Predicting Travel Time from Path Characteristics for Wheeled Robot Navigation

Peter Regier Marcell Missura Maren Bennewitz

***Abstract*— Modern approaches to mobile robot navigation typically employ a two-tiered system where first a geometric path is computed in a potentially obstacle-laden environment, and then a reactive motion controller with obstacle-avoidance capabilities is used to follow this path to the goal. However, when multiple path candidates are present, the shortest path is not always the best choice as it may lead through narrow gaps and it may be in general hard to follow due to a lack of smoothness. The assessment of an estimated completion time is a much stronger selection criterion, but due to the lack of a dynamic model in the path computation phase the completion time is typically a priori not known. We introduce a novel approach to estimate the completion time of a path based on simple, readily available features such as the length, the smoothness, and the clearance of the path. To this end, we apply non-linear regression and train an estimator with data gained from the simulation of the actual path execution with a controller that is based on the well-known Dynamic Window Approach. As we show in the experiments, our method is able to realistically estimate the completion time for 2D grid paths using the learned predictor and highly outperforms a prediction that is only based on path length.**

I. Introduction

A fundamental ability of a mobile robot is collision-free navigation in the environment. Many state-of-the-art navigation systems employ a two-stage approach to realize this in an efficient manner. Here, the first stage is dedicated to plan a spatial global path through the environment from the position of the robot to the goal location. Such global path planning is commonly carried out using a grid-based A* planner [1]. To smoothly follow the globally computed 2D path afterwards, one typically employs a local reactive collision avoidance system that efficiently generates velocity commands for the robot [2], [3], [4].

Robots nowadays are facing the challenge of having to solve tasks with ever increasing complexity. In several real world applications, the predictability of the completion time of such tasks plays an important role. In particular in multi-robot scenarios where the actions of multiple robots need to be accurately coordinated, the prediction of the task completion time is pivotal for time-efficient planning. Such scenarios include cooperative floor-cleaning and household tasks, long term autonomous driving systems [5] that need to estimate their travel time for a more efficient power supply management, and museum tour-guide robots that have to schedule their guiding precisely in order to guarantee sufficient entertainment of the visitors [6].

All authors are with the Humanoid Robots Lab, University of Bonn, 53113 Bonn, Germany. This work has been supported by the European Commission under contract number FP7-610532-SQUIRREL.

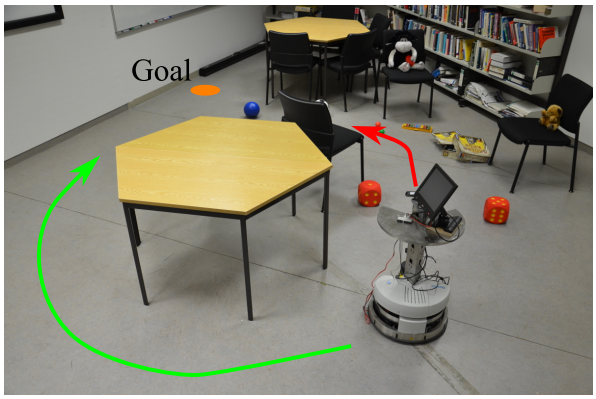

Fig. 1. Motivation of our approach. The robot can choose between two options to reach the goal location. The shorter path (red) leads through dense clutter where the robot needs to drive carefully and needs accurate sensing and pose estimation to avoid collisions. The second path (green) is longer but leads through wide free space where the robot can drive with a faster velocity profile. The work presented in this paper learns to predict the travel time along 2D paths from training data to decide which path leads to the fastest completion time.

When it comes to navigation, and most applications of mobile robots involve phases of locomotion in an environment, the travel time can make up a significant amount of the total task completion time. The fastest option, however, often differs from the shortest or cost-minimal solution found using 2D path planning on a cost grid map. Consider for example Fig. 1, where the robot has two options to reach the goal location. The first option (red path) contains narrow passages with several cluttered objects. The second option (green path) traverses solely free space. Taking the red path, the robot needs to drive slowly since it needs accurate sensing and precise motion execution to avoid collisions. At very tight spots the robot even has to stop frequently and rotate in order to adjust its heading. In contrast, the green path leads through free space where the robot can drive with higher velocities. Thus, the robot is faced with the question which route to choose to reach the goal location as quickly as possible.

With two-stage navigation systems where a global path planner is combined with a local motion controller, the precise outcome of the motion execution is typically difficult or impossible to predict in advance, especially when traversing narrow or cluttered passages. A reactive robot control system such as the popular Dynamic Window Approach (DWA) [4] can neither ensure a time-optimal trajectory, nor control stability, nor convergence of the system [7]. Additionally, many sources of noise randomly influence the navigation performance, i.e., the slippage of the wheels, noise in the

sensor measurements, and inaccuracies in the localization. The resolution of the grid-based environment representation or the choice of the navigation cost function can also influence the performance significantly.

In this paper, we present a novel method to predict the travel time for a mobile robot based on path features that are available ahead of the execution time. This will allow the robot to evaluate different options and choose the path that is predicted to be the most time-efficient. Given a 2D path in a grid representation of the environment, our method predicts the completion time by means of regression analysis based on general path characteristics such as its length, clearance, and curvature. We extensively evaluated our method in various environments of different complexity. As the experiments show, our method is able to realistically estimate the completion time of 2D grid paths and outperforms a prediction that is solely based on the path length. To the best of our knowledge, this is the first approach that can efficiently predict the completion time of navigation tasks without applying computationally expensive calculations using an exact kino-dynamic model of the robot.

II. Related Work

Grid-based planners are usually very fast in computing a spatial global path to a goal location and are widely used in navigation systems for wheeled robots [8], [9], [10]. Typically, these systems employ a local reactive collision avoidance method to generate actual velocity commands for the robot to follow a globally computed 2D path such as the DWA [3], [4] or the Nearness Diagram Method [2]. Note that the global paths typically contain sharp corners in the vicinity of obstacles so that these low-level reactive systems explicitly take into account deviations from the global 2D grid path to achieve smooth trajectories with a faster progress towards the goal location compared to stopping and turning on the spot. A different technique was developed by Stachniss and Burgard who suggested to plan directly in a 5D space in a local channel around the global 2D path with A* [11]. This space additionally contains the orientation as well as discretized translational and rotational velocities. In this way, smooth trajectories that directly consider kino-dynamic constraints are obtained.

All the approaches mentioned above assume that the time-optimal trajectory lies close to the computed 2D path, which is often the case but might not be true in the presence of many obstacles. In such cases it might actually be better to also take into account different paths with fewer obstacles that need to be passed. Therefore, we present a method that learns the time the robot needs to navigate along a given 2D path based on path characteristics. Based on general features of a 2D path, the robot can then estimate the completion time it would need to follow the path towards the goal location and choose the best option among different possibilities.

Murphy and Newman considered robots operating in large outdoor environments and developed an approach to trade off the risk of planning a path with suboptimal length for planning time and plan over probabilistic costmaps [8]. To create such a probabilistic costmap, one typically needs a priori knowledge about the terrain such as an overhead image of the environment. The work of Murphy and Newman focuses on traversing special types of terrain, whereas our approach is optimized for dealing with challenging indoor environments with mainly flat floor where the terrain properties play a minor role for the performance. Zhu and Qingbao proposed path planning based on a genetic algorithm [12]. The authors introduced functions to describe path characteristics that allow to choose an optimized path from a given set. This approach does not consider the motion control system of the robot. Philippsen [13] used probabilistic navigation functions to trade off the risk of colliding with dynamic obstacles against the length of a detour to avoid those. However, the approach requires tuning and user-defined heuristics and does not involve a trained model.

Lau *et al.* [7] developed an approach to time-optimal control from sparse way points to the goal based on quintic Bézier splines. Starting from a given straight-line path, the trajectory is optimized for smoothness and time taking into account the constraints of the system. In this paper, we consider general navigation in environments of different complexity also containing highly cluttered and narrow passages. Our goal is to estimate the travel time based on simple, readily available features describing the path characteristics and in this way enable the robot to choose the best option, i.e., the path assumed to lead fastest to the goal using a standard DWA-based controller that generates velocity commands in an efficient manner.

Recently, Regier *et al.* proposed to estimate obstacle densities beyond observed areas based on already detected objects and predict corresponding traversal costs [14]. The authors hereby assume that the robot possesses only partial knowledge about the obstacles in its surrounding. The work presented in this paper can be combined with such a prediction step in order to recompute the best path whenever new information about obstacles becomes available.

III. Navigation Framework

In this work, we assume that a 2D grid map representation of the environment is given and consider a mobile robot control system that applies a classical two-stage navigation approach, where the global path is computed by a grid planner on a costmap.

Afterwards, the task of the reactive local controller is to find velocity commands that allow the robot to follow this global path with collision-free motions. A well-known approach is to use roll out or look-ahead methods as in the DWA [3], [4]. A DWA-controller considers at each time step only a local costmap, which is a small fraction of the environment model allowing the system to operate in real time. The basic principles of the DWA approach are as follows:

At each time step, a local goal on the global 2D path is determined right outside the local costmap. In the second step, a set of feasible steering commands from the robot control space is computed to reach the local goal. For each sampled velocity command a simulated trajectory is determined and

evaluated through a predefined cost function. Based on the evaluation, the velocities of the best trajectory are taken for the control. The terms of the cost function used to evaluate the trajectories are based on the distance to the global path, the distance to the local goal, and the traversal cost given by the costmap.

Such a two-layered approach can generate robust, collision-free motion even in obstacle-laden environments. However, the highly unpredictable nature of the DWA controller as well as the influence of noisy perception and localization make the estimation of the completion time of a motion task a difficult endeavor. The cheapest path in a costmap is not always the best choice as it may lead through narrow or cluttered passages and it may be in general hard to follow without slowing down and rotating on the spot.

To illustrate this, we performed two navigation experiments in a large cluttered environment where the robot had the choice to drive through or around the cluttered region. Fig. 2 shows this scenario with the two paths and their corresponding velocity profiles. The red profile in Fig. 2 shows that driving around a cluttered region allows the robot to navigate at full speed and reach the goal in a shorter time even though the total path length is longer. Driving through the dense clutter, however, leads to higher localization errors due to more frequent rotations, repeated velocity drops in order to avoid collisions and on-spot rotations, which are necessary in regions with very little space to navigate.

Thus, the estimated completion time is in many situations a much stronger selection criterion than the path costs, but due to the lack of a dynamic model in the path computation phase the completion time is typically not known a priori. In the following, we introduce a novel approach to estimate the completion time from path features to enable the robot to choose the most promising path among different possible routes through the environment.

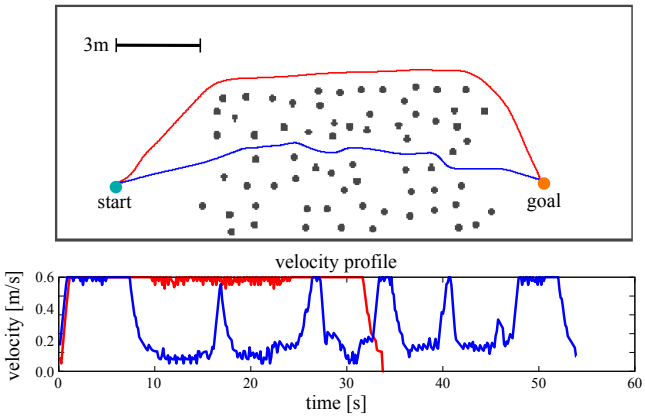

Fig. 2. Example velocity profiles of a robot driving through and around the cluttered region. Obstacles are displayed in grey. In order to reach the goal, the robot has the choice to either navigate through (blue path) or around (red path) the clutter. The corresponding velocity profiles are displayed to indicate common navigation issues that arise from navigating close to multiple obstacles. Considering the red profile, it is easy to see that the robot can constantly drive close to the maximum velocity and, thus, reaches the goal after only 34s with a traveled distance of 18.54m. The blue profile shows that constant speed drops occur, which are necessary in order to avoid collisions. Additionally, on-spot rotations are performed if too tight directional changes are necessary. This leads to a lower traveled distance of 14.46m while the execution time increased to 55.36s.

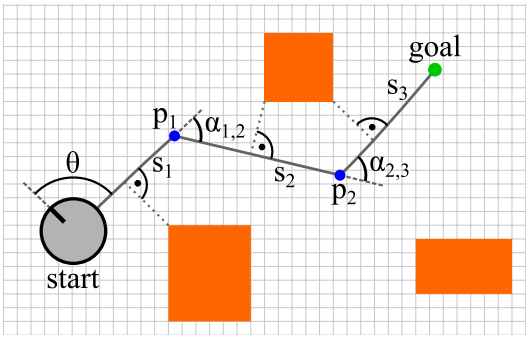

Fig. 3. Visualization of the features we use for path characterization. The figure shows an example path from the robot's current positions (grey circle) to the goal location through an environment with three obstacles (yellow rectangles). The path consists of three segments and two nodes. The angles $\alpha_{1,2}$, $\alpha_{2,3}$, and θ used in Eq. (2) are also shown. The shortest distances between the segments and the obstacle cells are illustrated as black dashed lines and are used in Eq. (3).

IV. Predicting Travel Time from Path Characteristics

In principle, the only way to predict the completion time is to simulate the path execution and measure the time the robot takes to navigate to the goal. Our idea is to apply a machine learning approach and to train a predictor function for the execution time based on a small number of generic features that can be efficiently computed from a given global 2D grid path.

A. Features for Describing Path Characteristics

We define a path $\mathbf{P} = \{\mathbf{p}_0, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ between the current position of the robot and the goal location as a sequence of two-dimensional coordinates (nodes) $\mathbf{p}_i = (x_i, y_i)$, $i \in \{0 \ldots n\}$, as illustrated for an example path in Fig. 3. A segment $\mathbf{s}_i$ of the path is then given by the vector $\mathbf{s}_{i+1} = \mathbf{p}_{i+1} - \mathbf{p}_i$. We found out that the length of the path, its clearance, and smoothness are expressive features that can be used to effectively estimate the time the robot needs to travel along the path towards the goal location. These features are described in detail in the following:

1) The *total length* of the path is given by the sum of the lengths of each path segment:

$$L_p = \sum_{i=1}^{n} |\mathbf{s}_i| \tag{1}$$

2) The *average smoothness* of a path expresses its deviation from being a straight line:

$$S_p = \frac{\theta + \sum_{j=1}^{n-1} \alpha_{j,j+1}}{n}, \tag{2}$$

where θ is the angle between the initial heading of the robot and the first path segment, and $\alpha_{j,j+1}$ denotes the angle between two path segments $\mathbf{s}_j$ and $\mathbf{s}_{j+1}$. For example, $\alpha_{1,2}$ in Fig. 3 denotes the angle between $\mathbf{s}_1$ and $\mathbf{s}_2$.

3) Finally, the *average path clearance* is computed as follows:

$$C_p = \frac{\sum_{i=1}^{n} \max\{D_{max} - D_{min}(\mathbf{s}_i, c_{occ}), 0\}}{n} \quad (3)$$

using the shortest distances $D_{min}(\mathbf{s}_i, c_{occ})$ between each path segment $\mathbf{s}_i$ and the occupied cells c_{occ} closer than a threshold $D_{max} > 0$. We assume that obstacles with a distance greater than D_{max} have no effect on the task execution. The clearance is illustrated in Fig. 3 as a dashed line from a path segment $\mathbf{s}_i$ to its closest obstacle.

B. Prediction of Travel Time

Using the three path features defined above, we train a predictor function

$$T_p = \mathscr{F}(L_p, S_p, C_p) \quad (4)$$

that estimates the expected path execution time T_p based on the total length, average smoothness, and average clearance of the path. These features are readily available before the actual path execution starts by a local controller.

C. Regression Models

Regression is a common tool in statistical analysis to find relationships among variables. The goal of the regression analysis is to find a model that fits well the given data points use it for prediction afterwards. Different models can map different types of relationships between the variables. Linear regression, for example, is a very fast algorithm, but can only model linear coherences. A special case of linear regression is the simple linear regression that fits the data with a simple regression line. Linear regression, in contrast, models the relationship among several independent variable to predict the requested dependent quantity. For systems with non-linear behavior, linear predictors are often not sufficient. Better results can be achieved with more advanced methods. Support vector machines, for example, are kernel methods that map the data input into a high-dimensional feature space using kernel functions. This kernel trick allows to detect non-linear coherence in data sets.

To find the right regression approach for our problem, we evaluate the simple linear regression, linear regression, and support vector method for the task of completion time prediction based on the path features length, smoothness, and clearance that are described above.

V. Experiments

In this section, we discuss the data collection process, the regression analysis, as well as the prediction results in different environments.

A. Data Collection

Our goal is to obtain a single regression model that covers as many scenarios as possible. In order to gather data that is well distributed over the feature space, we performed experiments on a variety of maps such as the Willow office environment and artificially created maps (see Fig. 4). One

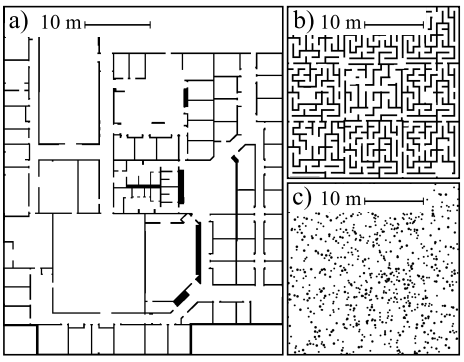

Fig. 4. Maps used in the experiments. (a) Office environment created by Willow Garage, (b) narrow maze-like environment, and (c) cluttered environment with many randomly distributed obstacles.

type of artificial maps we used are highly cluttered maps consisting of uniformly distributed or Gaussian distributed pillars, where pillars are randomly generated in varying quantity of 75 pillars per hundred square meters, 50 pillars per hundred square meters, and 25 pillars per hundred square meters with varying radii from 20-60 cm according to the distribution used. Another type of artificially created maps consist of narrow maze-like structures with a corridor width between 0.6 m and 0.9 m. We generated three different artificial maps of both types.

To collect training data, we used the Gazebo simulation environment [15] to simulate a model of the omnidirectional Robotino robot by *Festo Didactics*. We first compute a global path from the current position of the robot to a goal position and then let the robot follow this path with a DWA-controller. To obtain ground truth data, we measure the task completion time when the position of the robot is close to the x-y-coordinates of the goal position. The final heading is not considered. In each experiment, the start position, the initial heading of the robot, and the goal position were chosen randomly. We used an A*-planner for computing the global path. The lengths of the grid-based paths varied between 4 m and 50 m. The A*-planner and the DWA-controller are implemented in the ROS navigation stack [16].

The choice of the parameters of the navigation systems has a pivotal influence on the performance of the robot during the experiments. We found the following parameters to work best in practice. We used a resolution of 5 cm for the global costmaps of the environments and 1 cm for the local map. The frequency of the control loop was set to 8 Hz and the size of the local costmap was chosen to be 1.5 m $\times$ 1.5 m. The maximum linear velocity was set to $0.6\frac{m}{s}$ and the maximum rotational velocity was set $0.6\frac{rad}{s}$. The acceleration limits for linear and rotational movement were set to $0.7\frac{m}{s^2}$ and $0.7\frac{rad}{s^2}$, respectively. Naturally, the capabilities of the underlying physical system are instilled into a trained regressor. A deviation from the configuration parameters at a later time may work to some extent, we have not evaluated this in our work so far, but in general it must be assumed that the model is not transferable to a new system with significantly different navigational capabilities. The training

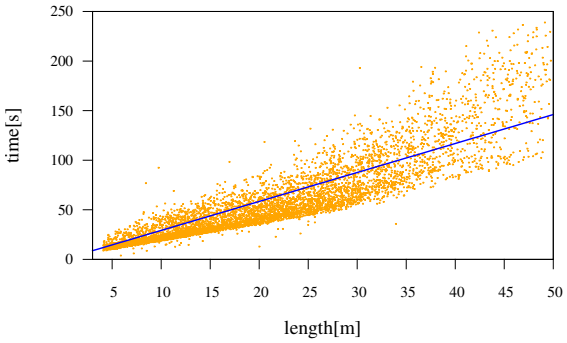

Fig. 5. Completion time of the simulated execution of the generated paths in the three environments (yellow) over path length. With increasing length, the data spreads broader around the regression line (blue). These results were obtained from the experiments with noisy localization.

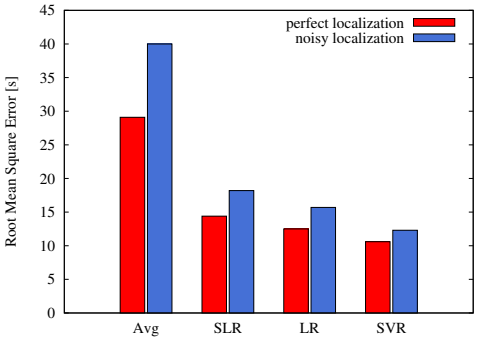

Fig. 6. Comparison of four different regression methods for perfect (red) and noisy (blue) localization. The regression methods are the constant average estimate (Avg), a simple linear estimate (SLR) based on path length only, linear regression (LR) using all features, and support vector machine regression (SVR) also using all features. As can be seen, the SVR has the smallest RMSE of all approaches. This non-linear model seems to be the best approximation for our robot and controller setup. Furthermore, we see a clear improvement of the LR model in comparison to SLR with the additional independent features.

must be performed for each individual combination of robot and navigation software.

We created two data sets with each containing 5500 navigation tasks. The first dataset was gathered without any sources of noise, i.e., no noise in the sensors and without slippage of the wheels. In particular this also includes a perfect localization. Naturally, this model is not entirely realistic, but it helps to analyze the data with respect to the correlation of the features with the estimate. The second data set was collected from experiments with a localization system that adds noise to the simulation due to faulty pose estimates. Note that in the second data set, the sensors and motion itself are still noise-free. Using these two data sets, we can evaluate our model with noise in comparison to noise-free results and also see how much the noise in the system affects the navigation performance.

B. Regression Results

In this section, we present the results of our regression analysis. For every data set, we learned an estimator for each of the different approaches. We used a simple linear regression (SLR) method based on the path length alone as computed in Eq. (1), a linear regression (LR) model which considers all the features mentioned above (see Sec. IV-A), and we trained a support vector machine for regression (SVR), also using all features. For training and testing we used WEKA, a well-established data mining software [17]. To evaluate the different regression models, we performed a 10-fold cross validation on the data set, i.e., during one validation run, 90% of the data set is used for training and the other 10% for testing this specific model. In the next validation round, another subset of 10% is used for testing and we repeated this process 10 times until every subset of the data set has been tested. We computed the average of the root mean square errors (RMSE) of every ten testing runs. As a reference, we additionally computed the RMSE of the constant average estimator over the entire data set.

It is not sufficient to assume a linear distribution, as in particular in the presence of clutter and narrow gaps our navigation system exhibits a highly non-linear behavior. Nevertheless, linear models are easy to fit and fast to compute and thus serve as a good reference. Fig. 5 illustrates the completion time of every run in the dataset over the path length (computed according to Eq. (1)). Note that the spread of the data points (yellow) around the linear regression line (blue) increases with path length.

The regression results depicted in Fig. 6 show that the features introduced in Sec. IV have a substantial influence on the time estimation, as we can see a 14% improvement for both data sets of the LR compared to SLR. A further reduction of the prediction error can be achieved using the non-linear model. Using the SVR results in a RMSE that is further reduced by 15% compared to the LR for the data set with perfect localization and 22% for noisy localization. These results support that the system behaves highly non-linear and a linear regression method is not sufficient if a more accurate estimate is desired.

Comparing both data sets against each other, we can clearly see the influence of the noisy localization which is close to real-world runs. The results also show that some of the noise can be estimated by our approach, since the error reduction from LR to SVR is larger for noisy compared to perfect localization. This improvement stems from the fact that a much higher localization error correlates with certain non-linear behaviors, e.g., rotating fast on the spot or traversing monotone environments with only few features. Thus, the non-linear SVR method is best suited for real-world scenarios.

The evaluation shown in Fig. 6 is well suited for a comparison of the different regression approaches. Additionally, we are interested in the relative root mean square error of the estimate, which is defined as follows:

$$\sigma_{est} = \sqrt{\frac{\sum_{i=1}^{N} \frac{(Y_i - Y_i')^2}{Y_i^2}}{N}}\ , \tag{5}$$

where Y_i is the completion time of experiment i, Y_i' is the corresponding estimated value, and N is the number of

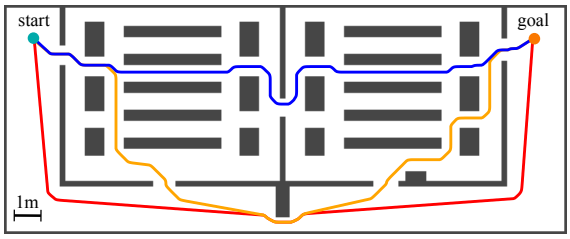

Fig. 7. A environment with two rooms and a corridor, that was not used for the training data, with three different path options to get from the start to the goal. The longest path (red) leads the robot through wide free-space area. The shortest path (blue) guides the robot through narrow space between obstacles. An alternative path (yellow) leads partly through the narrow passages and through wide-space. The the evaluation of the path is shown in Tab. I.

experiments in a set. As we evaluated a wide variety of scenarios which contained both very short and very long paths, σ_{est} is a better measurement of the relative deviation per experiment, as we first scale every separate squared error by the corresponding completion time. We computed the values of Eq. (5) for both the LR and SVR due to the superior performance compared to SLR. For the LR and SVR, σ_{est} evaluates to 0.32 and 0.13, respectively. These results show that the use of SVR not only highly decreases the average deviation, but also shows an improved estimate for the whole spectrum of path lengths. As these results show, our approach is able to predict the path completion time with an error of only 13% in average.

C. Temporal Gain

To demonstrate the temporal gain when applying our prediction, we performed an experiment on a completely new map (see Fig. 7). In this experiment, three different path choices to navigate from start to the goal location exist. The first option is the shortest path (blue), which leads through narrow areas. The red path is the longest, but it is smooth and has a high clearnace to obstacles. The third alternative consists of segments of the other two paths. Based on the completion time predicted by our approach, the longest path is chosen as the fastest option followed by the shortest path. The third path is the slowest according to our prediction. By executing all three options in simulation, the actual completion time in Tab. I confirms the prediction and the path choice. The actual temporal gain when executing the red path in comparison to the execution of the shortest path amounts to 6 s, which is 9.8% of the travel time.

VI. Conclusions

In this paper, we presented a technique to estimate the completion time for 2D grid paths. The completion time is in general not known in advance as it strongly depends on the capabilities of the underlying motion controller. Through a low-dimensional categorization of the paths using three generic features—their length, smoothness, and clearance—and the simulation of a large variety of motion tasks on different types of maps, we were able to regress an estimator that predicts the path completion time with a low error of around 10% before motion execution starts. Naturally, as the completion time depends strongly on the navigation performance of the robot, it needs to be trained individually for a specific hardware and motion controller combination.

TABLE I

Evaluation of the paths shown in Fig. 7

	red	yellow	blue
length	28.5145 m	26.0843 m	20.9335 m
clearance	0.2685	0.4401	0.5131
smoothness	0.0137	0.0457	0.0483
prediction time	47.373 s	58.756 s	50.392 s
completion time	55.512 s	63.751 s	61.511 s

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics*, vol. 20, no. 1, 2004.

[3] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, pp. 23–33, 1997.

[5] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous vehicles in city traffic*. Springer, 2009.

[6] S. Thrun, M. Bennewitz, W. Burgard, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[7] C. Lau, B.and Sprunk and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2009.

[8] L. Murphy and P. Newman, "Risky planning on probabilistic costmaps for path planning in outdoor environments," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 445–457, 2013.

[9] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the PR2," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011, pp. 5568–5575.

[10] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[11] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2002.

[12] H. Jun and Z. Qingbao, "Multi-objective mobile robot path planning based on improved genetic algorithm," in *Int. Conf. on Intelligent Computation Technology and Automation (ICICTA)*, vol. 2, 2010, pp. 752–756.

[13] R. Philippsen, S. Kolski, K. Macek, and B. Jensen, "Mobile robot planning in dynamic environments and on growable costmaps," in *Workshop on Planning with Cost Maps at the IEEE Intl. Conf. on Robotics and Automation*, 2008.

[14] P. Regier, S. Oßwald, P. Karkowski, and M. Bennewitz, "Foresighted navigation through cluttered environments," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2016.

[15] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, 2004.

[16] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.