Systematic Analysis of Global and Local Planners for Optimal Trajectory Planning

Maximilian Pittner[a], Markus Hiller[a], Florian Particke[a], Lucila Patiño-Studencki[a], and Jörn Thielecke[a]
[a]Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Institute of Information Technology, Erlangen

Abstract

In this paper, common global (Dijkstra's algorithm and A* algorithm) and local planners (Dynamic Window Approach, Elastic Band and Timed Elastic Band) are analyzed and investigated for different scenarios in simulation and in real world. Since path planning algorithms are often used without reasoning and proper comparison to other possible solutions, this paper tries to close this gap. The results of our systematic analysis not only shows the key features of the algorithms but additionally expose intelligent combinations of global and local path planners. Hence, these results give stakeholders a sound decision background on trajectory planning methods for their specific industrial scenario.

1 Introduction

Navigation of autonomous mobile robots forms an important task that has to be mastered. Being part of the navigation, trajectory planning tries to find low-cost collision-free paths to any goal. Many different approaches exist for solving this problem, mostly by dividing it into global (based on a-priori knowledge) and local (updated knowledge) path planning. However, in practice one algorithm is often simply chosen without reasoning about the suitability for the target scenarios. Examples are [1] or [2]. In this paper, we systematically analyze common global and local path planners for different situations. The global planners examined in this contribution are based on Dijkstra's algorithm [3] and the A* algorithm [4]. Regarding the local planners, we focus on three different solutions. We investigate the Dynamic Window Approach (DWA) described in [5], and two local planners that use an elastic band model: the classic Elastic Band (EBAND) [6] and Timed Elastic Band (TEB) [7]. In Section 2, Dijkstra's algorithm and the A* algorithm are shortly described. The local planners DWA, EBAND and TEB are introduced in Section 3. The algorithms are evaluated in Section 4, where the global planners are analyzed in Subsection 4.1 using simulated environments of varying complexity. In Subsection 4.2, the local planners are evaluated on the robotic platform *Turtlebot* [8] for different real world scenarios with static and dynamic obstacles. In Subsection 4.3, proposals for reasonable combinations of global and local planners are given. The results are summarized in Section 5.

2 Global Path Planning

The goal of global path planning is to find the optimal path to any goal using a given map, mostly aiming at a short path length and low computing time required for the path search. Dijkstra's algorithm [3] and the A* algorithm [4] use a discrete search method to find the goal node by utilizing a graph model. The graph consists of nodes, which can be formed by the cells of an underlying grid map. Both algorithms manage to find optimal paths, hence optimal ways through the graph, regarding the costs defined by connections between the nodes. The cost $g(X)$ of reaching the node X is defined by

$$g(X) = g(X_{parent}) + d(X_{parent}, X), \tag{1}$$

where $g(X_{parent})$ is the cost to the predecessor X_{parent} and $d(X_{parent}, X)$ represents the cost for the transition from the predecessor X_{parent} to the node X. The algorithms then iteratively explore cells with lowest cost, until the goal node is reached. In contrast to Dijkstra's algorithm, the A* additionally uses a heuristic function for estimating yet arising costs to the goal. The criterion of choosing the next node is thus given by the function

$$f(X) = g(X) + h(X), \tag{2}$$

where $h(X)$ describes the heuristic function.

After reaching the goal node, the actual geometrical path points have to be determined. This can either be achieved by tracing back the route to the start along the connection of nodes using the underlying grid map, or by using a gradient descent method [9]. The path points are then set by iteratively calculating the gradient on a potential field consisting of the explored cells.

We examine both the impact of different heuristic functions as well both ways of setting path points in Section 4.1.

3 Local Path Planning

Local path planning uses sensor information to adjust and in best case improve given global paths with respect to geometric path length as well as the amount of time required for transition. Besides, local path planning must pay attention to changes in the environment not enclosed in the global map. We investigate the three most popular local

ISBN 978-3-8007-4699-6

planners that use varied approaches: DWA [5], EBAND [6] and TEB [7].

The aim of DWA is to determine a velocity command for the next control interval provoking a low-cost trajectory. The set of velocity commands is restricted by the so called *dynamic window*, which only includes velocities that can be reached within the next time step considering dynamic and kinematic constraints of the robot platform. The velocities are weighted by an objective, scoring the resulting trajectory. The velocity command with highest score is chosen and applied for the next control interval.

We compare this planner to the EBAND and the TEB. The EBAND models the planing task using a set of configurations forming states of a virtual band. The trajectory optimization is performed by applying artificial forces to this band. A contracting force is used to smooth and shorten the band, while a repulsive force stretches the band in case of approaching obstacles.

For the TEB, time intervals for the transition between two consecutive configurations are added to the set of states. Those states are affected by applying different kinds of cost functions, considering geometrical (e.g. clearance to obstacles) and temporal constraints (e.g. amount of required time for trajectory transition). All cost functions form edges in a hypergraph connecting nodes provided by the states of the TEB. The best trajectory is then found by optimizing the hypergraph. The properties of these different approaches lead to varying motion behaviors depending on the given scenario, and are analyzed in Section 4.2.

4 Evaluation

In this section, the global planners are analyzed in Subsection 4.1 for simulated scenarios with varying degrees of complexity. The local planners are investigated in Subsection 4.2 for real world scenarios considering collision avoidance for both static and dynamic obstacles. For evaluation of local planners we use the robot platform Turtlebot [8]. In Subsection 4.3, proposals for reasonable combinations of global and local planners are provided.

4.1 Evaluation of Global Planners

We compare two global planners, one based on Dijkstra's algorithm and the other on the A* algorithm. We further investigate variations of the A* using different heuristic functions, in particular Manhattan and euclidean distance.

For the first scenario, we consider a simple navigation task, where the goal is located at the end of a straight hallway without any obstacles. As indicated in Table 1, our results show that while both algorithms achieve an optimal path length, the A* variations require a significantly smaller amount of computing time. The reason is the notably smaller number of cells being explored by the algorithms using a heuristic function. Especially for simple scenarios, heuristic functions estimate the following costs quite well, since the real path proceeds straight, similar to the predicted path. Dijkstra's algorithm however explores a large set of cells in an undirected, circular fashion.

As second scenario, we consider a more complex task,

Algorithm	Path length	Computing time
Dijkstra	$14,35\,m$	$14,45\,\mu s$
A* (Manhattan)	$14,35\,m$	$0,15\,\mu s$
A* (euclidean)	$14,35\,m$	$0,29\,\mu s$

Table 1 Comparison of criteria of success of different global planners for a simple planning scenario.

where the robot has to navigate through a maze-like environment. The goal is located such that the path includes curves and detours. The comparison of measured computing time (cf. Table 2) shows that the usage of the heuristic function in general is less beneficial for more complex scenarios. The reason is that the heuristic function sometimes estimates paths leading into deadlocks. Furthermore, there is a notable difference of computing time between the two A* planners. The A* that uses the Manhattan distance requires only about 12 % of the calculation time of A* with euclidean distance. This is due to the fact that the Manhattan distance represents a better estimation for the real costs according to the underlying graph model. For the model used in the investigated approach, one cell (node) is only connected to four nodes, in particular its vertical and horizontal neighbor cells. Thus, the actual cost from one node to the goal node can be defined by the Manhattan distance between these nodes, which results in the A* with Manhattan distance estimating costs more precisely than the A* with euclidean distance.

Algorithm	Computing time
Dijkstra	$70,63\,\mu s$
A* (Manhattan)	$4,69\,\mu s$
A* (euclidean)	$37,65\,\mu s$

Table 2 Comparison of criteria of success of different global planners for a complex planning scenario.

After the goal node has been found, the path is usually determined by tracing back across the explored cells. To avoid a stairs-like path consisting of horizontal and vertical line segments, also diagonal connections are considered. However, natural trajectories do not consist of horizontal, vertical and diagonal line segments. To receive a smooth trajectory, the gradient descent method described in Section 2 for determining the path points can be used. Figure 1

Figure 1 Paths determined with grid path (blue) and gradient descent method (red) for the second scenario.

shows the paths for using the grid path and the gradient descent method for the second scenario. Both algorithms use Dijkstra's algorithm for search. The path retrieved by the gradient descent method is not limited to the grid structure and therefore results in a smooth and short trajectory (cf. Table 3).

Method of setting path points	Path length
Grid path	$70,63\,m$
Gradient descent	$63,21\,m$

Table 3 Comparison of path length for the two methods of setting path points for the second scenario.

Based on the previous results, combining A* utilizing a precisely estimating heuristic for search with a gradient descent method for setting path points might sound promising. However, our results show that this combination does not lead to better performance. The reason is that the gradient descent method requires a large set of explored cells to provide a potential field of sufficient size for which a gradient calculation is sensible. Dijkstra's algorithm provides a gapless, circular potential field. In contrast, an efficient A* heuristic reduces the set of explored cells as much as possible. As these requirements of minimal exploration and sufficient size of the potential field are somehow contradicting, we have to choose between the planners A* algorithm with grid path method or Dijkstra's algorithm with gradient descent method. The decision depends on whether few computing time or short path length is more important. To avoid such a choice and improve the algorithm on both cost criteria, we suggest a solution in Section 4.3.

4.2 Evaluation of Local Planners

For several scenarios, DWA, EBAND and TEB are investigated on their ability to further adjust and optimize paths provided by the global path planning algorithms. We demonstrate the varying behaviors of path adjustment for a scenario without obstacles. Furthermore, we examine the algorithms on their ability of avoiding spontaneous static obstacles, as well as dynamic obstacles crossing the robot's initially planned (global) trajectory. We purposefully consider scenarios where the variance of motion behavior due to the functional differences is obvious. For a scenario without obstacles (cf. Figure 2), the robot's start pose is located in opposite direction of the goal. Here, the EBAND algorithm chooses a trajectory of shortest geometric path length. The reason is that the EBAND configurations are initially located on the global path and are not being changed, since there are no artificial forces applied on the band. Thus, the robot simply tries to pass the configurations by first rotating for 180° and then driving linearly towards the goal. The TEB algorithm prefers a trajectory of higher path length. The determined movement consists of a right curve driven backwards followed by a left curve driven forwards. This behavior is due to the algorithm's ability of taking into account time intervals required for the transition between consecutive configurations. In contrast to the other solutions, the algorithm considers the dynamic constraints of both maximal translational and rotational velocity and then reasons that the amount of time needed for an in-place rotation followed by a translational movement exceeds the time needed for the backwards curve followed by the forward curve. Thus, TEB finds a cheaper trajectory in terms of transition time than EBAND. By using DWA, a suboptimal trajectory results with respect to both spatial and temporal dimension, because the objective weighting the velocity commands prefers higher positive translational velocities. Consequently due to the starting pose, the robot first heads into the wrong direction.

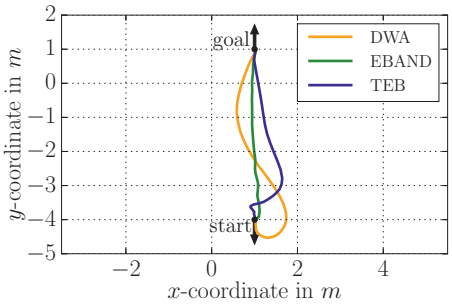

Figure 2 Trajectories for a scenario without obstacles with start configuration headed in opposite direction as the goal configuration.

In case of static obstacles spontaneously appearing directly on the global path (cf. Figure 3), the results show that the EBAND algorithm does not find an admissible path, while DWA and TEB succeed in circumnavigating the obstacle. The reason for the failure of the EBAND algorithm is due to with the definition of the repulsive force. The force on a specific configuration is directed in the opposite direction of the motion of the obstacle nearby the configuration. Since the observed obstacle is located in between two consecutive configurations, applying a repulsive force only pushes the configurations away from the obstacle. In this case the configurations are only moved tangential to the band, which does not lead to a circumnavigation. DWA instead directly calculates velocity commands. The objective incorporates a factor preferring velocities that result in large clearance to obstacles, which initiates the circumnavigation as soon as the obstacle is perceived. The TEB on the other hand adds new configurations by minimizing the cost function penalizing distance to obstacles. Thus, a way around the obstacle is constructed and circumnavigation is achieved.

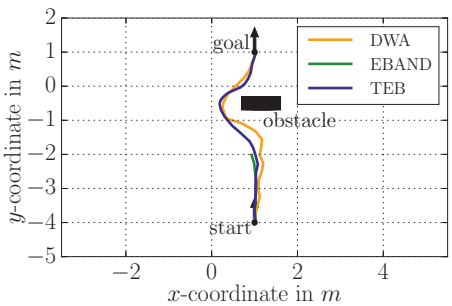

Figure 3 Trajectories for a scenario with a static obstacle.

ISBN 978-3-8007-4699-6

For the last scenario, we consider a dynamic obstacle crossing the global path of the robot as can be seen in Figure 4. In this case the algorithms based on band models, in particular EBAND and TEB, manage to avoid the obstacle, whereas using DWA leads to a collision. The failure of the DWA is due to the fact that no geometrical paths are determined by the planner, in strict sense. Since only velocity commands are considered, a straight motion is applied as long as the planner does not predict a collision for the next time step. Only at the time the obstacle is located right on the robot's path, the algorithm applies high costs to forward velocities. However, initiating an avoidance maneuver is too late at that time. In contrast, planning with geometrical bands pushes away configurations, which were set beforehand, as soon as the obstacle gets close up to the affected configurations.

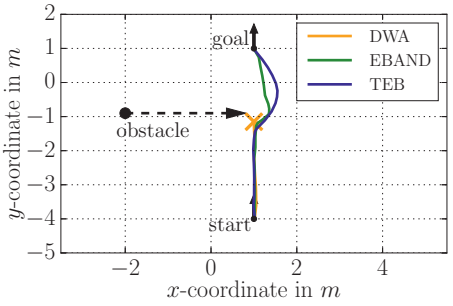

Figure 4 Trajectories for a scenario with a dynamic obstacle.

4.3 Combining global and local planners

At last, a suggestion for efficiently combining global and local planners is presented. As our investigations in Section 4.1 showed, we can choose between A* (for less computing time) and Dijkstra with gradient descent method (for shorter and smoother paths). Dijkstra's algorithm sometimes requires much more computing time, while applying the gradient descent method instead of grid path method enlarges computing time additionally. We therefore suggest to use A* algorithm with a precisely estimating heuristic function. The problem of path length can be solved by applying a local planner which enables a targeted trajectory smoothing. In Section 4.2 we demonstrated that EBAND and TEB succeed in finding short local trajectories. An example for a successful global path adjustment that results in a shorter trajectory is shown in Figure 5. Here, the contracting force discards slacks and corners in the path which usually come with applying grid path method. For TEB, the adjustment improves the trajectory with focus on shorter transition time instead of geometrically short paths.

5 Conclusion and Outlook

In this paper, a systematic analysis of common global and local planners was presented with the aim of reasoning about the applicability of the algorithms for different scenarios. For the global planners, we showed that A* algorithm with a precisely estimating heuristic reduces computing time, while Dijkstra's algorithm results in a shorter path length when being combined with the gradient descent method. Our suggestion for combining global and local planners revealed an opportunity to accomplish short trajectories with small computing time. The evaluation of local planners showed that the EBAND leads to geometrically short paths, whereas TEB performs better in achieving shorter transition times. We demonstrated that the EBAND algorithm fails on finding local paths in special cases of static obstacles, whereas the DWA has problems concerning avoidance of dynamic obstacles. According to our tests, TEB showed best results among the evaluated algorithms. To summarize, none of the investigated planning algorithms represents a holistic solution covering all situations. Depending on the given scenario, our analysis and proposals should facilitate the choice of path planning algorithm for stakeholders.

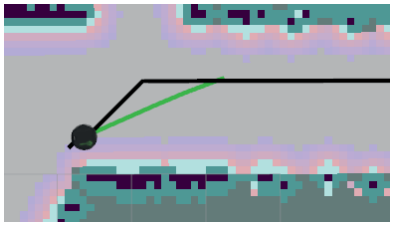

Figure 5 Global path set with grid path method (black) adjusted by applying EBAND (green).

6 Literature

[1] Brock, O.; Khatib, O.: High-speed navigation using the global dynamic window approach: Robotics and Automation: 1999

[2] Keller, M.; Hoffmann, F.; Hass, C.; Bertram, T.; Seewald, A.: Planning of optimal collision avoidance trajectories with timed elastic bands: IFAC Proceedings Volumes: 2014

[3] Dijkstra, E. W.: A note on two problems in connexion with graphs: Numerische Mathematik: 1959

[4] Hart, P. E.; Nilsson, N. J.; Raphael, B.,: A formal basis for the heuristic determination of minimum cost paths: IEEE transactions on Systems Science and Cybernetics: 1968

[5] Fox, D.; Burgard, W.; Thrun, S.: The dynamic window approach to collision avoidance: IEEE Robotics & Automation Magazine: 1997

[6] Quinlan, S.; Khatib, O.: Elastic bands: Connecting path planning and control: Robotics and Automation: 1993

[7] Rösmann, C.; Feiten, W.; Wösch, T.; Hoffmann, F.; Bertram, T.: Trajectory modification considering dynamic constraints of autonomous robots: Robotics: Proceedings of ROBOTIK 2012

[8] O. S. R. Foundation: Turtlebot.com - turtlebot 2: http://www.turtlebot.com/turtlebot2/: 02.11.2017

[9] Amato, N.: Randomized motion planning - potential field methods: University of Padua: https://parasol.tamu.edu/∼amato/Courses/padova04/: 2004

ISBN 978-3-8007-4699-6