3.3 Parameter Tuning for Regulated Pure Pursuit (RPP)

In the previous sections, DWB and MPPI were discussed as dynamic methods that generate multiple trajectories and use *critic* functions to select the optimal one. In contrast, geometric controllers like the RPP algorithm strictly follow the global path without deviation. While this approach ensures precise navigation in controlled environments, it becomes problematic with dynamic obstacles, as the planner stops and waits for a new path or for the obstacle to move, rather than actively avoiding it. Their simplicity allows them to evaluate geometric, kinematic, and control-law-derived functions very quickly, often exceeding 1 kHz in operation speed [32]. This subsection will address this issue by optimizing the parameters to enhance performance in navigating both static and dynamic environments, ensuring a balance between strict path adherence and adaptive obstacle avoidance.

To begin with, as detailed in 2.2.3, the primary aim of the RPP method is to prune the global path, transform it to the robot's local coordinate frame, and assign a lookahead point based on the distance defined by the *lookahead distance* d_{L} parameter. Setting a large value beyond $d_{\mathrm{L}} = 0.8\,\mathrm{m}$ demonstrated poor performance following the global path. The robot tends to deviate from the path, increasing the risk of collisions over time, especially when navigating around obstacles or taking sharp turns. This deviation occurs because increasing d_{L} reduces the curvature based on equation 1. Reducing the lookahead distance helps the robot stay closer to the path generated at the edge of the inflation radius. However, this causes the robot to rotate more frequently to follow the lookahead point, resulting in the robot braking each time an angular velocity is applied. The following Figuredisplays the robot's trajectory and shows the effect of changing d_{L} to observe the behavior during obstacle avoidance.

It can be observed in Figure 17 that higher values of d_L cause the robot to move closer to obstacles. Values starting from 0.5 m result in the robot approaching obstacles more closely, increasing the risk of collisions. On the other hand, values below 0.5 m maintained a safe distance from obstacles but caused the robot to oscillate along the path. This occurred because using lower d_L led to frequent corrections, causing the robot to rotate excessively to stay aligned with the global path. Consequently, a value of 0.5 m is chosen as it demonstrated better performance in avoiding obstacles and maintaining proximity to the global path without causing oscillations.

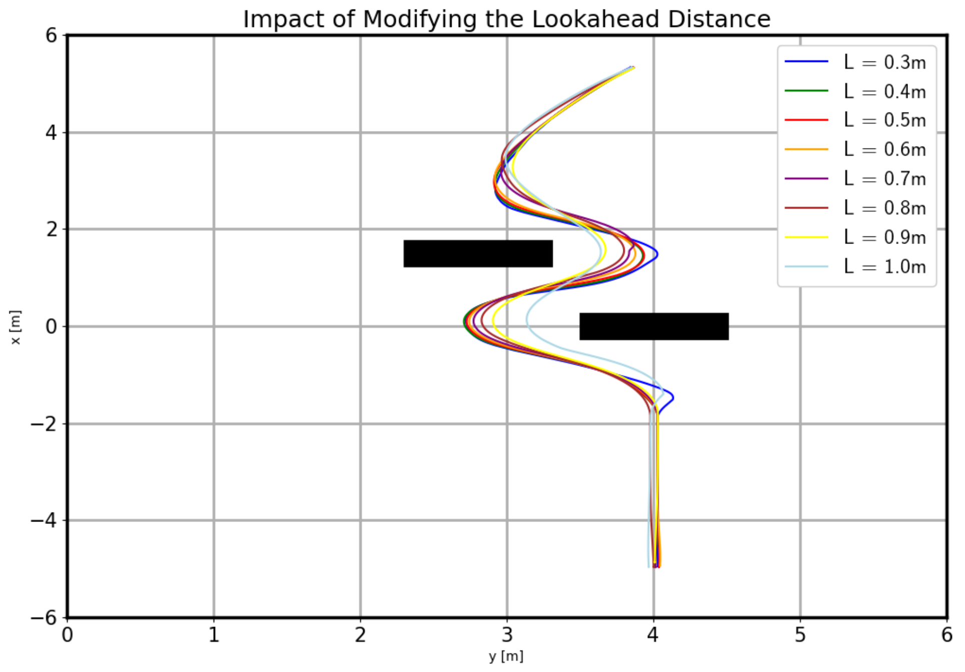

Figure 17: *Effect of changing the lookahead distance*

Additionally, a time parameter for the maximum allowable time before a potential collision, denoted as t_carrot, is implemented. This parameter projects the current velocity command to anticipate possible collisions, allowing the robot to stop and wait for further instructions. The robot can either wait for a re-planned global path around the inflation radius or activate a recovery behavior. These recovery behaviors include spinning in place to compute a new path from a different heading, waiting a few seconds for an obstacle or pedestrian to move away, or backing up a certain distance to re-compute the path from a position away from the obstacle. Moreover, enabling collision detection $B_\mathrm{collision}$ is necessary to use this feature. When setting this parameter, there is a maximum limit on collision inspection relative to the lookahead distance. A value of 0.785 s was chosen to project the command velocity to the maximum lookahead distance of 0.5 m. This value is based on the current speed and projects forward until the lookahead point, which can be calculated using the following equation. This parameter is constrained by a limit on inspection up to the lookahead distance; increasing the time beyond this limit will have no effect other than checking for collisions up to the lookahead point.

$$t_{\mathrm{carrot}} = \frac{d_{\mathrm{L}}}{V_{\mathrm{max}}} \tag{9}$$

Furthermore, enabling the rotation to align on the current global path heading B_{heading} allows the robot to align approximately with the lookahead point current heading for a more efficient start. However, it is essential to disable the reversing feature B_{reverse} which could result in an unclear situation. Initially, the difference between the current heading of the robot and the heading of the global path is calculated. If this difference exceeds the minimum threshold angle A_{thres}, set at 0.785 rad, the robot will continue to rotate until this difference is reduced to within the threshold. During this process, the robot prioritizes angular velocity over linear velocity, to achieve the correct heading alignment before moving forward.

In the ROS 2 implementation of this motion planner, two main features are implemented for dealing with obstacles or highly curved paths. These improvements regulate the current linear velocity to a minimum value V_{reg}. This parameter sets the minimum speed the robot will reach when either safety feature is triggered. To enhance safety, half of the maximum velocity is used as the minimum speed the robot will adjust to in such situations. The following two sub-sections discuss both features in more detail.

3.3.1 Proximity Heuristic

The proximity heuristic is a method designed to reduce the speed of the robot when it approaches a dynamic or sudden static obstacle. To enable this feature the boolean parameter to trigger the proximity feature H_{prox} should be set to *true*. This feature relies on two main parameters: the threshold distance between the robot and the obstacle d_{prox} to trigger the scaling of the linear velocity and a multiplier gain α. According to equation 3, d_{prox} represents the proximity distance to the obstacle at which the feature starts to get activated. As stated in [23], this value should be set to a value lower than the inflation radius, as the minimum distance to the obstacle is derived from the inflation costmap layer and will always be compared to d_{prox}. A threshold of 0.4 m is set to activate this feature when a dynamic obstacle is detected in the inflation zone, quickly reducing the robot's speed. However, in narrow spaces between obstacles, it's better to disable or reduce this feature to avoid slowing down the robot unnecessarily. Therefore, it's best to use this feature only in environments where high traversal costs are constant.

On the other hand, the multiplier gain determines how aggressively the velocity of the robot should be reduced to reach the minimum regulated speed. Setting this gain to a value lower than 0.5 will cause the robot's current velocity to decrease rapidly to the minimum value. However, in certain situations, the robot might not exactly reach the regulated minimum speed. This change occurs because the robot's distance from the obstacle can vary, potentially increasing, which deactivates the feature and allows the robot to accelerate back to its maximum speed. Setting the value lower than 0.5 has two advantages. First, the robot will reduce its speed before getting too close to the obstacle, providing smoother and more flexible maneuvering around the obstacle, and ensuring safe rotations.

Second, as the robot slows down, it gives additional time for the global path to re-plan which the algorithm highly depends on, allowing the robot to readjust its course more effectively. Therefore, a value of 0.4 is assigned. The following Figuredemonstrates how the robot's velocity decreases once the proximity distance is greater than the minimum distance to the obstacle when the robot is navigating through two close obstacles.

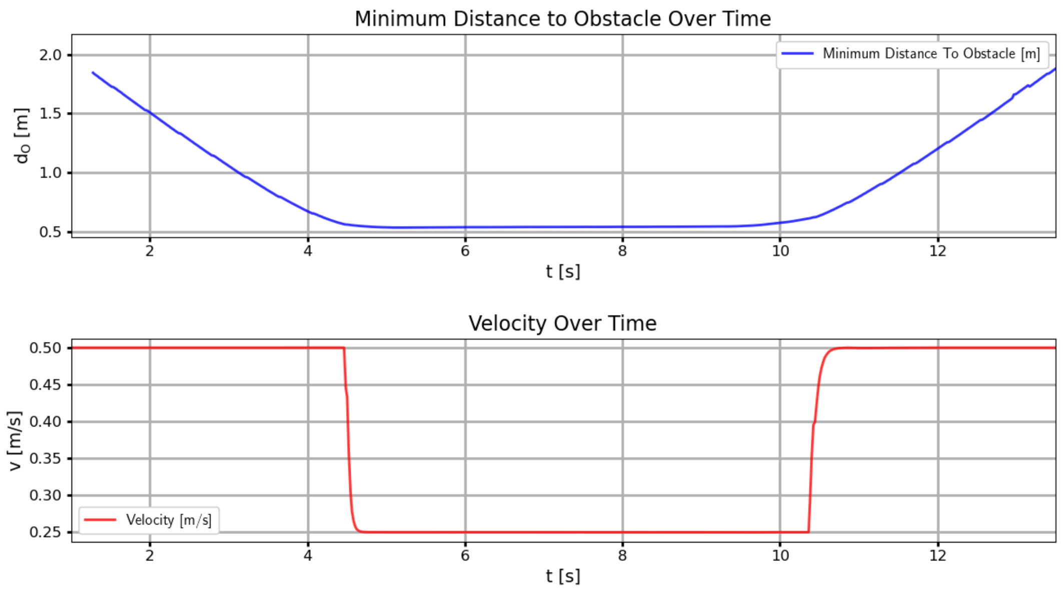

Figure 18: *Robot behavior when $d_{prox} > d_O$, at $d_{prox} = 0.6\,\mathrm{m}$*

At $t = 4.3\,\mathrm{s}$, depicted in Figure18, the robot is positioned close to the obstacle, entering the inflation radius with a value of 0.7 meters. Thus, this triggers the proximity feature due to the distance to the obstacle falling below the threshold of a value of 0.6 m. As a result, the robot's velocity decreases to the regulated minimum speed, reaching $0.25\,\mathrm{m\,s^{-1}}$. As the robot starts to exit this region at $t = 10.3\,\mathrm{s}$, its speed begins to increase and returns to its maximum velocity.

3.3.2 Curvature Heuristic

The curvature heuristic is a feature activated when the robot performs sharp turns, significantly enhancing safety during blind turns. To enable this feature the boolean parameter to activate the curvature heuristic H_{curv} should be set to true. This feature primarily relies on the turning radius for which the regulation features are triggered, denoted as r_{k} in equation 2. Activation depends on both d_{L} and the y-coordinates of the lookahead point the robot aims to reach. For example, with a lookahead distance of $0.7\,\mathrm{m}$ and a lookahead point y value of $0.4\,\mathrm{m}$ which means that the point is located at least $70°$ counterclockwise from the heading of the robot, from equation 1 the curvature is $1.63\,\mathrm{m\,s^{-1}}$. The inverse of this curvature yields a radius of $0.45\,\mathrm{m}$. This radius is then compared with r_{k} to determine the velocity according to equation 2.

Furthermore, increasing r_{k} enhances the smoothness of the robot's motion, as it reduces the speed of the robot more significantly when taking turns or navigating around obstacles. This increased smoothness, however, comes at the cost of longer navigation times. The speed of the robot decreases, reaching a value of $0.25\,\mathrm{m\,s^{-1}}$, which is the regulated speed the robot will move with when performing the sharp turn. As the heading of the robot aligns more closely with the lookahead point, the speed gradually increases, eventually reaching the maximum speed when the curvature exceeds again r_{k}. To balance these trade-offs, a threshold value of $0.7\,\mathrm{m}$ is selected for the minimum radius. This parameter ensures that navigation times remain efficient by preventing significant speed reductions during turns and avoiding unnecessary slowing down when encountering dynamic obstacles.

In summary, increasing the minimum radius parameter will slow down the speed of the robot more frequently during rotations or when navigating curved paths. However, decreasing this value will prevent the feature from being triggered unless the robot encounters a very sharp turn with a low curvature. The frequent activation of the feature was avoided by setting the minimum radius parameter in relation to the lookahead distance and the y values of the lookahead point to $0.7\,\mathrm{m}$. This adjustment helps reduce the overall time taken for the robot to complete its navigation process. In Table 14 the RPP parameters can be visualized.

Table 9 offers a comprehensive comparison of the three local trajectory planners: DWB, MPPI, and RPP. It covers their controller types, methods for selecting optimal paths, factors influencing the number of generated trajectories, capability for