Expert-Guided Kinodynamic RRT Path Planner for Non-Holonomic Robots

José María Sanz, Miguel Hernando, Guillermo Zaragoza and Alberto Brunete

Abstract— **In this paper, an Expert-Guided Kinodynamic RRT algorithm (EGK-RRT) is presented. It aims to consider how a human pilot would navigate a kinodynamic robot. One of the characteristics of this algorithm is the fact that, unlike the original RRT for kinodynamic systems, it generates deterministic control sequences which can be reproduced as long as the sequence of references (sampled states) are known. Here, the performance of the proposed algorithm is tested against the basic RRT, showing that the EGK-RRT greatly improves in terms of execution speed. In addition to this, the influence of using a visibility check and an inertia estimation in order to select the nearest neighbor is also analyzed, demonstrating that a combination of both factors leads to a better overall performance, both in execution speed and in quality of the generated path.**

I. Introduction

In the past years, the field of path planning has dramatically improved, partly thanks to the introduction of the sampling-based planners. These kind of planners, such as Rapidly-Exploring Random Trees (RRT) [1] and Probabilistic Roadmaps (PRM) [2], are able to solve problems in environments with high-dimensional state spaces by avoiding the explicit construction of the obstacle region $\mathcal{C}_{obs}$.

Moreover, the RRT algorithm is of particular interest since it can be easily tuned to introduce new ideas, such as bidirectionality or rewiring. Considering this, a wide array of RRT versions have been created in the recent years. One of the most successful versions is RRT*, which introduced the idea of rewiring in order to ensure a minimum-cost path and is provably asymptotically optimal [3]. The RRT-RT algorithm described in [4] was designed with rough terrain navigation in mind, and proposes the use of a roughness-based metric, further proving the flexibility of RRT to adapt to different situations.

This paper is focused on the path planning of an Autonomous Underwater Vehicle (AUV). In this kind of system, dynamics play an important role, requiring both velocity and acceleration bounds to be satisfied. Therefore, specific kinodynamic planners have to be used, such as the kinodynamic RRT algorithm, which samples control actions in order to find the best one [5]. Extensions for other popular planners have been designed to account for kinodynamic problems, such as RRT* [6], which also includes the idea of using a fixed-final-state-free-final-time controller. However, these planners behave in a similar way when dealing with complex problems or with simple ones, such as planning in an open space. Considering this, we pose the following question: does it not make sense to try to directly reach the goal if it lies within sight?

In this paper, we propose the use of a human-like local controller in order to connect a randomly sampled state with the closest state from the motion tree. This idea gives the controller greater flexibility and the ability to perform better than a generic kinodynamic-RRT in specific situations, such as open spaces. The control sequence generated by the controller relies on the fixed-final-state-free-final-time concept, which allows the reach of the sampled state in all cases. This differs from the standard kinodynamic RRT algorithm in that no control sampling is required. Instead, the control action selection is deterministic, thus generating the same path as long as the environment remains unaltered. However, the main strategy of state sampling from RRT remains unchanged so as to keep the RRT exploration feature.

In addition to the core idea, two modifications are proposed and evaluated. A visibility check punishes the nodes from which there is no visibility to the sampled state, with the objective of generating the straightest path to the sampled state. The concept of visibility has been applied multiple times in other planning algorithms, such as [7] and [8], and here it is analyzed as a part of the proposed implementation. On the other hand, a custom metric aims to reward the closest nodes taking into account the dynamics of the robot. The proposed metric is computed by propagating each node considering a control action of value zero until the robot stops. The metric is computed as the Euclidean distance between the stop position and the sampled state. These modifications are related to the proposed concept of a human-like controller, but independent of it. Therefore, they can be of use, and have been considered as such, in other planning algorithms.

II. System

The proposed algorithm is to be tested on an AUV (Autonomous Underwater Vehicle). The robot, named Wasabi (Fig. 1), is being designed by Robdos Team Underwater Robotics, from the Universidad Politécnica de Madrid. The robot has three degrees of action: two horizontal propellers for the surge movement (along the longitudinal X axis) and one vertical propeller for the heave movement (along the Z axis).

The state vector of the robot is 12-dimensional, with position, orientation and linear and angular velocities. However, the Wasabi AUV is built in such a way that it is always in a horizontal position. Therefore, the roll and pitch

José María Sanz is with GMV Innovating Solutions, Tres Cantos, Madrid 28760 Spain (e-mail: jmsanz@gmv.com).

Guillermo Zaragoza, Miguel Hernando, IEEE Member, and Alberto Brunete are with the Centre for Automation and Robotics (CAR UPM-CSIC), Universidad Politécnica de Madrid, Spain (e-mail: g.zaragoza@alumnos.upm.es, {miguel.hernando, alberto.brunete}@upm.es).

Fig. 1: Robot Wasabi, made by Robdos Team

degrees of freedom are not controllable and constant. This reduces the state space to an 8-dimensional space. In addition, environmental disturbances and the Coriolis effect are disregarded due to not having a great impact while making the model more complex. The following equations, in [13] and [14], describe the dynamic behavior of the robot.

$$\ddot{x} = \dot{u} = \frac{(\tau_2+\tau_3)-(X_u+X_{u|u|}|u|)\cdot u}{m+X_{\dot{u}}} \tag{1}$$

$$\ddot{y} = \dot{v} = \frac{-(Y_v+Y_{v|v|}|v|)\cdot v}{m+Y_{\dot{v}}} \tag{2}$$

$$\ddot{z} = \dot{w} = \frac{\tau_1 - (Z_w + Z_{w|w|}|w|)\cdot w + (W - B)}{m + Z_{\dot{w}}} \tag{3}$$

$$y\ddot{a}w = \dot{r} = \frac{(l_2\tau_2 + l_3\tau_3) - (N_r + N_{r|r|}|r|)\cdot r}{l_z + N_{\dot{r}}} \tag{4}$$

Where X_u and $X_{u|u|}$ represent the linear and quadratic damping parameters for each motion (surge, sway, heave and yaw), $X_{\dot{u}}$ represents the added mass, τ_i represents the force commanded to each motor, W is the weight of the robot and B the weight of the water displaced by the robot.

Consequently, the state space of the Wasabi robot studied is an 8-dimensional vector including the following variables.

$$state = [x \quad y \quad z \quad yaw \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad y\dot{a}w] \tag{5}$$

III. Kinodynamic RRT

One of the main features of the RRT algorithm is that it takes into account the possibility of working with non-holonomic and dynamic restrictions in environments with a high number of degrees of freedom. This feature differentiates the aforementioned RRT algorithm from other path planning algorithms, like PRM or potential fields. Due to their structure, these algorithms do not seem ideal for kinodynamic planning, though they have been adapted in [9] and [10] under specific assumptions. The RRT algorithm for kinodynamic systems is presented in Fig. 2.

The idea of the kinodynamic RRT that sets it apart from the generic RRT is that the control action u cannot be computed. It is instead sampled in the NEW_STATE function, randomly choosing the value of the control action as well as its duration. From all the sampled actions, the one that pushes x_{near} the closest to x_{rand} is the selected motion.

IV. Expert-Guided Kinodynamic RRT

The proposed algorithm is presented in Fig. 3 and is given the initial state x_{init}, the goal state x_{goal} and the number of iterations $K_{iteratrions}$ after which the sampled state is

```
BUILD_RRT($x_{init}$)
1  $\mathcal{T}$.init($x_{init}$);
2  for $k = 1$ to $K$ do
3      $x_{sampled} \leftarrow$ RANDOM_STATE();
4      EXTEND($\mathcal{T}, x_{sampled}$);
5  Return $\mathcal{T}$

EXTEND($\mathcal{T}, x$)
1  $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x, \mathcal{T}$);
2  if NEW_STATE($x, x_{near}, x_{new}, u_{new}$) then
3      $\mathcal{T}$.add_vertex($x_{new}$);
4      $\mathcal{T}$.add_edge($x_{near}, x_{new}, u_{new}$);
5      if $x_{new} = x$ then
6          Return $Reached$;
7      else
8          Return $Advanced$;
9  Return $Trapped$;
```

Fig. 2: Kinodynamic RRT algorithm

```
BUILD_EGKRRT($x_{init}, x_{goal}, K_{iterations}$)
1  $\mathcal{T}$.init($x_{init}$);
2  for $k = 1$ to $K$ do
3      if $mod(k, K_{iterations}) = 0$ then
4          $x_{rand} \leftarrow x_{goal}$;
5      else
6          $x_{rand} \leftarrow$ RANDOM_STATE();
7      EXTEND($\mathcal{T}, x_{rand}$);
8  Return $\mathcal{T}$

EXTEND($\mathcal{T}, x$)
1  $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x, \mathcal{T}$);
2  $\{u_{new}, \Delta t\} \leftarrow$ LOCAL_PLANNER($x_{near}, x$);
3  $x_{new} \leftarrow$ PROPAGATE($x_{near}, u_{new}, \Delta t$);
4  $\mathcal{T}$.add_motion($x_{new}, x_{near}, \Delta t$)
```

Fig. 3: Expert-Guided Kinodynamic RRT algorithm

substituted with x_{goal}. The main difference from the basic kinodynamic RRT lies in the fact that no sampling of the action space is performed, whereas the RRT algorithm samples the action control in order to choose the best one. Instead, the control action is deterministically computed by means of an expert system which encompasses the behavior that a human pilot would exhibit.

Regarding the proposed path planning algorithm, the considered states have the following structure:

$$state = [initialState \quad refState \quad time] \tag{6}$$

Each state represents where the motion starts (initialState), which state is used as a reference to generate the control sequence (refState) and for how long (time) the reference is used.

The key pieces composing the algorithm are described in the following sections.

A. Navigation function

The LOCAL_PLANNER function acts as a navigation task used to compute the control actions needed to reach the sampled state from the nearest one in the tree. This function is conceived as an expert system trying to emulate the behavior of a human pilot through a set of rules which can be modified depending on the system the algorithm is being applied to. The main objective is to avoid the creation of a whole motion tree in cases where the goal can be reached in a single step. The navigation function can be any as long as it does not include non-deterministic components. Therefore, for given initial and target states, the generated actions would always be the same.

Instead of creating one single control action, the proposed navigation function creates a whole control sequence composed of several control actions which have different durations. The consequence is that the space is rapidly explored, since the branches of the motion tree are long enough to reach the sampled state.

The AUV considered in this paper has three degrees of freedom in the action space (left, right and heave propellers), but a simplified state space of 8 dimensions. The proposed strategy considers two major areas regarding the position of the target space relative to the position of the robot. Any other implementation of the navigation function could have been used, as long as it can be used to reach any state in an open space. The proposed navigation function is shown in Fig. 4.

- If the target is within a visibility angle α of the surge direction of the robot (usually, the longitudinal axis), the robot moves straight to the target (purple-colored area in the figure).
- If the target is within a wider visibility angle β, the robot slightly turns left or right while moving forwards in order to correct its trajectory (blue-colored area).
- If the target is out of the visibility angle, the robot turns left or right making sharper turns the closer the target is to the robot (green, yellow, orange and red areas).The word "data" is plural, not singular.

In case that the navigation function encounters a situation in which, due to the workspace configuration, the heuristic fails, the state sampler in RANDOM_STATE allows the algorithm to continue the exploration and avoid getting stuck.

The navigation function validation is performed by running multiple planning tests in different configurations, as shown in Fig. 5, in which different configurations of initial speed angle and distance to the goal have been considered. In order for a navigation function to be considered as valid, all the tests in the validation process have to be completed with a successful path planning, between the initial state and the goal state.

B. *Propagator*

The PROPAGATE function propagates the state of the robot considering the generated control sequence and computes its final position and speed. The method used for the integration of the acceleration equations (1-4) is the fourth-order Runge-Kutta method.

C. *Visibility check*

The selection of the nearest state in the tree is remarkably important in order to generate the best possible trajectory. Keeping on with the idea of making the navigation as human-like as possible, a visibility component has been introduced in the metric used to compute the distance between two states, punishing the states of the tree which do not hold a direct line-of-sight with the target state. The implemented method is just a simplistic option seeking to encourage choosing visible states which, at first glance, have higher probabilities of resulting in collision-free trajectories.

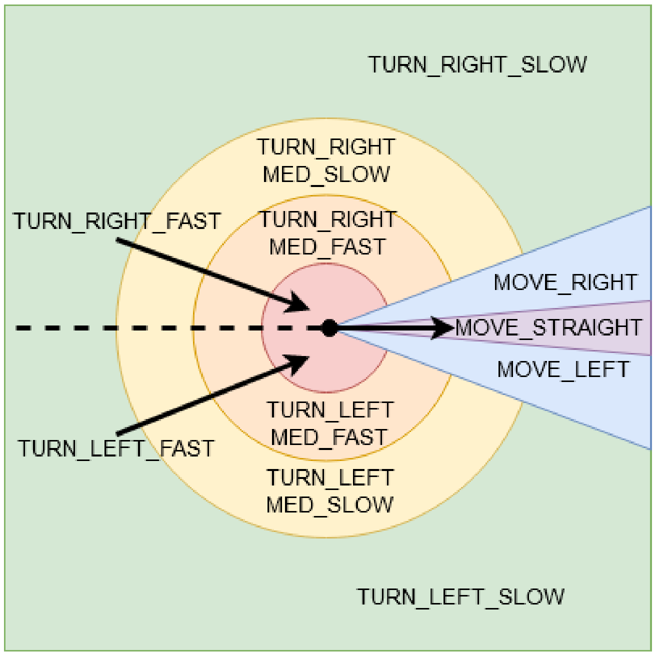


Fig. 4: Detail of the proposed navigation function

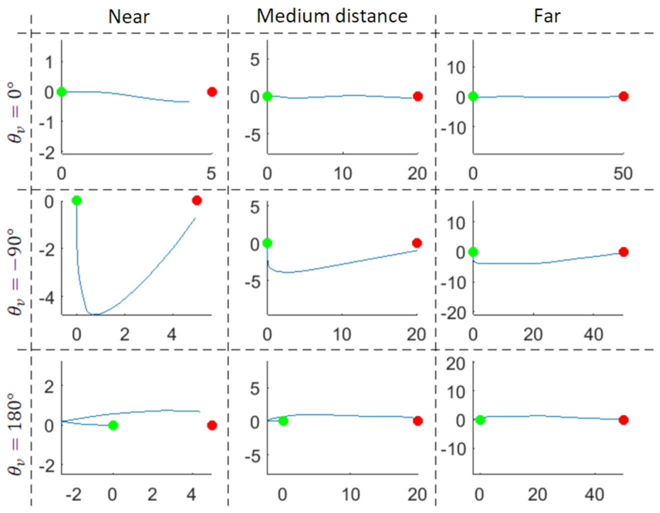


Fig. 5: Validation of the navigation function. Initial position in green and goal position in red.

D. *Inertial effect estimation*

When working with kinodynamic systems, the inertial effect has to be considered during the propagation. However, the inertial effect can also be determinant when choosing the nearest neighbor of the tree, since a state that is geometrically closer to the target can result in a longer trajectory than a farther state if the inertia moves it in a different direction.

In order to take this effect into account, the Euclidean metric has been modified by estimating the position where the speed of the system is minimal after applying a null control action. The exact value is not needed, since the idea is to simply know where the inertia would lead the robot. For the AUV, the equation system is described in (7).

$$\begin{cases} t = \dfrac{v_{surge} - v_{surge_0}}{a_{surge}} \\ t = \dfrac{v_{sway} - v_{sway_0}}{a_{sway}} \\ v = \sqrt{v_{surge}^2 + v_{sway}^2} \end{cases} \tag{7}$$

Solving the system and imposing the condition of the speed (v) being minimal, the resulting propagation time (t) is described by (8).

$$t = \frac{-v_{surge_0} \cdot \dfrac{a_{surge}}{a_{sway}} - v_{sway_0}}{a_{sway} \cdot \left(1 + \dfrac{a_{surge}^2}{a_{sway}^2}\right)} \tag{8}$$

V. Experimental Results

Several tests have been carried out using different configurations of the RRT and EGK-RRT. The objective of this section is to present those results and analyze the performance of the EGK-RRT and compare it to the generic RRT. The influence of the visibility check and the estimation of the inertial effect are studied independently to determine their influence in the effectiveness of the algorithm.

For ease of evaluation, the algorithms have been implemented in MATLAB. However, an implementation in OMPL [11] has been done as well in order to represent three-dimensional trajectories and benchmark [12] the EGK-RRT against other kinodynamic planners. For the MATLAB tests, the model has been simplified, considering only the two-dimensional situation, but including the dynamic model and restrictions.

Besides, the robot has been modeled as a dot, since its shape is not relevant in order to make the comparison between algorithms. However, in order for the algorithm to be implemented in the real robot, the elongated shape would have to be taken into account.

A. Environment considerations

For the tests results analyzed in this section, three different environments have been designed, all of which can be seen in Fig. 6.

The first environment represents a typical path planning problem, with sparse obstacles that the robot has to avoid in order to reach the goal. The second environment is devoid of obstacles, meaning that the optimal trajectory in order to reach the goal is a straight line. Lastly, the third environment poses the difficulty of including a narrow passage that the robot has to overcome.

For all the tests performed, the robot starts with zero speed and it is initially facing right. Besides, the executions are limited to a maximum number of iterations before the planning is considered to have failed.

B. Algorithm comparison

First, the proposed Expert-Guided Kinodynamic RRT is compared to the generic RRT in order to check whether the basic approach (not including either visibility or inertial effect estimation) improves the results of the generic RRT.

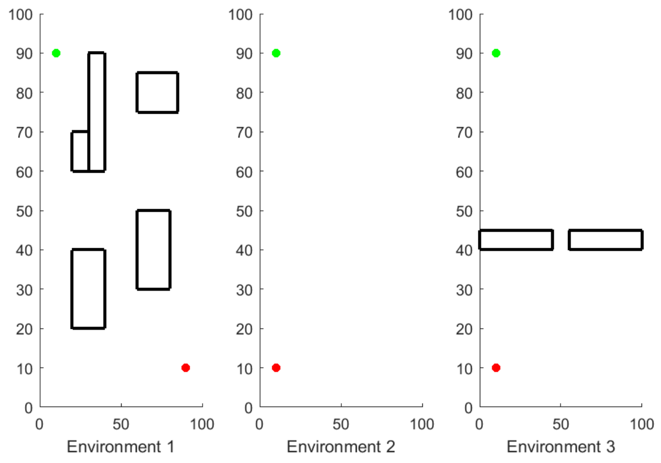


Fig. 6: Test environments. Initial position in green and goal position in red.

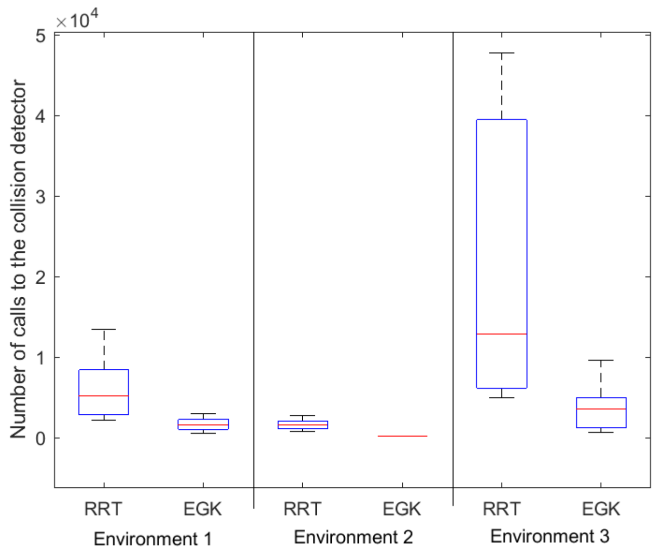


Fig. 7: Comparison of calls to the collision detector

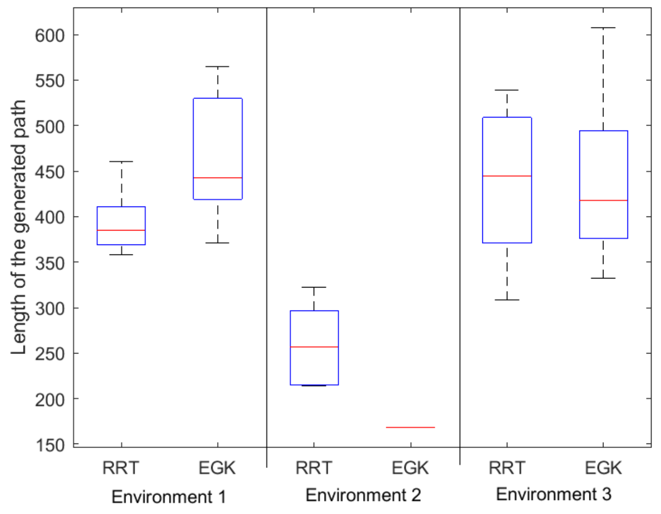
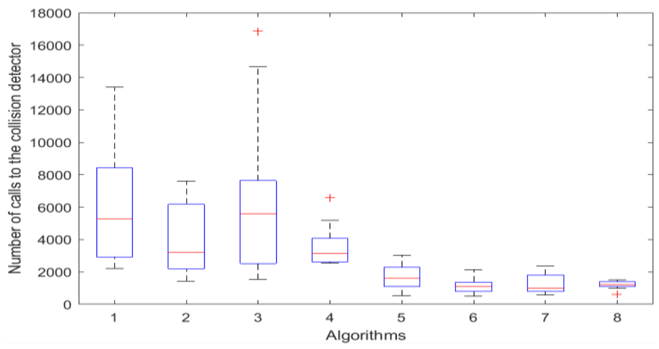


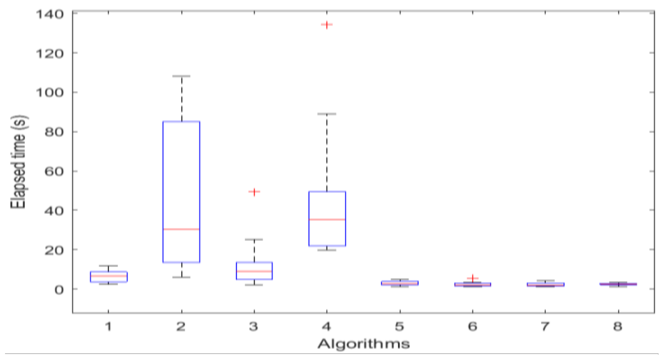
Fig. 8: Comparison of path length

The results for the three environments are shown in Fig. 7 and Fig. 8. It can be noticed that the proposed algorithm



Algorithm	1	2	3	4	5	6	7	8
Min	2222	1424	1542	2539	545	493	570	616
Max	13417	7594	16855	6617	3045	2143	2352	1524
Q1	3763	2193	25222	2633	1081	790	785	1082
Q2	5697	3215	5583	3144	1620	1092	985	1199
Q3	8420	6177	7638	4061	2304	1373	1805	1384

Fig. 9: Comparison of calls to the collision detector and quartile data. Algorithms: (1) basic RRT, (2) RRT with visibility, (3) RRT with inertia estimation, (4) RRT with both modifications, (5) basic EGK-RRT, (6) EGK-RRT with visibility, (7) EGK-RRT with inertia estimation, (8) EGK-RRT with both modifications.



Algorithm	1	2	3	4	5	6	7	8
Min	2.55	6.15	2.17	19.89	1.24	1.10	1.20	1.36
Max	11.61	108.19	49.53	134.43	4.84	5.52	4.12	3.19
Q1	3.68	13.53	4.68	21.89	2.02	1.55	1.47	2.18
Q2	6.49	30.16	8.96	35.35	2.86	2.17	1.83	2.55
Q3	8.69	85.13	13.64	49.55	3.82	3.03	3.11	2.87

Fig. 10: Comparison of elapsed time and quartile data. Algorithms: (1) basic RRT, (2) RRT with visibility, (3) RRT with inertia estimation, (4) RRT with both modifications, (5) basic EGK-RRT, (6) EGK-RRT with visibility, (7) EGK-RRT with inertia estimation, (8) EGK-RRT with both modifications.

significantly improves the results obtained when it comes to the time spent performing the calculations (the elapsed time heavily depends on the number of calls to the collision detector). However, the generated path is slightly longer than the one generated by the RRT, since the EGK-RRT does not provide optimal trajectories. The exception is the second environment, in which the path generated by the EGK-RRT is always the shortest possible, a straight line.

C. *Influence of other factors*

In this second part of the experiments, the influence of the visibility check and the inertia estimation are studied. These factors reinforce the idea of an expert-based algorithm, since they include ideas that are intuitive for a human pilot, such as heading straight towards a visible goal and taking into account the current inertia and speed of the robot. The results for the first environment are shown in Fig. 9 and Fig. 10.

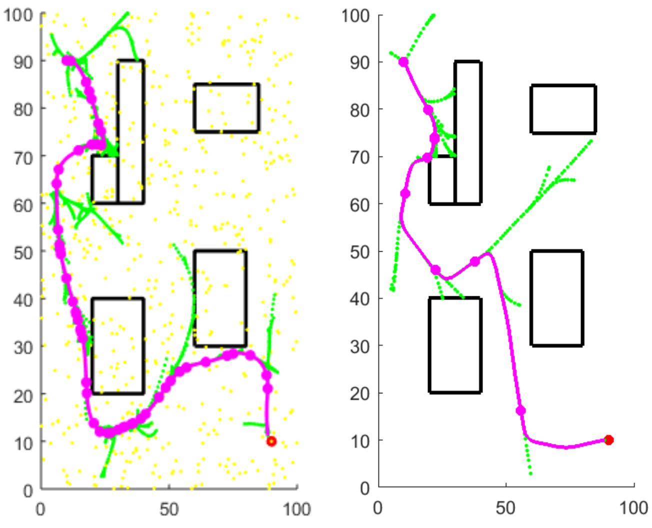


Fig. 11: Trajectory in environment 1. Left: RRT. Right: EGK-RRT

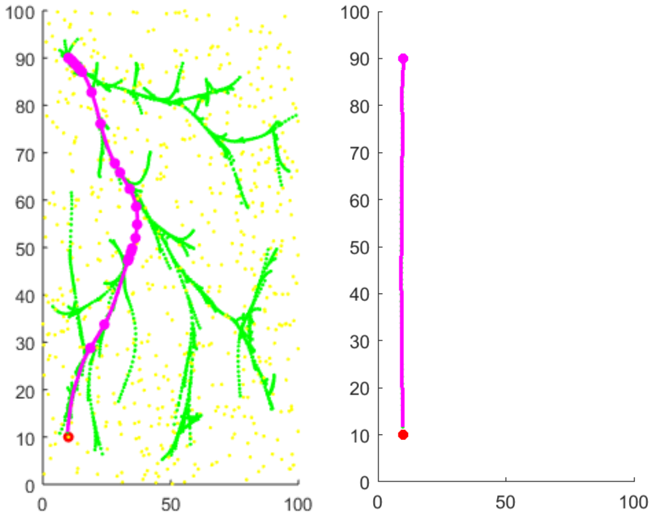


Fig. 12: Trajectory in environment 2. Left: RRT. Right: EGK-RRT

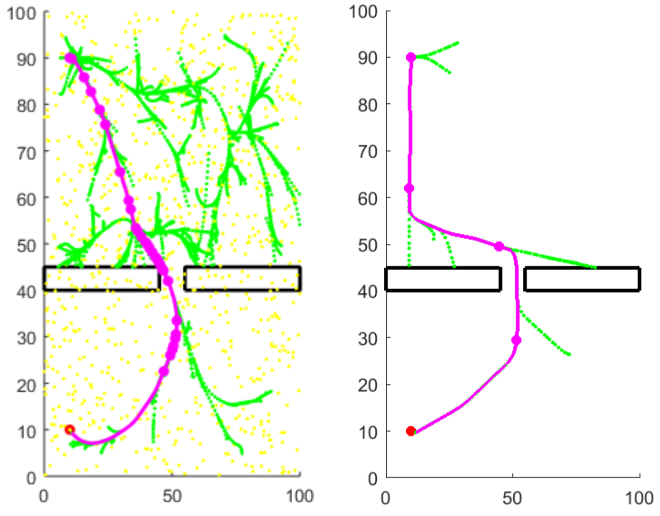


Fig. 13: Trajectory in environment 3. Left: RRT. Right: EGK-RRT

These factors are not exclusive of the Expert-Guided Kinodynamic RRT algorithm, but they can be applied to any algorithm, and have already been, in the case of the visibility check. Therefore, the two concepts have been applied to both the generic RRT and the proposed algorithm.

It can be seen that the elapsed time decreases when using the visibility check or the inertial effect estimation compared to when neither is used. However, when both are used at the same time, the deviation of the results is lesser than when using each factor on its own.

D. Generated trajectories

In this section, examples of the trajectories generated by the Expert-Guided Kinodynamic RRT algorithm are shown in Fig. 11, Fig. 12 and Fig. 13. It can be clearly seen that the generated trajectory is not optimal, but the space that has been explored is minimal, thus greatly reducing the planning time.

In Fig. 14, a test made in OMPL is shown. Using this framework, it is easier to represent three-dimensional trajectories and benchmark against other planners.

VI. Discussion, Conclusion and Future Work

In this paper, an Expert-Guided Kinodynamic RRT (EGK-RRT) algorithm has been proposed with kinodynamic systems in mind. It generates control sequences to reach the target state based on a human-like behavior which can be adapted to different applications. Besides, the influence of the visibility check and inertia estimation for nearest neighbor selection are also studied and analyzed.

The experiments performed show that the proposed algorithm significantly improves the execution time, though this comes at a cost of a less explored state space, in turn causing the resulting path to not being optimal. In particular, the results show that the algorithm performs at its best when the goal is within sight, since it can be reached directly with the navigation function, not having to rely on the sampling of the state space. Even when complicated obstacles, such as narrow passages, appear in the environment, the EGK-RRT still surpasses the original RRT.

Regarding the implementation of the algorithms, it was not fully optimized. Particularly, the visibility check is taking longer than it should. Therefore, the optimization of the algorithm could lead to even faster times. In addition to this, the implementation in OMPL has to be optimized as well in order to run a benchmark to check the whole implementation against other algorithms such as KPIECE or PDST-EXPLORE, though first implementations indicate that EGK-RRT has better results than the aforementioned algorithms.

Lastly, the algorithm has to be tested in the Wasabi robot by Robdos Team in order to check its performance in a real-world environment. The results presented in this paper have been obtained using this robot as a reference, but the algorithm is applicable to any other type of kinodynamic robot.

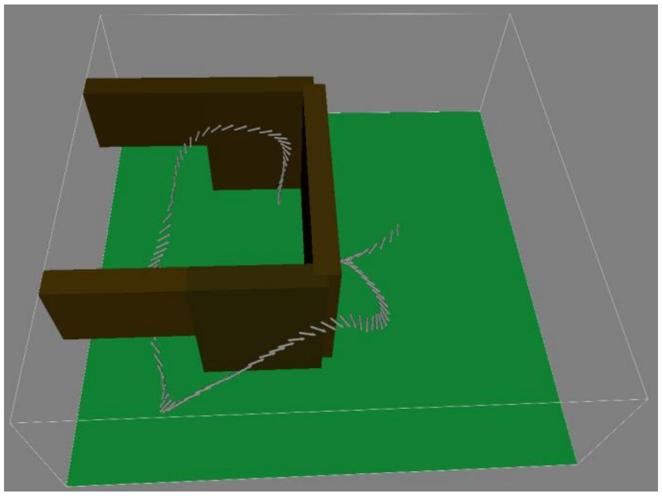


Fig. 14: Three-dimensional planning using EGK-RRT in OMPL

References

[1] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” Comput. Sci. Dept., Iowa State Univ., Ames, IA, USA, Tech. Rep. TR 98-11, 1998.

[2] L. E. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566-580, Aug 1996.

[3] S. Karaman, E. Frazzoli, "Sampling-based algorithms for optimal motion planning", *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846-894, Jun. 2011.

[4] A. Tahirovic, G. Magnani, "A roughness-based rrt for mobile robot navigation planning", *IFAC World Congress*, vol. 18, pp. 5944-5949, 2001.

[5] S.M. LaValle and J.J. Kuffner, Randomized kinodynamic planning, *Intl. J. of Robotics Research*, vol. 20, pp. 378–400, May 2001.

[6] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, 2013, pp. 5054-5061.

[7] T. Siméon, J.P. Laumond, C. Nissoux. "Visibility based Probabilistic Roadmaps for Motion Planning". In *Advanced Robotics Journal*, 14(6), 2000.

[8] F. Shkurti and G. Dudek, "Maximizing visibility in collaborative trajectory planning," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014, pp. 3771-3776.

[9] A. Ladd and L. E. Kavraki, "Generalizing the analysis of PRM," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, Washington, DC, USA, 2002, pp. 2120-2125 vol.2.

[10] A. Sánchez, R. Cuautle, R. Zapata, M. Osorio, "A reactive lazy PRM approach for nonholonomic motion planning", *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, pp. 542-551, 2006.

[11] I. A. Sucan, M. Moll and L. E. Kavraki, "The Open Motion Planning Library," in *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72-82, Dec. 2012.

[12] M. Moll, I. A. Sucan and L. E. Kavraki, "Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization," in *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96-102, Sept. 2015.

[13] T. I. Fossen, Handbook of Marine Craft Hydrodynamics and Motion Control, Jhon Wiley & Sons Ltd, 2011.

[14] J. Vervoort, Modeling and Control of an Unmanned Underwater Vehicle, Canterbury: University of Canterbury, 2008.