Benchmarking Mobile Robot Local Motion Planning

Maria Salgueiro
Instituto Superior Técnico
Lisbon, Portugal

***Abstract*—Benchmarking, specifically in the robotics field, gained an undeniable relevance due to its use in comparing and evaluating results from different experiments. Despite the relevance and recognized importance, benchmarking is barely used in current robotics research. In this specific field, the formulation of a general assessment of performance is already difficult due to the complexity of the robot systems. The benefits of such general assessments would bring great benefits to robotics research, either for beginners who would have an easy start in their journey in the field or for experienced researchers who would have it easy when comparing results of different approaches, but also when assessing the quality of the research presented by them.**

This work presents a benchmarking framework for local motion planners. Despite the fact that mobile robot local planning approaches have been thoroughly investigated, the lack of an unified benchmark for performance evaluation remains. This aims to facilitate the comparison of planners by helping to select the one that performs best on some problems of interest.

In order to solve this, we created a simulated version of a differential robot model and multiple simulated environments with different characteristics. In order to evaluate the performance, different metrics were designed.

We applied this benchmark in two of the most popular local planners, Dynamic Window Approach and Pure Pursuit. The results obtained show differences in the performance depending on the type of planner and in which environment the local planners were used. With this, the decision of the local planner can be done accordingly.

I. Introduction and Motivation

In the last few years there have been significant progress in multiple fields, namely in machine learning, computer vision and speech processing. Multiple reasons can explain this progress, but there is one that tends to be overlooked: the development of a conducive research ecosystem. [2] The existence of multiple platforms like PyTorch and TensorFlow reduces the difficulty of starting research in those areas. These common platforms and datasets have led to standardized evaluations and benchmarks which translate into a better progress quantification in those areas. [2]

When it comes to robotics the progress and exciting innovations are also inarguable. However, when compared to other research areas it is harder for a new researcher to get started and contribute to progress in the field due to the lack of standard development tools.

As described in [3] the validation of robotic research, a few years ago, was relatively straightforward. If a researcher had published some of his work in a journal mentioning, for example the equations and solver details of the software used it was possible for the readers to re-run numerical simulations in a short period of time. People in the field would have the chance to test, validate and possibly generalize the algorithm. A lot has changed in the robotics field since then. Machines are much more complex and the robots no longer live confined to specific environments, but are present in our daily life, from our houses to airports and even Mars.

The progress in the field of robotics have to, and will continue and it is our duty to understand how we can change the method of research in a way that leads to more and faster innovation. Quoting Steven Johnson [13]: "If you look at history, innovation does not come just from giving people incentives; it comes from creating environments where their ideas can connect". The research community needs to exchange experiences, algorithms, pieces of code to replicate the experiments, find bugs or theoretical flaws, and to *benchmark*. This last is the specific theme we address here.

At [16] benchmarks were defined as "process of identifying the highest standards of excellence for products, services and processes and then making the improvements necessary to reach those standards", the definition presented in [7] adds numerical evaluation of results (performance metrics) as key element.

Benchmarks are important instruments used to compare and evaluate results of experiments. They can play an important role in various aspects, several areas of research require benchmarking to assess the value of new results against some reference performance. Benchmarking can also be of great help to scientists and technical developers, establishing clear technical targets to demonstrate a certain readiness level of the technology. [5]

In this work the focus will be on the benchmarking of a specific functionality: robot local motion planning.

Local path planning means that the path planning is done while the robot is moving, which means that the algorithm is capable of producing a new path in response to changes in the environment. If no obstacles are on the way the shortest path between the start point and the end point is a straight line. Therefore the robot will proceed along this path until an obstacle is detected. At this point the algorithm responsible for the path planning must be able to find a feasible path around the obstacle. After avoiding the obstacle, the robot continues to navigate toward the end-point along a straight line until another obstacle is detected or the end position is reached. [14]

II. Local Planners

A. Dynamic Window Approach

The DWA is a local motion planner for mobile robots proposed in [6]. Unlike other methods of this type, DWA is derived directly from the dynamic features of the robots and is designed with the goal of dealing with constraints imposed by limits on velocity and acceleration of the robot. In this approach the search for commands controlling the robot is carried out directly in the space of velocities. [6]

With mathematical manipulation it is possible to approximate the trajectory of a robot by a sequence of circular arcs, commonly referred as curvatures. Each of these curvatures is determined, in an unique way, by a velocity vector (v_i, w_i), where v_i is the linear and w_i the angular velocity. From this point on we will refer to this vector as velocity. In order to generate a trajectory to a given goal, for the next n time intervals, velocities have to be determined, one for each of the n time intervals. This would result in a search space for these vectors exponential in the number of the considered intervals.

To avoid this complexity and make the optimization feasible, the DWA considers a short time interval and assumes that the velocities will stay constant in the remaining time intervals, which equals to assume zero accelerations in that time interval. This approximation is done based on three observations: the reduced search space is two dimensional, which easily allows to reach a solution. The search is repeated after each time interval and velocities will stay constant unless new commands are given. [6]

Restrictions on the velocity are imposed when there is an obstacle in the closer environment of the robot. For example some pairs of velocities could lead to a collision if followed by the robot. Considering V_s the space of possible velocities (both linear and angular) we can state that inside V_s are some pairs of velocities that correspond to trajectories that are not allowed and other pairs that are admissible. A velocity is considered admissible if the robot is able to stop before it reaches the obstacle. Having this said, the set of admissible velocities V_a, is defined as

$$V_a = \{(v,w) \mid v \leq \sqrt{2\ dist(v,w)\ \dot{v_b}} \ \wedge \ w \leq \sqrt{2\ dist(v,w)\ \dot{w_b}}\} \tag{1}$$

where $dist(v,w)$ is the distance to the closest obstacle on the corresponding curvature defined by (v,w), and the pair $(\dot{v_b}, \dot{w_b})$ represents the accelerations of braking. The distance is obtained by the product of the angle γ between the robot and obstacle with the radius r of the curvature:

$$dist(v,w) = \gamma \cdot r \tag{2}$$

The search space is then reduced to the dynamic window. This set only contains the velocities that can be reached within the next time interval. Considering t the time interval where the accelerations ($\dot{v}$ and $\dot{w}$) will be applied and (v_a, w_a) the actual velocity, the dynamic window can be defined as:

$$V_d = \{(v,w) \mid v \in [v_a - \dot{v}t,\ v_a + \dot{v}t] \ \wedge \ w \in [w_a - \dot{w}t,\ w_a + \dot{w}t]\} \tag{3}$$

This dynamic window is set around the center of the actual velocity and the extension of the set depends on the accelerations that are feasible. The curvatures outside the dynamic window can not be reached by the robot "immediately" and therefore are not considered.

The search space used in DWA after the restrictions imposed above can be obtained by

$$V_r = V_s \ \cap V_a \ \cap V_d \tag{4}$$

After having determined the resulting search space from Equation (4), an optimal velocity is selected. In order to achieve this, the objective function is defined as

$$G(v,w) = \sigma\ (\alpha \cdot heading(v,w) + \beta \cdot clearance(v,w) + \gamma \cdot velocity(v,w)) \tag{5}$$

and the pair (v,w), that achieves the maximum value of $G(v,w)$ is chosen as the optimal. The objective function takes three main criteria into account: the heading, clearance and velocity. We now take a deeper look into these criteria.

Target Heading: This criterion is a measure of progress towards the goal location (maximal if the robot moves directly towards the target). The target $heading(v,w)$ measures the alignment of the robot with the target direction. It is obtained by the angle of the target point relative to the robot's heading direction. Due to the change of this direction with different velocities, the target heading θ is computed for a predicted position as shown below:

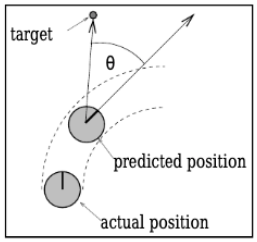

Fig. 1: Target heading [6]

The predicted position is determined by assuming that the robot moves with the selected velocity in the next time interval and considering the kinematic dynamics, the target heading is computed at the position that is reachable by the robot when a maximal deceleration is performed after the next interval. [6] This results in a smooth turning to the target when the robot avoids an obstacle.

Clearance: $Clearance(v,w)$ is the distance to the closest obstacle on the trajectory, or in other words, the closest obstacle that intersects with the curvature. The value of clearance is set to a large constant if no obstacle is present.

Velocity: The $velocity(v,w)$ function has the role of evaluating the progress of the robot in the trajectory. It can be described as the forward velocity of the robot and is simply a projection on the translational velocity v.

The values of all three criteria are then normalized to $[0,1]$ and the objective function is computed. Depending on the values of α, β and γ of 5 these criteria can have different

influences on the objective, but all of them are equally crucial. If only the clearance and the velocity were maximized, the robot would travel into free space but not towards a goal. On the other hand if heading was the only criteria taking into account, the robot would get stopped by the first encountered obstacle and would not be able to move away from it. In the last years, some improvements to this local planner have been suggested. More recently, and worth mentioning, in [22] some problems that arise from the use of DWA are mentioned. The main issues are the ones that come when dense objects appear on the robot's global planning path. When this is the case, the robot is easily trapped. The robot is incapable to judge the density of the environment and therefore needs more turning, acceleration and deceleration operations to get out of trouble. This lack of perception in the density of objects, the robot will not surround the dense area but enter it. The native DWA explained above, only considers factors such as heading angle, target distance and speed. It does not consider the density of objects in the environment, which will inevitably lead to the robot being easily trapped and therefore, very time-consuming. Considering this, the papers suggests an improvement by adding a density evaluation factor to the evaluation function. With this feature it is possible to evaluate if a certain obstacle is in the dense area and solve the problem that the local trajectory planning algorithm DWA can not avoid dense objects in advance. The density factor is responsible to predict the dense area based on the update of the information retrieved by the sensor, pre-perceive the distribution of dense objects and reduce the number of times the robot falls into dense areas. According to the results shown in [22], this improvement results in a more efficient DWA and can be widely used in robots that perform their tasks in dense environments.

B. Pure Pursuit

Pure Pursuit is a well-known planner that follows a given path, thus it is usually denominated as a geometric path tracking method. The Pure Pursuit algorithm calculates the angular velocity to reach a target point using the current position, linear velocity of a robot and the curvature passing the target point.[10] One of the first detailed descriptions of the implementation of the Pure Pursuit algorithm was described in [17].

Geometry Deviation: The Pure Pursuit approach enables the determination of the curvature that will drive the robot to a goal point.

After retrieving the initial position, a search for the target waypoint or goal point as previously mentioned, is performed. The target waypoint is a point that lies on the path and on a given search circle. The radius of this circle is set as a parameter, the lookahead distance. After this, an arc that joins the current point and the goal point is constructed and the chord length of this arc is the lookahead distance. This acts as constraint so that it is possible to determine a unique arc that joins the two points.

In the Figure below we can see the point (x,y), which is one lookahead distance, l from the origin. The point (x,y) is constrained to be on the path. The objective is to calculate the curvature of the arc that joins the origin to (x,y) and whose chord length is l.

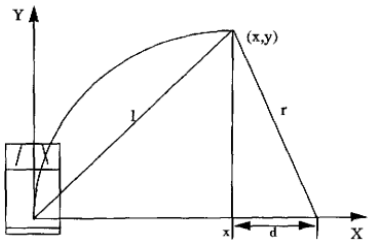

Fig. 2: Illustration of the Pure Pursuit algorithm [17]

The next step is the calculation of the curvature of the circle passing the vehicle position and the target waypoint. Considering the Figure (2), the curvature k is obtained by

$$\begin{aligned} x+d &= r \\ x^2+y^2 &= l^2 \end{aligned} \tag{6}$$

The first equation from the ones above describes the circle of radius l. This area includes the possible locations of goal points. The second one states that the radius of the arc and the x offset are independent and differ by d.

From above and with the goal of relating the curvature of the arc to the lookahead distance, the following equations hold

$$\begin{aligned} d &= r-x \\ y^2+(r-x)^2 &= r^2 \\ x^2+r^2-2rx+y^2 &= r^2 \\ l^2 &= 2rx \\ r &= \frac{l^2}{2x} \\ k &= \frac{2x}{l^2} \end{aligned} \tag{7}$$

Implementation: The pure pursuit algorithm is responsible to produce the steering angle required to bring the vehicle back to the reference path. The classic frame that describes this algorithm can be seen below:

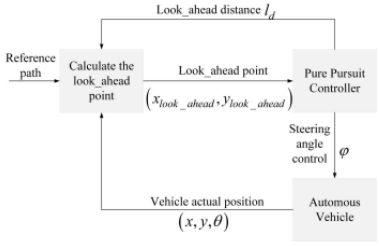

Fig. 3: Pure Pursuit algorithm frame [19]

The first step of the algorithm is to determine the current location that consists of a two dimensional position and the

orientation. After retrieving this, the next step is to find the path point closest to the vehicle. As mentioned before, the goal point should be within one lookahead distance of the vehicle. However it is possible that multiple points fall in this category. If this is the case the vehicle should steer towards the closest point one lookahead distance from its current location which commences up the path.

The next step is to find the goal point. This occurs by moving up the path and calculating the distance between that path point and the vehicle's current location. The moment this goal point has been found, it has to be transformed to the vehicle's local coordinates since path point locations are recorded in the global frame. By using the Equation 6, the curvature is calculated and it is transformed into steering wheel angle by the controller. At last, the position of the vehicle is updated.

To better understand how the steering angle and the rest of the variables are obtained Figure 4 shows a schematic representation of the algorithm:

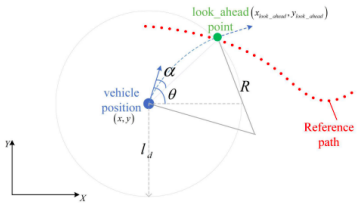

Fig. 4: Pure Pursuit algorithm schematic [19]

First, the angle α is obtained:

$$\alpha = \left| \theta - \arctan\left(\frac{y_{lookahead} - y}{x_{lookahead} - x} \right) \right| \tag{8}$$

The angle α is the angle between the body of the robot and the line that connects the lookahead point and the rear wheel position of the robot. In the above expression, $x_{lookahead}$ and $y_{lookahead}$ describe the position of the lookahead point corresponding to the robot position, x and y describe the position of the robot and θ is the heading of the vehicle.

Another variable that can be retrieved with geometric manipulation is the radius of curvature R that the vehicle needs to follow:

$$R = \frac{l_d}{2\sin\alpha}, \tag{9}$$

where l_d is the lookahead distance.

By using an Ackermann geometric model for the robot, the steering angle can be obtained by:

$$\phi = \arctan\left(\frac{2L\sin\alpha}{l_d} \right), \tag{10}$$

where L is the wheelbase of the vehicle.

III. State of the Art

Benchmarking sampling-based planners is non-trivial for several reasons. First, since the planners rely on sampling, the performance can not be evaluated from a single run. Instead, benchmarks need to be run multiple times to obtain a distribution of the performance metric of interest in that case. Besides this, the optimization for different metrics depends on the application and there is no universal metric to assess the performance of local planner algorithms across all benchmarks. [15] Despite this, some general infrastructures for comparing different planning algorithms have been proposed in order to overcome the problem of scarce literature in this area [4]. Here we single out some that motivated the methods here proposed.

As one of the first references to benchmarking in robotics, [11] introduces a method for measuring odometry errors of mobile robots, focusing on a quantitative evaluation of systematic and non systematic errors. In [18] an early benchmark for robot navigation is presented. However, the way to evaluate the robot performance in a changing environment is not taken into account.

Considering a static environment, an evaluation of two specific robot tasks, namely path planning and obstacle avoidance, is done in [21].

One of the most used approaches consists of the evaluation of performance taking the results within the same environment into consideration - namely in the same moment with the same constraints. [3] This is the case of robotic competitions. Competitions, such as the DARPA Grand Challenges and RoboCup can be considered as an option. Inside the RoboCup initiative, we find RoboCup@Home interesting to mention, that defines a live competition of service robots that need to fulfill a series of tests in a domestic environment. [20] Instead of defining a final challenge, this competition is defined as a benchmarking process that guides the system development from simple tasks towards successively more complex tasks in realistic environments. [20]

Competitions like the ones mentioned above provide an opportunity for researchers to benchmark different approaches and understand the relative advantages of each one, considering a defined measure of success. However, competitions lack the possibility to repeat an experiment with different methods. [23]

With the aim of developing competitions that come close to scientific experiments while providing an objective performance evaluation of robot systems, the RoCKIn project is born. [8] This new approach aims to tackle the issue that competitions provide benchmarking at the system-level based on a single high-level measure by moving to more sophisticated benchmarking activities, while retaining some of the traditional values of competitions as producing a rank among alternative solutions at competition time, assigning prizes and awards to the best teams and push for progress. By using the results of the first RoCKIn competition, the authors were able to show how this project can provide a set of tools to enable the replicability of experiments involving autonomous robots.

In the last few years, with the awareness of how important benchmarking is to objectively compare different solutions, other approaches have been proposed. One example can be

the one proposed in [9], it addresses the problem of evaluating automated guided vehicles with different degrees of autonomy by defining benchmark tools to grade the performance of each approach.

The benchmark methodology described in [9] includes the definition of different types of scenarios, including wide and narrow corridors, small and large isles, zig-zag maneuvering and scenarios with moving objects (other vehicles or people). For each scenario described above, a set of pairs with start or goal positions are also defined. Another relevant defined set is the one containing the maximum speeds, that the vehicle can move when the driver fully accelerates. At last, random tests are generated with combinations of start/goal positions and maximum speeds. The number of tests for each algorithm is given by:

$$N = \frac{n!\ m}{2!(n-2)!} \tag{11}$$

where n is the dimension of the set of pairs start/goal positions and m is the dimension of the set of maximum speeds. [9]

Crucial aspects like robustness, efficiency, safety and comfortability are also evaluated. Robustness is measured by the number of experiments in which the vehicle reaches the goal without any collision, efficiency as a measure of mean time, velocity or mean path length. Safety is measured in terms of average minimum distance to obstacles and risk of collisions, comfortability in terms of the type of the described trajectory and variations of acceleration (jerk).

More recently, in [12] a complete and principled evaluation framework for comparing the performance of local planners is established. To achieve this, multiple simulation scenarios were developed and different types of evaluation metrics were proposed.

Indoor scenarios were used to make an overall evaluation of the local planners. This category includes large-scale-office- and small-scale-family-house-like scenarios. Narrow space scenarios (like U-shaped or Z-shaped corridors) were designed to challenge the flexibility and smoothness of the local planners. In the previous categories the complete prior map is given as the input to local planners. To test the adaptability, the authors blurred the maps individually and input the incomplete map to the local planners.

At last, the robustness of local planners is challenged in dynamic scenarios that include simulations where only two people are walking, but also a more complex one where six people are walking around in an open space environment. In this last one, the robot needs to pass through the crowd to reach the goal at the other, reproducing a crowded scene in the real world. This type of scenarios challenges the safety, flexibility and real-time performance of local planners.

The evaluation of the performance of the local planner is done from different aspects by using different metrics.

Safety metrics are used to evaluate the security performance of the local planner. These metrics can be evaluated in terms of minimum distance to the closest obstacle and percentage of time spent by the robot in the surrounding area of the obstacles.

In [12], d_o is defined as the minimum distance to the closest obstacle and p_o as the percentage of time spent by the robot in the area around obstacles, and the safety metric is expressed as:

$$d_o = min\{d_i\}, 1 \leq i \leq n \tag{12}$$

$$p_o = \frac{\sum(t_b - t_a)}{t_n - t_1} \times 100\% \tag{13}$$

where a and b represent the indices of the timestamps that satisfy $d_k \leq d_{safe}, a \leq k \leq b$, where d_k is the distance to the closest obstacle for $i = k$ and d_{safe} is the preset safe distance to obstacles.

Another type of metrics is presented: efficiency metrics. These are used to evaluate motion efficiency, described by how quickly the local planner guides the robot to the local, and computational efficiency, which evaluates the real-time performance of local planners.

The motion efficiency uses the total travel time, T:

$$T = t_n - t_1 \tag{14}$$

And the computational efficiency is given by the average time consumption of a single local planner iteration:

$$C = \frac{1}{n}\sum_{i=1}^{n} c_i \tag{15}$$

Lastly, [12] also refers to smoothness metrics that are used to assess the quality of motion commands given by the local planners. The overall smoothness performance is evaluated by the path and velocity smoothness. The path smoothness is defined as:

$$fps = \sum_{i=2}^{N-1} ||\Delta x_{i+1} - \Delta x_i||^2 \tag{16}$$

where $\Delta x_i = x_i - x_{i-1}$ represents the displacement vector at $x_i = (x_i, y_i)^T$. The velocity smoothness is measured by computing the average of acceleration:

$$fvs = \frac{1}{n-1}\sum_{i=1}^{n-1} \left| \frac{v_{i+1} - v_i}{t_{i+1} - t_i} \right| \tag{17}$$

IV. Implementation

To comprehensively evaluate the performance of local planners, different scenarios were simulated to recreate different types of rooms in where a robot can be used and can be seen below:

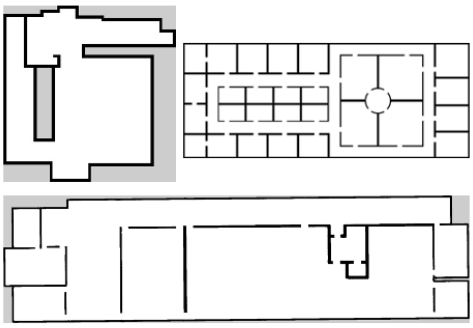

Fig. 5: Top left: **Airlab**; Top right: **Office_b**; Bottom: **7A-2**. Figures not to scale.

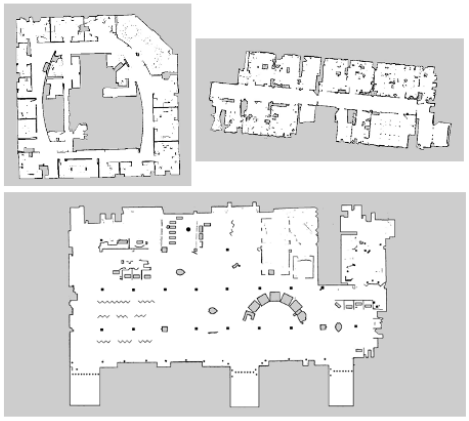

Fig. 6: Scenarios used in the experiments. Top left: **Intel**; Top right: **Fr079**; Bottom: **Mexico**. Figures not to scale.

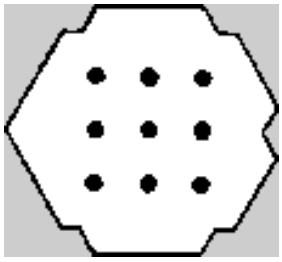

Fig. 7: Scenario used in the experiments: **Turtlebot_world**. Figure not to scale.

We choose Gazebo as the simulation platform since its high popularity among the ROS community. In the evaluation, we focus on the local planning problem of differential drive mobile robots. We refer to the popular model Turtlebot3 Waffle and design a robot model in Gazebo through the XML macros language Xacro. The footprint of the robot is set to a circle with a radius of 0.22m. On this basis, a lidar sensor is mounted on the robot. The sensor has a maximum and minimum range of $3.5m$ and $0.12m$, respectively, and an update rate of $5Hz$.

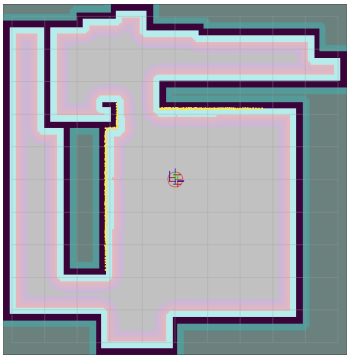

Fig. 8: Simulator view with a simulated scenario and the simulated robot model

We select multiple sets of different goal poses to test local planners in each scenarios. In addition to this, we obtain the robot pose from the ground truth provided by Gazebo to avoid the influence of localization.

The local planners used are the ones mentioned in II and their implementation can be consulted in [1]. It is important to note that the algorithms presented have different names and are called DWB for the Dynamic Window Approach one and Regulated Pure Pursuit for the implementation of Pure Pursuit. The last one implements additional regulation terms on collision and linear speed.

Metrics serve as the evaluation needed to assess the performance of our benchmarking objects. In our case, we use metrics as tools to evaluate the performance of the local planners. In order to do this, different metrics were chosen. The first metric used to assess the performance was the number of times that the robot reaches the target pose and in which scenarios this happens, more or less frequently. The execution time was used as a metric and can be described as the time that goes from the moment when the target pose is accepted, to the time where is actually reached or not, like shown below:

$$T = t_n - t_0, \tag{18}$$

where t_n represents the timestamp of the call in which the robot reaches the target pose (or not) and t_0 represents the timestamp of the call in which the target pose is accepted.

Having this said, we used this metric in different situations, the first one being the total execution time independently of the fact that the target pose is reached or not and then only for the situations where the target pose is reached. And lastly for the situations where the target pose is not reached.

Besides the execution time, another metric that was used to evaluate the performance was the path length. This is done by splitting the path into small segments and considering the general length of one segment. Since dx and dy are infinitesimally small, the length dL can be approximated using the Euclidean Distance. By summing all these consecutive segments of the curve, we get the integral:

$$L = \int_{t_0}^{t_n} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt \tag{19}$$

Like mentioned before for the execution time, this metric was also assessed in total and in two different cases, one when the target pose is reached and one when it is not.

V. Experimental Results

As previously mentioned, for this benchmark we used different local planners, different scenarios and metrics were chosen to evaluate the performance. The main goal of this work is to benchmark local planners and having this in mind two different local planners were chosen. The use of different metrics and scenarios is key to assure a fair evaluation.

We now present the results obtained by using DWB and RPP. To obtain these results, 1000 runs were done.

A. DWB

In this section the results regarding DWB are presented. First, we start with the results concerning the percentage of times that the target pose was reached:

TABLE I: Evaluation of DWB by percentage of times that the target pose is reached

Target Pose Reached	Percentage of Times
Yes	91.83
No	8.17

Considering the same metrics but taking the scenarios used in the runs into consideration, the following results are obtained:

TABLE II: Evaluation of DWB by percentage of times that the target pose is reached in different scenarios

Scenario	Percentage of Times Target Pose is reached
7A-2	95.83
Turtlebot_world	100
Airlab	100
Intel	94.20
Office_b	100
Mexico	55.07
Fr079	95.89

In the next table the planner is evaluated in terms of average execution time and respectively standard deviation:

TABLE III: Evaluation of DWB by average execution time and standard deviation

	Average Execution Time	Standard Deviation
Total	61.545	55.374
If Target Pose is Reached	63.206	47.403
If Target Pose is Not Reached	42.86	109.074

Now the planner is evaluated in terms of average path length and respectively standard deviation. The results are the following:

TABLE IV: Evaluation of DWB by average path length and standard deviation

	Average Path Length	Standard Deviation
Total	14.19	12.648
If Target Pose is Reached	14.845	12.016
If Target Pose is Not Reached	6.827	16.612

Having these results, we can conclude that the planner was successful in the vast majority of times with exceptions in bigger environments. However, it is also interesting to note that when the target pose is not reached, the average execution time and the average path length is considerably smaller which is a positive takeout since it means that when the path is not feasible the amount of wasted time is small.

B. Regulated Pure Pursuit

In this section the results regarding RPP are presented. Firstly, the results concerning the percentage of times that the target pose was reached are presented:

TABLE V: Evaluation of RPP by percentage of times that the target pose is reached

Target Pose Reached	Percentage of Times
Yes	90.51
No	9.49

Considering the same metrics but taking into consideration the scenarios used in the runs the following results are obtained:

TABLE VI: Evaluation of RPP by percentage of times that the target pose is reached in different scenarios

Scenario	Percentage of Times Target Pose is reached
7A-2	96
Turtlebot_world	100
Airlab	100
Intel	93.06
Office_b	95.83
Mexico	58.33
Fr079	91.04

In the next table the planner is evaluated in terms of average execution time and standard deviation respectively:

TABLE VII: Evaluation of RPP by average execution time and standard deviation

	Average Execution Time	Standard Deviation
Total	36.967	63.117
If Target Pose is Reached	35.69	25.434
If Target Pose is Not Reached	49.135	188.751

Now, the planner is evaluated in terms of average path length and standard deviation respectively. The results are the following:

TABLE VIII: Evaluation of RPP by average path length and standard deviation

	Average Path Length	Standard Deviation
Total	15.658	20.242
If Target Pose is Reached	15.897	12.657
If Target Pose is Not Reached	13.372	57.748

With these results, we can conclude that the planner was overall successful in reaching the target pose despite exceptions in the bigger environments with obstacles. However, in this case the average execution time when the target pose is not reached is larger than when it is reached which can be a disadvantage and can be seen as a waste of time. Regarding the average path length, the results for the total average path length and when the target pose is reached are similar to the ones obtained with DWB. However, when the target pose is not reached the average path length when RPP is used is considerably larger than when using DWB.

C. Comparison Between Planners

When comparing local planners, execution time and path length can be highly important. In a real life situation robots can be required to perform tasks where they need to be fast and also reach the target pose by following a short path. Having this in mind, we compared the behavior of both local planners in terms of average execution time and average path length in a general way, then considering only when the target pose is reached and lastly, when the target pose is not reached.

To better understand the difference in the performance of the local planners, the results presented in V-A and V-B are combined:

TABLE IX: Evaluation of DWB and RPP by percentage of times that the target pose is reached in different scenarios

Scenario	**Percentage of Times Target Pose is reached**	
	DWB	RPP
7A-2	95.83	96
Turtlebot_world	100	100
Airlab	100	100
Intel	94.20	93.06
Office_b	100	95.83
Mexico	55.07	58.33
Fr079	95.89	91.04

TABLE X: Evaluation of DWB and RPP by average execution time and standard deviation

	Average Execution Time		**Standard Deviation**	
	DWB	RPP	DWB	RPP
Total	61.545	36.967	55.374	63.117
If Target Pose is Reached	63.206	35.69	47.403	25.434
If Target Pose is Not Reached	42.86	49.135	109.074	188.751

TABLE XI: Evaluation of DWB and RPP by average path length and standard deviation

	Average Path Length		**Standard Deviation**	
	DWB	RPP	DWB	RPP
Total	14.19	15.658	12.648	20.242
If Target Pose is Reached	14.845	15.897	12.016	12.657
If Target Pose is Not Reached	6.827	13.372	16.612	57.748

With these results, we can see that the percentage of times that the target pose is reached is the same for both planners in small environments such as Airlab and Turtlebot_world. The results obtained for the other scenarios are also similar, it is worth mentioning the difference for the Office_b - an office-like room, where DWB reaches the target pose every time and RPP only reaches it in 95.83% of the times. Regarding average execution time, RPP, in average, needs a smaller amount of time to reach the target pose which can be seen as an advantage. However, the execution time when the target pose is not reached is bigger than when it is reached. This can be problematic since the user will need more time until figuring out that the path is not feasible. Regarding average path length, the results are very similar for the measurements when the target pose is reached and in total. However and once again, the value is rather different when the target pose is not reached.

Besides the tables above we decided to present the results in the following way:

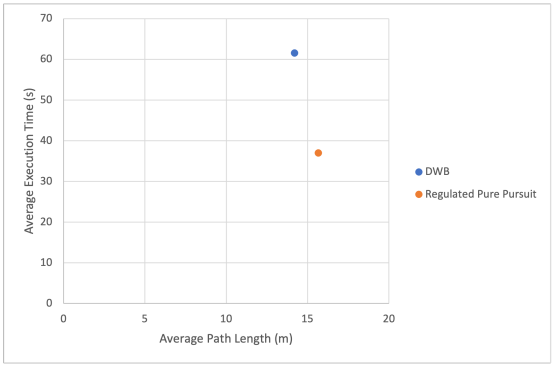

Fig. 9: Total average execution time in function of total average path length for both planners

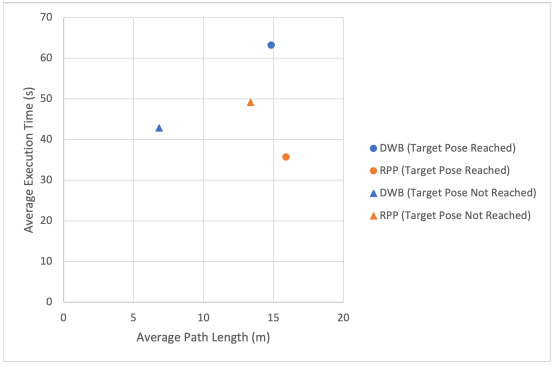

Fig. 10: Average execution time in function of average path length for both planners, when the target pose is reached or not

Analyzing the results above, we can conclude that overall the average path length is slightly bigger when using RPP. The difference in average execution time between the planners is larger. In this case DWB has a much higher average execution time than RPP.

In the case where the target pose is reached, the results are the same as the above. The situation is rather different in the case where the target pose is not reached since RPP has

a higher average execution time and the difference between average path lengths also increases with RPP having the largest one, once again.

After this analysis, we considered that it was important to know the relation between the average execution time and the different scenarios for both planners. The following results were obtained:

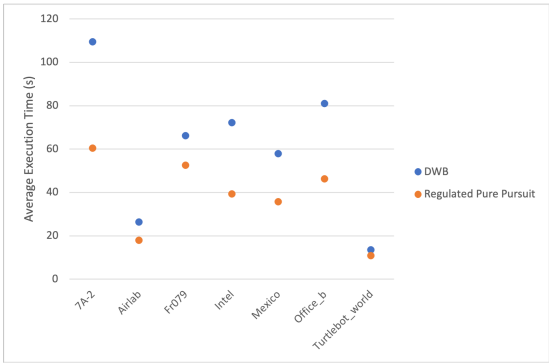

Fig. 11: Relation between average execution time and the different scenarios for DWB

Considering the above, the average execution time is bigger for DWB overall. However, in small and relatively simple scenarios (like Airlab and Turtlebot_world) the results are very similar.

After this, we decided to do the same as previously presented but this time considering average path length. The results are the following:

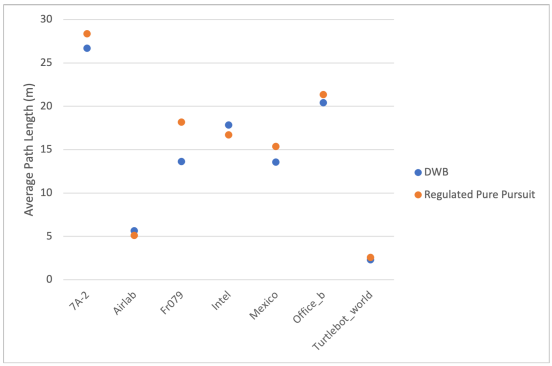

Fig. 12: Relation between average path length and the different scenarios for DWB

The results were very similar in this case with the exception of the use in the larger scenario (Mexico) where the average path length obtained by using RPP was higher than the one achieved by using DWB.

VI. Conclusion

Our goal with this work was - more than creating a method for absolute evaluation of local planners - to help researchers in the robotics field to choose their local planner accordingly to the situation where it is going to be used. Having this said, we tested two local planners suitable for differential drive robots, one of the most commons drive mechanisms, and made simulations in different scenarios with different characteristics from open-space to office-like rooms.

The comparison of the local planners was not possible if tools for evaluation were not used. For this reason, we implemented different metrics: average execution time, average path length and number of times that the pose is reached to be able to help the decision of when to use a certain local planner.

References

[1] (accessed: 30.10.2021). URL: https://navigation.ros.org/.

[2] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, Abhinav Gupta. “PyRobot: An Open-source Robotics Framework for Research and Benchmarking”. In: *Carnegie Mellon University* (2019).

[3] Gianluca Antonelli. “Robotic research: are we applying the scientific method?” In: *Frontiers in Robotics and AI, Volume 2* (2015).

[4] Christoph Sprunk, Jörg Röwekämper, Gershon Parent, Luciano Spinello, Gian Diego Tipaldi, Wolfram Burgard and Mihai Jalobeanu. “An Experimental Protocol for Benchmarking Robotic Indoor Navigation”. In: *Department of Computer Science, University of Freiburg, Germany; Microsoft Robotics, Microsoft Corporation, USA* (2014).

[5] Diego Torricelli, Jose L. Pons. “Benchmarking: The Keystone of Robotics Innovation”. In: (2019).

[6] Dieter Fox, Wolfram Burgard, Sebastian Thrun. “The Dynamic Window Approach to Collision Avoidance”. In: *IEEE Robotics and Automation Magazine* (1997).

[7] Rüdiger Dillmann. “Benchmarks for Robotics Research”. In: *European Robotics Network* (2004).

[8] Francesco Amigoni, Emanuele Bastianelli, Jakob Berghofer, Andrea Bonarini, Giulio Fontana, Nico Hochgeschwender, Luca Iocchi, Gerhard K. Kraetzschmar, Pedro Lima, Matteo Matteucci, Pedro Miraldo, Daniele Nardi and Viola Schiaffonati. “Competitions for Benchmarking Task and Functionality Scoring Complete Performance Assessment”. In: *IEEE Robotics and Automation Magazine* (2015).

[9] Hector Yuste, Leopoldo Armesto and Josep Tornero. “Benchmark Tools for Evaluating AGVs at Industrial Environments”. In: *TIEEE/RSJ International Conference on Intelligent Robots and Systems* (2010).

[10] Hiroki Ohta, Naoki Akai, Eijiro Takeuchi, Shinpei Kato and Masato Edahiro. “Pure Pursuit Revisited: Field Testing of Autonomous Vehicles in Urban Areas”. In: *IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications* (2016).

[11] J. Borenstein and L. Feng. “Umbmark: A benchmark test for measuring odometry errors in mobile robots”. In: *Proc. SPIE, 2591:113–124* (1995).

[12] Jian Wen, Xuebo Zhang, Qingchen Bi, Zhangchao Pan, Yanghe Feng,Jing Yuan, Yongchun Fang. "MRPB 1.0: A Unified Benchmark for the Evaluation of Mobile Robot Local Planning Approaches". In: (2020).

[13] Steven Johnson. *Where Good Ideas Come from: The Natural History of Innovation*. Penguin Books, 2010.

[14] Kamran H. Sedighi, Kaveh Ashenayi, Theodore W. Manikas, Roger L. Wainwright, Heng-Ming Tai. "Autonomous Local Path Planning for a Mobile Robot Using a Genetic Algorithm". In: *Proceedings of the 2004 Congress on Evolutionary Computation* (2004).

[15] Mark Moll, loan A. Sucan, Lydia E. Kavraki. "Benchmarking Motion Planning Algorithms". In: *IEEE Robotics and Automation Magazine* (2015).

[16] Martin Hägele, Kai Pfeiffer. "RoSta Deliverable D 4.1: Report on State of the Art on Benchmarks for Mobile Manipulation and Service Robots". In: *Fraunhofer IPA* (2007).

[17] R. Craig Conlter. "Implementation of the Pure Pursuit Path Tracking Algorithm". In: *The Robotics Institute of Carnegie Mellon University* (1992).

[18] R. Knotts, I. Nourbakhsh and R. Morris. "Navigates: A benchmark for indoor navigation". In: *Int. Conf. and Exp. on Robotics for Challenging Environments* (1998).

[19] Rui Wang, Ying Li, Jiahao Fan, Tan Wang and Xueta Chen. "A Novel Pure Pursuit Algorithm for Autonomous Vehicles Based on Salp Swarm Algorithm and Velocity Controller". In: *IEEE Access, Volume 8* (2020).

[20] Sven Wachsmuth, Dirk Holz, Maja Rudinac, Javier Ruiz-del-Solar. "RoboCup@Home – Benchmarking Domestic Service Robots". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).

[21] W. Nowak, A. Zakharov, S. Blumenthal, and E. Prassler. "Benchmarks for mobile manipulation and robust obstacle avoidance and navigation". In: *BRICS Deliverable D3.1* (2010).

[22] Xiquan Mai et al. "An improved dynamic window approach for local trajectory planning in the environment with dense objects". In: *Journal of Physics: Conference Series 1884 012003* (2021).

[23] Zhi Yan, Luc Fabresse, Jannik Laval, Noury Bouraqadi. "Building a ROS-Based Testbed for Realistic Multi-Robot Simulation: Taking the Exploration as an Example". In: (2017).