3.2 Parameter Tuning for Model Predictive Path Integral (MPPI)

This section outlines the key parameters involved in MPPI and provides guidelines on how to adjust them for improved control and trajectory planning. The MPPI also uses critic functions to score the generated trajectories as mentioned in 2.2.3 which will be further discussed.

3.2.1 MPPI: Forward Simulation

MPPI is a predictive that utilizes temporal information to compute optimal paths over time. Unlike the DWB, which relies on the current environment, predictive controllers update the previous time step's trajectory with each new request by creating a detailed plan for the robot's movement over time. They model trajectories as sequences of velocity instructions for future intervals, predicting the robot's future positions and movements. This enables them to avoid local minima and reduce rapid velocity changes and other undesirable behaviors, resulting in more intelligent and robust navigation.

A key function of the MPPI is the prediction of future states to achieve the best possible control performance for trajectory planning and obstacle avoidance. The prediction horizon plays an important role in this process. This is referred to as the time window in which the algorithm predicts the system's behavior. Two main parameters controlling the size of the prediction horizon t_{horizon} are the sampling time t_{sampling} and the number of prediction steps N_{step}. The sampling time represents the time between two consecutive points, while N_{step} represents the total number of points that make up a trajectory. It is crucial to set t_{sampling} to a value that matches the frequency of the MPPI. Therefore, a value of 0.05 s is chosen to correspond to the planner's controller frequency, which in this case is 20 Hz. To decide the size of t_{horizon} equation 8 is used, therefore setting the N_{step} to 60 produced a 3 s horizon. Increasing both parameters extends the prediction horizon which can improve long-term planning, however, this will come with a cost of global path misalignment.

$$t_{\text{horizon}} = t_{\text{sampling}} \cdot N_{\text{step}} \tag{8}$$

If the horizon is too short, the MPPI may lack enough foresight to make valid choices, especially in circumstances where actions have long-term effects. In contrast, if the horizon is too long, computational complexity might become costly, particularly as MPPI involves sampling many trajectories. Furthermore, projections for the far

future may be less reliable due to uncertainty in system dynamics or disturbances such as the accumulation of noise and disturbances that significantly deviate the system from the predicted path.

Moreover, the prune distance d_{prune} parameter needs to be configured as explained in the 3.1.1 to avoid limiting the maximum speed of the robot when evaluating the trajectories. Using equation 14, d_{prune} is set to 1.5 m when choosing a t_{horizon} of 3 s and a V_{max} of $0.5\,\text{m}\,\text{s}^{-1}$.

3.2.2 MPPI: Critic Functions

Although, both the DWB and MPPI planners evaluate the generated trajectories using critic functions, each controller has its way of scoring the cost of the trajectory for each critic. There are 7 different critics which are going to be discussed in the following:

- **Path Follow Critic** P_{Follow}: is responsible for calculating the distance between the last point on the generated trajectories and the furthest point on the global path that the algorithm predicts can be reached.
- **Path Align Critic** P_{Align}: Scores trajectories based on alignment to the global path by calculating the distance between every nth point (determined by the "trajectory_point_step" parameter) and the nearest point on the global path. The distances are averaged and multiplied by a cost scale. The trajectory with the lowest cost is selected.
- **Path Angle Critic** P_{Angle}: This critic is triggered when the angle between the robot's current direction and the direction to the global path exceeds a certain angle.
- **Cost Critic** C_{Cost}: is responsible for checking if the generated trajectories are in collision or not. This happens by evaluating each trajectory by summing the costs of the points along the trajectory. Costs are retrieved from the costmap, and higher costs are assigned for points near obstacles or in a collision. The trajectory with the lowest total cost is selected as the best trajectory for the robot to follow.
- **Goal Critics**: Those critics are triggered as the robot gets closer to the target goal within a certain threshold, and the **Goal Align Critic** G_{Align} and **Goal Critic** G_{cr} are given the highest priority for scoring trajectories. Their main objective is to encourage moving towards the goal and achieving the correct angle.

- **Constraint Critic** C_{cons}: This critic evaluates the velocity of the trajectory point and compares it with the maximum and minimum velocity of the robot. A penalty is given for trajectories that violate the constraints of the robot's dynamics.

Setting the weight γ for each critic determines the influence of that critic on the evaluation process. The Path critic functions are the main features in controlling the behavior of the robot throughout the whole navigation process. Starting with the *Path Angle* critic which is crucial when using the MPPI controller. Disabling this critic can cause issues when the robot is given a goal behind it. In such a scenario, the robot will continue to move backward because it believes that it's already aligned with the global path during the triggering of both P_{Follow} and P_{Align} critics. Enabling the *Path Angle* critic addresses the issue by allowing the controller to detect the error between the robot's heading and the global path. This enables the robot to rotate in order to minimize the heading error. When handling dynamic obstacles, setting the P_{Angle} critic weight to $\gamma = 4.0$ is sufficient to activate this feature effectively. However, assigning a higher weight may restrict the robot's backward motion in scenarios where it could benefit from some space to move backward. This restriction would cause the robot to rely more on trajectories that allow it to rotate in place, rather than providing more flexibility to adjust its heading with a safer backward motion.

Moreover, the *Path Align* critic was assigned a weight of $\gamma = 14.0$ to prioritize trajectories aligning the robot with the global path. However, if the backward motion is enabled and an obstacle suddenly appears in front of the robot, it may move backward to create space for rotation away from the obstacle. Due to dependence on P_{Align}, the robot may vigorously move towards the global path, potentially bringing it closer to obstacles within the inflation radius even after moving backward to provide more space for the robot to maneuver. Assigning the *Cost critic* weight to 6.0 penalizes collision-prone trajectories more heavily by providing more time to move backward and rotate safely away from obstacles without entering the inflation layer again. However, this feature can lead the robot to avoid narrow spaces, even if it could navigate through them, resulting in unnecessarily longer paths to avoid going inside. The *PathFollow* critic addresses this issue by setting its weight to $\gamma = 10.0$, ensuring that in some situations, the robot follows the path through confined spaces while still considering trajectories penalized by the C_{Cost}. Additionally, it is beneficial to set a lower weight for the P_{Follow} feature compared to the P_{Align} feature. This is because using a lower weight for P_{Follow} helps to prevent it from overly influencing the overall evaluation of the trajectories. If P_{Follow} has too much influence, the robot might select trajectories that, although closer to the desired path, enter the inflation zone, increasing the risk of collision.

Furthermore, Goal critics influence the navigation process only when the robot reaches a specific threshold distance from the goal, denoted as d_{thres}. This parameter is consistently applied across all Path critics to ensure uniform behavior. Once the robot is within this threshold, other critics are deactivated, allowing the Goal critics to take priority and guide the robot directly to the target.

In this context, d_{thres} is set to 1.0 m, meaning that when the robot is within 1.0 m of the goal, the planner will prioritize the Goal critics over the rest. This ensures precise navigation to the target, especially when dynamic obstacles and backward motion are involved. Increasing both weight critics more than $\gamma = 6.0$ didn't show much effect in the end while reaching the target, thus this value was chosen. However, in some cases, the user may prioritize achieving the goal precisely. In certain situations, though, the robot might overshoot the final destination due to the deactivation of the path critics once the goal critics take the lead. To mitigate this, the d_{thres} can be set to a lower value, ensuring that the robot is as close as possible to the goal with the best possible heading before the goal critics take priority. This approach helps the robot achieve precise goal localization without overshooting, allowing for finer control over the final positioning.

Additionally, the *Constraint* critic is utilized, in scenarios where the robot might violate velocity constraints during sharp turns. This is important because sudden obstacles could appear, and if the robot is moving too fast, it might not have enough time to reduce its speed, potentially leading to collisions. To summarize, effective parameter adjustment for the MPPI is critical for improving robot behavior and trajectory planning. To enable optimal navigation, key factors such as the prediction horizon, critic functions, and weights must be properly tuned. The MPPI controller achieves reliable and efficient path planning by balancing setting the prediction horizon parameters to manage computational load and future state prediction, as well as assigning appropriate weights to critic functions to maintain path alignment, avoid collisions, and respect dynamic constraints. Tables 12 and 13 represent the tuned parameters: