FDA*: A Focused Single-Query Grid Based Path Planning Algorithm

Submitted: 4th October 2021; accepted: 9th February 2022

Mouad Boumediene, Lamine Mehennaoui, Abderazzak Lachouri

DOI: 10.14313/JAMRIS/3-2021/17

Abstract:

Square grid representations of the state-space are a commonly used tool in path planning. With applications in a variety of disciplines, including robotics, computational biology, game development, and beyond. However, in large-scale and/or high dimensional environments the creation and manipulation of such structures become too expensive, especially in applications when an accurate representation is needed.

In this paper, we present a method for reducing the cost of single-query grid-based path planning, by focusing the search to a smaller subset, that contains the optimal solution. This subset is represented by a hyper-rectangle, the location, and dimensions of which are calculated departing from an initial feasible path found by a fast search using the RRT algorithm. We also present an implementation of this focused discretization method called FDA*, a resolution optimal algorithm, where the A* algorithm is employed in searching the resulting graph for an optimal solution. We also demonstrate through simulation results, that the FDA* algorithm uses less memory and has a shorter run-time compared to the classic A* and thus other graph-based planning algorithms, and at the same time, the resulting path cost is less than that of regular RRT based algorithms.*

Keywords: *motion planning, grid-based, path planning, mobile robots*

1. Introduction

Motion planning for robots has received a substantial amount of attention in the last two decades, due to the increasing integration of robots in modern industry and even in many tasks of our daily lives. In addition to this, motion planning plays a key role in autonomous robot navigation, and numerous other disciplines such as computational structural biology [1, 18], crowd simulation [14, 19], and video game development [2].

The fundamental problem to be solved is how to compute a path that allows the robot to move from an initial to a target location in the state-space while avoiding the surrounding obstacles. One of the most common approaches to solving this problem is to discretize the continuous state space, through a deterministic grid with a pre-defined resolution. From there the implicit graph within this grid is searched using graph search algorithms, such as A* [6] and Dijkstra's [3], to obtain a resolution-optimal solution to the problem.

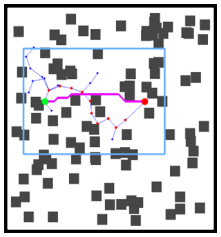


Fig. 1. Example of FDA* in a 2D Euclidean space with random obstacles, the new state-space represented by the blue borders is discretized and searched by the A* algorithm and finally the resolution optimal path is generated between the start and goal states

This is known as grid-based path planning, however, the number of grid cells required for this process grows exponentially with the number of the state space dimensions. in addition to this, the grid resolution needed for an accurate representation often leads to extremely large search spaces which cause the use of grids to be costly in terms of time and memory resources.

Furthermore, in large-scale environments, only a small subset of grid cells will contribute to finding the optimal solution especially if the distance between the start and goal states is relatively small, therefore cells outside of this subset will only consume memory space that could otherwise be allocated for other purposes.

However, finding the optimal path is not always a requirement. In some instances of the path planning problem, only an obstacle-free path is needed. In such a case many cheaper solutions can be applied. For example, various stochastic algorithms avoid creating expensive grids altogether, and instead, generate random samples in the planning domain to incrementally grow a search tree in the free space.

The Rapidly-exploring Random Trees (RRTs) presented by Laval [12] is one of these algorithms, it efficiently provides an obstacle-free path, making it useful, especially in high dimensional environments. We also mention the RRT* presented by Karaman and Frazzoli [8], which converges to the optimal solution asymptotically, as the number of iterations approaches infinity.

While discretizing the full state-space using a grid can get expensive in large or/and high dimensional environments, one possible solution is to discretize only a smaller subset of the planning domain, under the a

condition that the optimal solution belongs with certainty to this subset. The states outside this subset are therefore not necessary for the computation of the optimal path and they will only constitute a burden on the available resources. This subset as in [5]and [15], can be described by a prolate hyper-ellipsoid, and the calculation of its dimensions is carried out departing from the cost of an initial feasible path, which can be obtained efficiently by conducting a low-cost search using stochastic algorithms such as those mentioned above.
In this paper, we present an efficient method for grid discretization of the state-space to be used by grid-based planning algorithms. In this method, we consider discretizing only a rectangular subset, that tightly bounds the ellipsoidal subset where the optimal path is guaranteed to be found. For that purpose, RRT* is used to rapidly obtain an initial path, that can then be used for calculating the boundaries of this informed subset. We also present an implementation of this method, we call it FDA*(focused discretization A*), where the A* algorithm is used to plan the optimal path in the obtained grid .
FDA* uses FD(focused discretization) to improve upon classic grid-based algorithms in terms of storage-space and execution time, exploiting the fact that a lower number of states have to be handled using FD. The rest of the paper is organized as follows:
Section 2 provides a formulation of the path planning problem and a review of the related work including a comprehensive overview of the RRT* algorithm. Section 3 presents the focused discretization (FD), a method for building size-efficient grids for the use of single query grid-based planning algorithms. In section 4, an implementation of the focused discretization method called FDA* is presented. As for section 5, it is devoted to the presentation of the results of several experiments conducted on FDA* and A* algorithms, to compare the performance of our algorithm with the classical grid-based planners. Section 6 provides interpretations and presents drawn conclusions regarding results presented in section 5. Finally, section 7 concludes the paper with thoughts of future work.

2. Background

2.1. Problem Definition

In our proposed algorithm, we tackle two variations of the path planning problem, namely the feasible and the optimal planning. Therefore, in this section and similarly to [7] we will formally define these two variants and provide some notations that will be used in the rest of this paper.

Let $X \subseteq \mathbb{R}^n$ be the open set that represents the state-space of the planning problem, where $n \geq 2$. Let $X_{obs} \subsetneq X$ be the obstacle region, that is consisted of the states in collision with obstacles, and $X_{free} = X/X_{obs}$ be the free space. Let $x_{start} \in X_{free}$ be the initial state, and $x_{goal} \subseteq X_{free}$ be the goal region. Since we are dealing with a 2-Dimensional state space, let $d_2(x_1, x_2)$ be the 2-D Euclidean distance between two arbitrary states $x_1, x_2 \in X^2$.
Feasible path planning is concerned with generating $\pi : [0,1] \mapsto X_{free}$, an obstacle-free path, where $\pi(0) = x_{start}$ and $\pi(1) \in X_{goal}$ if such a path exists, otherwise it returns failure.
The optimal planning in the other hand deals with the generation of a path π^* such that :
(i) $\pi^* : [0,1] \mapsto X_{free}$, where $\pi^*(0) = x_{start}$ and $\pi^*(1) \in X_{goal}$.
(ii) $C(\pi^*) = \min_{\pi \in \sum} C(\pi)$.
where C is the cost function, and $\sum$ is the set of all nontrivial paths from x_{start} to x_{goal}, however if no such path exists then it returns failure.

2.2. Related Work

The early work on path planning was generally dominated by classical methods such as: artificial potential fields (APF) [9], roadmaps, and cell decomposition [13].
The APF method proposed by Khatib in 1986 deals with the environment as a continuous state space, and use it to generate a scalar field, where the robot is attracted to the goal and repulsed from the obstacles.
Cell decomposition methods on the other hand build a graph by discretizing the continuous state-space using cell decomposition techniques, and converting the problem to a graph search problem .
Approximate cell decomposition is a class of cell decomposition, that overlays a grid with a deterministic resolution over the state space, and then generates a route in this grid by searching the embedded graph within it. Hence, this is usually known as grid-based path planning .

Grid-based path planning Among the graph search algorithms used in this approach, A* is one of the most famous methods, it finds the optimal path in a graph using a heuristic function upon which its efficiency depends heavily. Later, many variations of the A* algorithm were developed to deal with its limitations.
D*(dynamic A*) [17] allows for efficient online re-planning in partially known as well as dynamic environments. LPA*(life long A*) [11] uses heuristics to focus the replanning process when the implicit graph's topology or its edges costs change. However, these two algorithms come with high memory costs [17].
D* Lite [10] builds on LPA* but avoids the reordering of the priority queue, achieving a lower computational cost than LPA*, and guarantees an efficiency equal or greater than that of D*with a lower memory consumption than both of them. Field D* [4] uses linear interpolation during re-planning to calculate accurate path cost estimates for arbitrary positions within each grid cell, generating paths with a continuous range of headings as opposed to other A* variants mentioned above.

Stochastic methods The high memory and computation time requirements needed for grid-based planners in large search spaces inspired the development of incremental stochastic planning methods such as RRT, where random samples are drawn from a uniform distribution over the planning domain, and then used to build a search tree in the free space. However, since RRT produces only a feasible sub-optimal path, it was necessary to develop an optimal random sampling-based algorithm, and so RRT* was proposed by Karaman and Frazzoli in 2011 [8], where the concept of tree rewiring was used in order to enhance the paths extended by the tree using new samples. This algorithm is therefore considered to be asymptotically optimal since the cost of the generated paths approaches the optimum as the number of iterations approaches infinity, however, the rate of convergence to that optimal solution remained an issue.
In 2014 Gamel, Srinvansa, and Barfoot [5] presented a focused version of RRT* called informed-RRT*, they first use RRT* to find an initial path between the start pose and the goal region, this initial path is then used to calculate an ellipsoidal subset from which the new samples will be drawn instead of the full state space, focusing the search to only include this subset, which contains the states that may improve the initial path, they also proved that the informed RRT* outperforms RRT* in terms of convergence rate, final cost, and the ability to deal with narrow passages.

Learning-Based Methods recently, new learning-based algorithms were widely applied to solve path planning problems.
MPN(motion planning Networks) [16] developed by Qureshi et al, uses a neural motion planner called MPNet to plan a path from a start to a goal position directly from a provided point cloud and proved to be more efficient and consistent than the state of the art BIT* algorithm.
in [20], the authors implemented a policy-based search method that can improve planning times by learning implicit sampling distributions for particular environments, however, this method doesn't offer a solution for all types of problems, environments with narrow passages for example are still a serious challenge. [7] also proposes a method for sampling biasing, they use a conditional variational autoencoder (CVAE) to construct a nonuniform sampling distribution, but the need for a lot of preprocessed conditional information, and the fact that it is a multi-process generative model demands a great deal of time and effort for predicting the whole sampling distribution [21].

2.3. RRT*

Algorithm 1 describes the operation of the RRT* algorithm: It builds a tree rooted at the initial state x_{start}. At each iteration, a sample is drawn from a uniform distribution over X. If the new sample is located in an obstacle free location, the algorithm proceeds

Algorithm 1: $RRT^*(X, x_{start}, x_{goal})$

1: $V \leftarrow \{x_{start}\}$,$E \leftarrow \emptyset$
2: $T \leftarrow (V, E)$
3: $c_{init} \leftarrow \infty$
4: **for** $k \in \{1, ..., K\}$ **do**
5: $x_{rand} \leftarrow Sample(X)$
6: $X_{nearest} \leftarrow Nearest(T, x_{rand})$
7: $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$
8: **if** $ObstacleFree(X_{nearest}, X_{new})$ **then**
9: $V \leftarrow V \cup \{x_{new}\}$
10: $X_{near} \leftarrow Near(T, x_{new}, r_{RRT*})$
11: $x_{min} \leftarrow x_{nearest}$
12: $c_{min} \leftarrow cost(x_{min}) + d_2(x_{nearest}, x_{new})$
13: **for all** $x_{near} \in X_{near}$ **do**
14: $c_{new} \leftarrow cost(x_{near}) + d_2(x_{near}, x_{new})$
15: **if** $c_{new} < c_{min}$ **then**
16: **if** $CollisionFree(x_{near}, x_{new})$ **then**
17: $x_{min} \leftarrow x_{near}$
18: $c_{min} \leftarrow c_{new}$
19: **end if**
20: **end if**
21: **end for**
22: $E \leftarrow E \cup \{(x_{min}, x_{new})\}$
23: **for all** $x_{near} \in X_{near}$ **do**
24: $c_{near} \leftarrow cost(x_{near})$
25: $c_{new} \leftarrow cost(x_{new}) + d_2(x_{new}, x_{near})$
26: **if** $c_{new} < c_{near}$ **then**
27: **if** $CollisionFree(x_{new}, x_{near})$ **then**
28: $x_{parent} \leftarrow Parent(x_{near})$
29: $E \leftarrow E \{(x_{parent}, x_{near})\}$
30: $E \leftarrow E \cup \{(x_{new}, x_{near})\}$
31: **end if**
32: **end if**
33: **end for**
34: **if** $IsGoalState(x_{new})$ **then**
35: **return** $C_{init} \leftarrow cost(x_{new})$
36: **end if**
37: **end if**
38: **end for**
39:
40: **return** C_{init}

by extracting its nearest neighbor in the tree (line 6) $x_{nearest}$. The function steer then chooses an obstacle free configuration between these two labeled x_{new} so that it minimizes the distance $d_2(x_{nearest} - x_{new})$ while maintaining $d_2(x_{new} - x_{rand}) \leq n$ where n is a predefined value .
If the path from $x_{nearest}$ to x_{new} does not cross any obstacles, x_{new} is added to the RRT* vertex list (line 9). The algorithm then loops through the set of the near neighbors of x_{new} in the tree, and find the best parent for x_{new} from that set, a parent that provides the minimum cost to come to x_{new} from x_{start} through the tree. After that, the algorithm rewires the tree using x_{new}, by assigning it as the new parent of any node in its neighborhood, if the cost to come to that node through x_{new} is better than the cost to come through its old parent. The algorithm then keeps iterating until a termination condition is met.

In our case and at this point, we are only concerned with finding a feasible path, this means that when the RRT* tree reaches the goal region (line 34), the algorithm should be terminated. Also A maximum number of iterations K can be assigned to the RRT* loop, after which we consider the algorithm to have failed in finding the solution, and thus we return C_{init}, which in this case still have its initial value ∞ [8] [5].

3. Focused Discretization

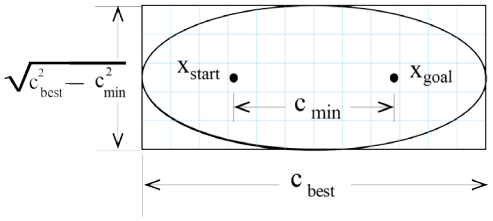


Fig. 2. This figure shows the ellipsoidal subset X_s in the case of a 2-D path planning problem calculated departing from the initial path between x_{start} and x_{goal} of cost C_{init} inside X_r which is the rectangular subset that tightly bounds X_s where its width and height are equal to X_s's diameters it is used to mark the new boundaries of the state-space to be discretized

In this paper, we are applying the concept of focused search from [8], in the context of grid-based planning, to decrease the high consumption of resources, that generally accompanies the use of grids in state-space discretization.
First, let C_{init} be the cost of a path π that connects x_{start} and x_{goal}, and let $C_x = f(x) + g(x)$ be the cost of the optimal path $\pi*$, that is constrained to include the state $x \in X$, where $f(x) = d_2(x_{start}, x)$ and $g(x) = d_2(x, x_{goal})$. Therefore, we can define a subset $X_s \subseteq X$ that consists exclusively of the states that fulfill the following inequality, $C_x \leq C_{init}$, allowing these states to be components of a path with a lower cost than C_{init}. And since the objective here is to find the optimal path, X_s can be a good basis for focusing the search process.
According to the condition above, X_s is an n-dimensional prolate hyper-ellipsoid, in the 2D case it is represented as an ellipse with x_{start} and x_{goal} as its focal points and $D_1 = C_{init}$ and $D_2 = \sqrt{C_{init}^2 - C_{min}^2}$ as its transverse and conjugate diameters, where $C_{min} = d_2(x_{start}, x_{goal})$ [5]. In order to successfully use a regular square grid for the discretization procedure we need to define another subset $X_r \subseteq X$, a hyper-rectangle that tightly bounds X_s and represents the new boundaries of the state-space as shown in figure 2. And finally X_r can be used instead of X in grid-based path planning, saving a substantial amount of resources that can be allocated to other tasks.

4. FDA*

FDA* is an implementation of the focused discretization concept discussed in the preceding section, which results in a more cost-effective grid-based planning as shown by the results presented in the next section. The pseudo-code for FDA* is shown in algorithm 2.
The procedure followed by grid-based methods can often be divided into two phases. The discretization phase where the original state-space is subdivided using a grid with an initial resolution then comes the planning phase where a graph search algorithm is used to obtain the optimal path within this grid. If no path was returned, the planner can increment the resolution and start again from phase 1. This procedure continues until either a solution is found or some other termination condition is triggered.

The only difference between FDA* and classic grid-based planning using A*is the introduction of lines 2 and 3 in algorithm 2. Where the RRT* algorithm is executed first for a few iterations until an initial path is found(line2). Its cost C_{init} is then passed to the function new_bounds as an argument, along with the original planning domain X and the x_{start} and x_{goal} configurations. This function will subsequently return the vertices of the rectangle X_r that tightly bounds the ellipsoidal subset X_s .
Following this, the new state-space is discretized (line 4), generating a square grid G_r that can be finally searched by the A* algorithm in order to compute the resolution optimal path between x_{start} and x_{goal}.

Algorithm 2: $FDA^*(X, C_{init}, x_{start}, x_{goal})$

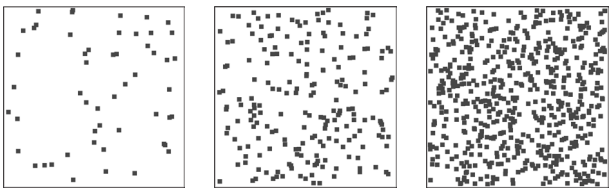
1: $L_r \leftarrow$ initialize to the original domain
2: $\boldsymbol{C_{init} \leftarrow RRT^*(X, x_{start}, x_{goal})}$
3: $\boldsymbol{L_r \leftarrow new\_bounds(X, C_{init}, x_{start}, x_{goal})}$
4: $G_r \leftarrow discretiz(X, L_r)$
5: $A^*(G_r)$

5. Results and Simulations

Initially in this section, we will present the results of an experiment designed for demonstrating the performance of our method regarding memory consumption, and then we will display the outcomes of an experimental comparison between FDA* and the classical A*, aimed for highlighting the difference in execution time between these two algorithms.

Each one of these experiments is conducted by executing a large number of Monte-Carlo simulations in environments populated with randomly generated (30x30) pixels obstacles. In addition to this, the obstacle density will be varied to create environments with different complexities (examples are shown in figure 3). It is important to note that the density levels used in our experiments do not exceed 32% since most real-life applications operate within these limits.

It should also be noted that both of these experiments are implemented with the same un-optimized code which eliminates any effects code-efficiency might have on the results of the comparison, also all experiments were run on the same 2.3GHz Intel i5 8th gen processor, 8GB memory machine.



(a) 3% **(b)** 11% **(c)** 31%

Fig. 3. Examples depicting some the of the environments used in the conducted experiments (a) a low density environment (b) a medium density environment (c) a high density environment

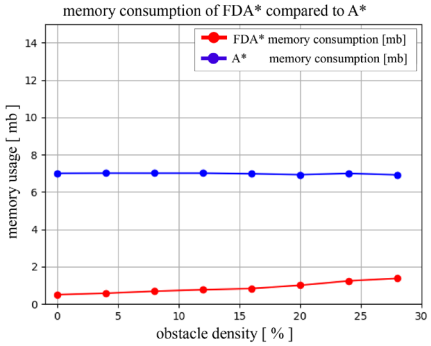


Fig. 4. Comparison between the average memory space consumed by FDA* and A* algorithms

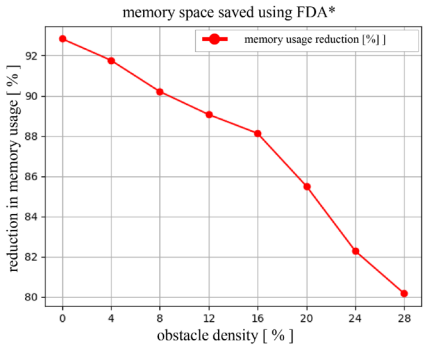


Fig. 5. The average percentage of memory space saved using FDA* instead of the classic A* algorithm

5.1. Memory Use

In this first experiment, we ran Monte-Carlo simulations for both the new proposed (FDA*) and A* algorithms in a 1200x1200 pixels map, while keeping a constant distance between the start and goal positions. Each time a simulation was executed, the memory space allocated for each algorithm was recorded, and at the end, the results were averaged and finally illustrated in figure 4.
This experiment shows a marked drop in memory consumption for the FDA* algorithm compared with A*, which is also displayed in figure 5, where their ratio is illustrated, showing an average of 92% reduction in terms of memory usage in relatively simple environments, such as environment (a) in figure 3. And even though this percentage decreases in more clustered spaces with narrow passages, more complex environments still reach an average of 83 % reduction.

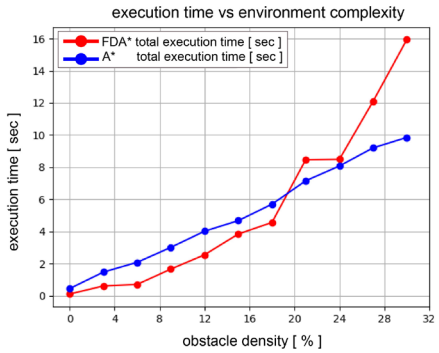


Fig. 6. Average execution time needed by FDA* and A* for finding the optimal path in 2-D environments with varying obstacle densities and a constant distance between the start and goal poses

5.2. Execution Time

In an experimental comparison, we ran both FDA* and the classical A* algorithm through the same number of simulations as the first experiment, in an environment of 600x1200 pixels and recorded then averaged the execution time for both of them.
Figure 6 shows the results of this experiment where we can clearly see that our algorithm is faster in environments where the obstacle density is less than 15 %, which is considered to be the range where most of the real-life applications operate in. However, beyond that our algorithm is out-preformed by the conventional A* path planning. Similar to the second experiment, several additional experiments were performed by varying only the size of the map. Table 1 illustrates the results where we can clearly see that the density from where our algorithm becomes slower than A* gets higher with the increase of the problem scale, giving our algorithm the upper hand in relatively large-scale problems.

6. Discussions and Conclusions

In the course of this paper, we discuss improving the expensive process of grid-based state-space discretization in single query path planning problems, through cropping the search space to only include a small subset, represented by a hyper-rectangle, in which the optimal solution can be found.
By doing so, we eliminate the states that are unnecessary for finding the optimal path and thus reduce the amount of resources dedicated to this process.

We have demonstrated in the first experiment, that focusing the discretization process can immensely decrease the memory requirements, especially in large-scale environments. The Focused Discretization technique benefits from the fact that in such environments a considerable amount of states does not contribute to solving the optimal path planning problem, and by discarding them, we save the memory space needed for storing these states.

Tab. 1. Comparison of the performance of FDA* with the classic A* algorithm regarding execution time in different scale environments

env	dimensions (pixels)	start-goal distance (pixels)	obstacle density from which FDA* becomes out-performed
1	512 x 512	300	0 %
2	700 x 700	300	16 %
3	900 x 900	300	16.50 %
4	1200 x 1200	300	19.37 %
5	1500 x 1500	300	21 %

We have further shown that this approach can reduce the overall time required to find the optimal solution in low and medium obstacle density environments, where most of the real-world applications operate. This is a result of the lower number of states (i.e.grid cells), which reduces the time needed for reading and writing the grid into memory, and thus reducing the overall execution time of grid-based planners.
Moreover, these effects are amplified in large-scale environments where even a greater number of states do not contribute to the optimal path planning.
This was made clear by comparing FDA*, to our implementation of Focused Discretization with the classical approach, using A* as a graph search algorithm for both of them. The results presented in figure 5 confirms the superiority of our technique in medium and low dense environments, this is due to the RRT* fast planning combined with the effect of the reduced number of states to be handled, resulting in the overall FDA * computation time being better than that of the conventional A*.
As the obstacles density increases, it becomes more difficult for the RRT* algorithm to find an initial path, causing our method to have an overhead in extremely clustered environments with narrow passages. However, the density in which our algorithm starts to be out-performed seems to grow higher as the problem domain grows in size, and that is caused by the higher number of states handled by the classical approach.
The path length generated by FDA* is theoretically the same as A*, due to the fact that A* is used as a planner in the second stage of FDA* after the first stage (focused discretization) is done, line 5 in algorithm 2. FDA* however, is more efficient in generating this solution. Such efficiency is the result of focusing the discretization process to a smaller subset of the state space, which leads to consuming less time and memory resources.

7. Future Work

In the future, we intend to investigate the possibility of using a learning-based technique for estimating the focused subset instead of the stochastic approach, this would immensely save the valuable computational and storage resources, increasing the efficiency of our proposed method making it more suitable for real-time applications.

AUTHORS

Mouad Boumediene – Laboratoire Automatique Skikda, Road Elhadaiek, BP.26, Skikda, Algeria, e-mail: mouadboumediene@yahoo.fr, www: http://vrpg.univ-skikda.dz/las/.
Lamine Mehennaoui* – Laboratoire Automatique Skikda, Road Elhadaiek, BP.26, Skikda, Algeria, e-mail: me_lamine@yahoo.fr, www: http://vrpg.univ-skikda.dz/las/.
Abderazzak Lachouri – Laboratoire Automatique Skikda, Road Elhadaiek, BP.26, Skikda, Algeria, e-mail: alachouri@yahoo.fr, www: http://vrpg.univ-skikda.dz/las/.
*Corresponding author

REFERENCES

[1] I. Al-Bluwi, T. Siméon, and J. Cortés, "Motion planning algorithms for molecular simulations: A survey", *Computer Science Review*, vol. 6, no. 4, 2012, 125–143, 10.1016/j.cosrev.2012.07.002.

[2] V. Bulitko, Y. Björnsson, N. R. Sturtevant, and R. Lawrence. "Real-Time Heuristic Search for Pathfinding in Video Games". In: P. A. González-Calero and M. A. Gómez-Martín, eds., *Artificial Intelligence for Computer Games*, 1–30. 2011. 10.1007/978-1-4419-8188-2_1.

[3] E. W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik*, vol. 1, no. 1, 1959, 269–271, 10.1007/BF01386390.

[4] D. Ferguson and A. Stentz, "Field D*: An Interpolation-Based Path Planner and Replanner". In: S. Thrun, R. Brooks, and H. Durrant-Whyte, eds., *Robotics Research*, Berlin, Heidelberg, 2007, 239–253, 10.1007/978-3-540-48113-3_22.

[5] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, 2997–3004, 10.1109/IROS.2014.6942976.

[6] 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, 1968, 100–107, 10.1109/TSSC.1968.300136.

[7] B. Ichter, J. Harrison, and M. Pavone, "Learning Sampling Distributions for Robot Motion Planning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, 7087–7094, 10.1109/ICRA.2018.8460730.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion plan-

ning", *The International Journal of Robotics Research*, vol. 30, no. 7, 2011, 846–894, 10.1177/0278364911406761.

[9] O. Khatib. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots". In: I. J. Cox and G. T. Wilfong, eds., *Autonomous Robot Vehicles*, 396–404. 1990. 10.1007/978-1-4613-8997-2_29.

[10] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain", *IEEE Transactions on Robotics*, vol. 21, no. 3, 2005, 354–363, 10.1109/TRO.2004.838026.

[11] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong Planning A∗", *Artificial Intelligence*, vol. 155, no. 1, 2004, 93–146, 10.1016/j.artint.2003.12.001.

[12] S. M. Lavalle. "Rapidly-Exploring Random Trees: A New Tool for Path Planning". Technical report, 1998.

[13] S. M. LaValle, *Planning Algorithms*, Cambridge University Press: Cambridge, 2006, 10.1017/CBO9780511546877.

[14] M. C. Lin, A. Sud, J. Van den Berg, R. Gayle, S. Curtis, H. Yeh, S. Guy, E. Andersen, S. Patil, J. Sewall, and D. Manocha, "Real-Time Path Planning and Navigation for Multi-agent and Crowd Simulations". In: A. Egges, A. Kamphuis, and M. Overmars, eds., *Motion in Games*, Berlin, Heidelberg, 2008, 23–32, 10.1007/978-3-540-89220-5_3.

[15] M. Otte and N. Correll, "C-FOREST: Parallel Shortest Path Planning With Superlinear Speedup", *IEEE Transactions on Robotics*, vol. 29, no. 3, 2013, 798–806, 10.1109/TRO.2013.2240176.

[16] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion Planning Networks". In: *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, 2019, 2118–2124, 10.1109/ICRA.2019.8793889.

[17] A. Stentz. "The D* Algorithm for Real-Time Planning of Optimal Traverses". Technical report, Carnegie Mellon University, 1994.

[18] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato, "Using Motion Planning to Study RNA Folding Kinetics", *Journal of Computational Biology*, vol. 12, no. 6, 2005, 862–881, 10.1089/cmb.2005.12.862.

[19] D. Thalmann and S. Raupp Musse, *Crowd Simulation*, Springer London: London, 2007, 10.1007/978-1-84628-825-8.

[20] C. Zhang, J. Huh, and D. D. Lee, "Learning Implicit Sampling Distributions for Motion Planning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, 3654–3661, 10.1109/IROS.2018.8594028.

[21] T. Zhang, J. Wang, and M. Q.-H. Meng, "Generative Adversarial Network Based Heuristics for Sampling-Based Path Planning", *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 1, 2022, 64–74, 10.1109/JAS.2021.1004275.