocities that improves functional safety for real-world deployed robot applications.

Our demonstrations on an industrial robot exhibited improved safety around blind turns and in confined settings without significant system-level changes. A high-quality implementation of Regulated Pure Pursuit is freely available at <https://github.com/ros-planning/navigation2> and is in use on robots today.

Conflict of Interest and Acknowledge

The authors have no conflicts of interest to declare that are relevant to the content of this article.

This work has been partially funded by Ministerio de Economía and Competitividad of the Kingdom of Spain under project PID2021-126592OB-C22 and by the European Commission under grant CoreSense (N1.101070254).

References

- [1] Fox D, Burgard W, Thrun S. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*. 1997;4(1):23–33. <https://doi.org/10.1109/100.580977>.
- [2] Rösmann C, Hoffmann F, Bertram T. Timed-Elastic-Bands for time-optimal point-to-point nonlinear model predictive control. In: 2015 European Control Conference (ECC); 2015. p. 3352–3357.
- [3] Huskić G, Buck S, Zell A. GeRoNa: Generic Robot Navigation. *Journal of Intelligent & Robotic Systems*. 2019 Aug;95(2):419–442. <https://doi.org/10.1007/s10846-018-0951-0>.
- [4] Jelavic E.: Planning and control for car-like vehicles. [Online; accessed 3-March-2022]. https://github.com/leggedrobotics/se2_navigation.
- [5] Macenski S, Martin F, White R, Ginés Clavero J. The Marathon 2: A Navigation System. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2020. .
- [6] Coulter RC. Implementation of the Pure Pursuit Path Tracking Algorithm; 1992. .
- [7] Campbell SF. Steering control of an autonomous ground vehicle with application to the DARPA Urban Challenge. Massachusetts Institute of Technology, Department of Mechanical Engineering.. Massachusetts Ave, Cambridge, MA 02139, USA; 2007.
- [8] Ohta H, Akai N, Takeuchi E, Kato S, Eda Hiro M. Pure Pursuit Revisited: Field Testing of Autonomous Vehicles in Urban Areas. In: 2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA); 2016. p. 7–12.
- [9] Samuel M, Hussein M, Mohamad M. A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicle. *International Journal of Computer Applications*. 2016;135:35–38.
- [10] Ohta H, Akai N, Takeuchi E, Kato S, Eda Hiro M. Adaptive Pure Pursuit Model for Autonomous Vehicle Path Tracking. *International Journal of Science*. 2017;4(3):70–73.
- [11] Ollero A, Heredia G. Stability analysis of mobile robot path tracking. In: Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots. vol. 3; 1995. p. 461–466 vol.3.
- [12] Robotics P.: Tiago. [Online; accessed 21-Feb-2021]. <http://pal-robotics.com/robots/tiago/>.
- [13] Paden B, Čáp M, Yong SZ, Yershov D, Frazzoli E. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on Intelligent Vehicles*. 2016;1(1):33–55. <https://doi.org/10.1109/TIV.2016.2578706>.

- [14] Macenski S, Jambrecic I. SLAM Toolbox: SLAM for the dynamic world. *Journal of Open Source Software*. 2021;6(61):2783. <https://doi.org/10.21105/joss.02783>.