Analysis of Minimal Clearance and Algorithm Selection Effect on Path Planning for Autonomous Systems

Ronald Ponguillo-Intriago
Dept. of Industrial Systems Engineering and Product Design
Ghent University
Industrial Systems Engineering (ISyE), Flanders Make
Ghent, Belgium
Facultad de Ingenieria en Electricidad y Computacion
Escuela Superior Politecnica del Litoral, ESPOL
Guayaquil, Ecuador
RonaldAlberto.PonguilloIntriago@UGent.be

Payam Khazaelpour
Dept. of Industrial Systems Engineering and Product Design
Ghent University
Industrial Systems Engineering (ISyE), Flanders Make
Ghent, Belgium
Payam.Khazaelpour@UGent.be

Ignacio Querol Puchal
SEAL Aeronautica S.L.
Barcelona, Spain
IgnacioQuerolPuchal@sealaero.com

Silvio Semanjski
SEAL Aeronautica S.L.
Barcelona, Spain
Silvio.Semanjski@sealaero.com

Daniel Ochoa
Facultad de Ingenieria en Electricidad y Computacion
Escuela Superior Politecnica del Litoral, ESPOL
Guayaquil, Ecuador
dochoa@espol.edu.ec

Sidharta Gautama
Dept. of Industrial Systems Engineering and Product Design
Ghent University
Industrial Systems Engineering (ISyE), Flanders Make
Ghent, Belgium
Sidharta.Gautama@ugent.be

Ivana Semanjski
Dept. of Industrial Systems Engineering and Product Design
Ghent University
Industrial Systems Engineering (ISyE), Flanders Make
Ghent, Belgium
Ivana.Semanjski@ugent.be

***Abstract*—There are many path planning algorithms in the literature, with different classifications, domains of use, efficiency to find the shortest path or to make a complete coverage of the area to be studied. In the literature, we can also find evaluations of all these algorithms in terms of their performance in the search for the shortest path, execution time and comparisons between them. In this work, twelve algorithms from the literature were studied to analyze their sensibility to the number of obstacles and the clearance value between them. Data analytics methods were used to make a qualitative study of the sensibility of these algorithms to the constraints studied. For investigation of the problem, two metrics were used, the length of the generated path and the number of iterations used to find the solution. The number of iterations here refers to the number of nodes evaluated by the algorithm when searching for the target node. The results are synthesized in two tables that show the sensibility of the algorithms to the change in the constraints studied and the immunity of others, and the correlation among the algorithms, the constraints and the metrics.**

Keywords*—robotics path planning, data analytics, clearance analysis, autonomous systems.

I. Introduction

In Robotics Path Planning, there are many algorithms, each one with its particularity to solve a problem under a specific domain and conditions. For example, there are algorithms to discover the shortest path between two points on a map avoiding all obstacles that are on the way. There are also algorithms that do not look for the shortest path between two points, but rather find the route with which they can travel the entire map in the most efficient way, that is, without repeating visited places, or being forced to go back or perhaps generate intersections of traveled segments.

In this work, we analyze algorithms based on the response to different numbers of obstacles and clearance values. We define the clearance value to free space within the robot configuration space, limited in dimensions by the obstacle space. In Figure 1, the idea of clearance is graphically shown.

Twelve algorithms have been chosen from the literature for evaluation. These algorithms are divided into deterministic and probabilistic. The deterministic algorithms considered are A*, bidirectional A*, Breadth First Search (BFS), Bidirectional BFS, Depth First Search (DFS), Dijkstra, Greedy Best First Search, and Visibility Road Map. The probabilistic algorithms analyzed are: Rapidly Exploring Random Tree (RRT), RRT with Path Smoothing, RRT with Sobol Sampler and RRT*.

A common characteristic of all these algorithms is that

 ISBN: 978-1-61208-891-4

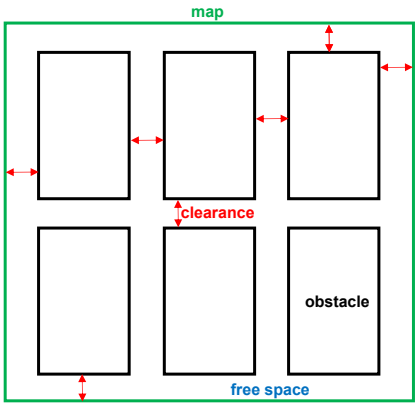


Figure. 1. Clearance graphically explained

they are algorithms that discretize the map to be traversed and create a graph on which they develop the search. All algorithms use a resolution of 1, that is, each node corresponds to a square on the grid map.

Among the deterministic algorithms is the group of (*) that use a heuristic to guide the search for the destination node. Probabilistic algorithms differ in the way they take the sample nodes or process the final path.

In Section II, a literature review is made showing a generality of each of the algorithms used, related previous works and the techniques for the evaluation of the works that were used. In the Section III, the construction of the scenarios is discussed in detail, namely, how the maps are constructed by making variations in the number of obstacles and clearance dimensions and the considerations taken into account when making changes in the position of the obstacles. The metrics used to perform the sensitivity analysis of the constraints chosen on the proposed scenarios are also defined. In Section IV, the results are shown using strip plots with the seaborn library in Python. These show the response of each algorithm to variations in the constraint's clearance and number of obstacles. The results are discussed, and a conclusion is given in Section V.

II. Literature Review

Among the path planning algorithms used in robotics there are algorithms that use a discretization of the map and convert this information into a graph that can then be traversed using different strategies to find the path between the starting node and the destination node. In this work, several of these algorithms are used that base their solution on graphs. The A* algorithm is one of the most used path planning algorithms in robotics. This algorithm was developed by Peter E. Hart et al. [8]. It combines the breadth first search technique with a heuristic to simplify the search and improve convergence times without being greedy. The Dijkstra algorithm was developed by Edsger Dijkstra [6] and is a complete algorithm that traverses the entire search space until the solution is found. Regarding the Bidirectional algorithm A* [14], it is based on A* with the variation that the route of the nodes is made in two directions, one from the start node to the goal node and one from the goal node to the initial node and ends when these two semi solutions are found. This algorithm manages to reduce the execution time to a value less than half the time used by A*. The BFS algorithm [13] [21] makes a search that goes down the levels, starting with the levels closest to the start node and moving up to the levels towards the destination node. Contrary to BFS, the Depth First Search (DFS) algorithm, as shown in [5] and [19], does the search by going through the tree branch by branch, that is, it advances through a branch and when it finishes it returns to the start node and goes through the neighboring branch, and so on until the tree is finished. Greedy Best First Search [7] uses a heuristic that tries to always predict which is the node that takes it closer to the destination node. As the last algorithm evaluated from the group of deterministic algorithms, we have the Visibility Road Map [10] [15], which is based on creating a graph, putting as nodes all the points or corners of the obstacles present on the map that are visible to the start node, goal node and among them. This created graph is much smaller than if the created graph with all the nodes of the free space is used and, therefore, it will be easier to solve. Then, to find the shortest path between the start and end nodes of this graph, any algorithm can be applied to traverse graphs, for example, Dijkstra, A*, etc.

Four probabilistic algorithms were considered, one of which is Rapidly Exploring Random Tree (RRT) [12] and the others are some variants of it. This algorithm does not go through all the nodes of the free space, but rather randomly takes a few that meet the condition of being within the defined circumference with the current node as the center and radius a number less than or equal to the expansion distance parameter. The points that coincide in the obstacle space are discarded and another random number is generated to replace it. This continues until eventually the destination node is found within the generating circle. With this methodology, by not completely covering the free space, it is possible to reduce the number of nodes traveled and, with this, the execution time. On the other hand, the price that compensates for this greater speed is that the generated path is not the smoothest possible and the length of the path obtained is not as good as what is obtained with deterministic algorithms, but it is quite close, which for many applications makes it more attractive. The other variant used, RRT Path Smoothing [3], applies the same original RRT to get the set of nodes between the start node and the goal node. Then runs a smoothing process to smooth the resulting path. This process is based on generating the least number of direct straight lines towards the destination node and eliminating the intersections that occur with obstacles. The RRT Sobol Sampler variant [11] [20], for its part, differs in the way it generates each random point in the search process for the destination node, for which it uses a technique called Sobol filter. Finally, the applied RRT* algorithm [16] uses a combination of the original RRT algorithm and a heuristic such as that of the A* algorithm, to guide the algorithm towards the destination node. This succeeds in eliminating unnecessary branches in other directions that are usually seen in the original

 ISBN: 978-1-61208-891-4

RRT.

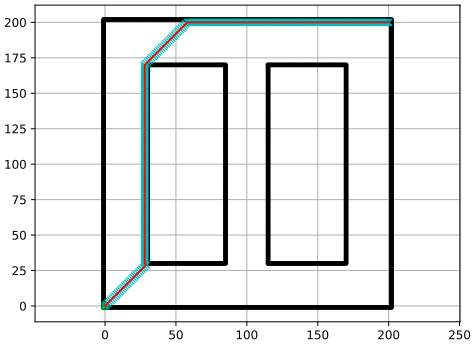
Some works in the literature that involve data analytics and robotics have been done in IoT or in robotics applied to medicine [1] [17]. These approaches use data analytics to make predictions with the data and improve their processes. The most used tools in data analytics are presented in [2] where the authors make an analysis of the leading tools in data analytics and locate the Python language in second place with a growth in terms of use in this area of more than 15% from 2015 to 2016. It also refers to the libraries used in data analytics such as pandas, scipy, matplotlib that we have also used in this work. Regarding the evaluation of path planning algorithms in robotics, we can find works that evaluate several algorithms and compare the performance between them, as in [9] where the authors make an evaluation of the performance of 5 algorithms from the literature, focusing on the length of the generated path and processing time, ranking the algorithms that obtained the best balance between both metrics as the best. In [4], the authors also analyze 5 path planning algorithms, 4 of them from the literature and one that they present in the same work. They show an analysis of the performance of these algorithms taking as metrics the length of the path obtained and the processing time. They evaluate the response of these algorithms to the variation in the size of the navigation map in terms of grid units. At the end, they implement their algorithm in ROS (Robot Operating System) [23] and make a comparison of their execution time there. In [22], the authors make an evaluation of the trajectory produced by 5 algorithms from the literature, in which they also combine the path length, processing time and curvature metrics. One of the five algorithms shown in the proposal is introduced by the authors in this work. Works that combine data analytics with the study of performance or constraints in path planning algorithms were not found and in this work an attempt is made to make that contribution to the state of the art.

III. Methodology

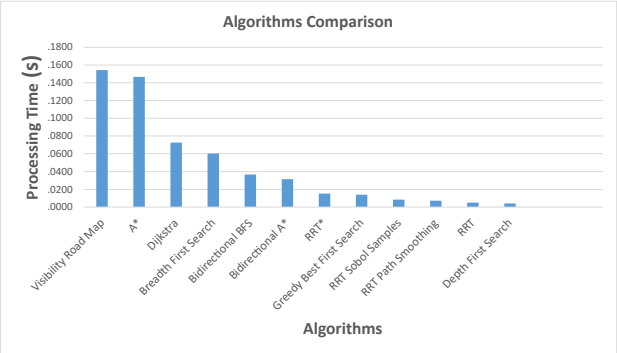
Prior to introducing the scenarios and the methodology used in this study, we want to give the reader a first impression of how different the selected algorithms are in terms of their processing time. A comparison among them is shown in Figure 2. The Figure 2(a) shows the test scenario for the algorithms. The scenario is a map with two obstacles, with the start point in the lower left corner with coordinates (0,0) and the goal point in the upper right corner with coordinates (200,200). The Figure 2(b) shows the processing time of each algorithm, measured in seconds. The simulation was run on a laptop with an Intel Core i9 processor and 32GB of RAM running the Microsoft Windows 10 operating system.

A. Scenarios

Several scenarios were built in which different number of obstacles are placed on a map of 200x200 units. These units (U) represent a way of generalizing the dimensions of the maps and can be changed to any measurement units, for example, cm, inches, feet, meters, km, etc. Maps were built with options



(a) Simple scenario to test algorithms.



(b) Processing time for algorithms under test.

Figure. 2. Processing time comparison among algorithms under study.

TABLE I
VARIANTS OF SCENARIOS BY NUMBER OF OBSTACLES

# Obstacles	Variants Scenarios
1	1
2	2
3	4
4	1
5	4
6	2
7	4
8	2

from one to eight obstacles, all of them maintaining clearance throughout the configuration space. The obstacles were moved from their position, when possible, to study if the position of the obstacles has any influence on the evaluated metrics. Thus, for the scenario with one obstacle, there are no variants since moving the obstacle from position maintaining the same clearance means rotating it and at any feasible angle of rotation it will always give the same square. Based on this criterion, in the Table I is shows the number of variants generated with the number of obstacles and their rotations.

To evaluate the effect of clearance on the metrics, the simulations were run by varying clearance values in intervals of 5 units, within the interval [5, 30].

The starting point is the origin of coordinates (0, 0) and the goal is the upper right corner with coordinates (200,

 ISBN: 978-1-61208-891-4

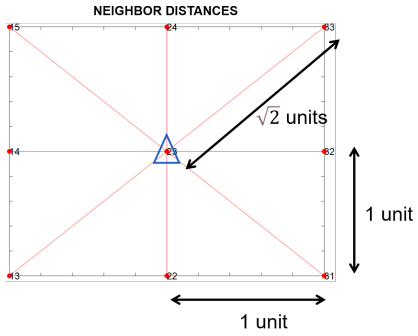


Figure. 3. Neighbor node concept illustrated

200). Obstacles are convex figures (squares and rectangles) distributed on the map in such a way that there is the same clearance throughout the map. In the resulting graph, each node is connected to its close neighbors. Close neighbors of a node are defined as those nodes that have a distance of 1 or $\sqrt{2}$ U from the original node. Figure 3 illustrates the neighbor node concept graphically.

B. Metrics

The metrics used were the length of the path generated by the algorithm and the number of iterations made by an algorithm in each scenario. The number of iterations in this work refers to the number of times that the studied algorithm accesses a node to operate on it. This metric is used since all the evaluated algorithms solve the path planning problem on a node-based map. On the other hand, this avoids using the convergence time of the algorithms, which is dependent on various factors such as the hardware, operating system or processes that run in the background on a computational platform and, therefore, makes the reproducibility of the results difficult.

C. Simulation

From the combination of 20 maps constructed by combining obstacles and clearance values according to those discussed in sub-section III-A, plus the 12 algorithms mentioned in Section I, 1440 scenarios were built on which the simulation was run to obtain the data from the Path_length and Iteration metrics that will later be used for the analysis. The simulations and data processing were done using the Python language. For the simulations, PythonRobotics [18] was used and, for the data processing, the pandas library [24] was used in the dataframe preprocessing and profiling analysis and the seaborn library [25] was used to graph the results.

IV. RESULTS AND DISCUSSION

Once the data has been processed, we can observe from the resulting graphs some behaviors of the algorithms with the chosen constraints. We will start by analyzing the effects of the clearance constraint on the Path_length metric.

Figure 4 illustrates the effect of the clearance value on the length of the path produced by each algorithm. For

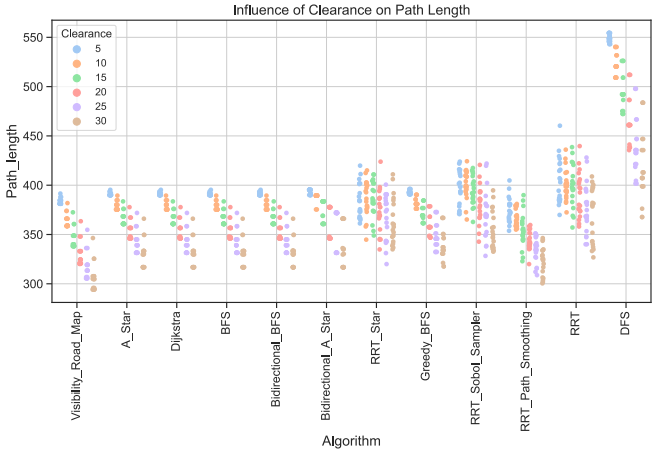


Figure. 4. Effects of the clearance over Path_length

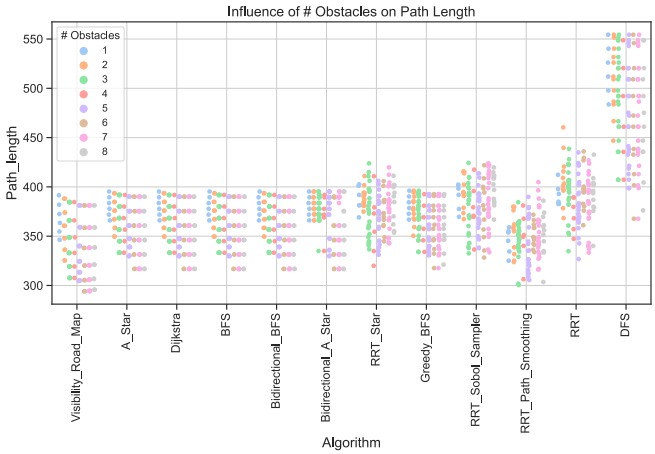


Figure. 5. Effects of the # Obstacles over Path_length

all algorithms, there is an inverse relationship between the clearance size and the length of the generated path, that is, as we increase the value of the clearance, the generated path is smaller in length. This occurs because, when we increase the clearance or, in other words, we increase the size of the free space within the configuration space, each algorithm has more options to search and establish better resulting path values.

In Figure 5, we can see that, for the algorithms A*, bidirectional A*, Bidirectional BFS, BFS, Dijkstra and Visibility Road Map, they do not show dependence on where the obstacles are located, but only on the number of obstacles and the clearance value between them. On the other hand, for the algorithms DFS, Greedy Best First Search and the group of probabilistic algorithms, there is influence of the position in which the obstacles are placed.

Reviewing Figure 6, we can see that, for the algorithms A*, Bidirectional A*, Bidirectional BFS, BFS and Dijkstra, the number of iterations increases as we increase the clearance value. This is because these algorithms, to a lesser or greater extent, sweep the available nodes when they apply their strategy to find the solution. By increasing clearance, we are

 ISBN: 978-1-61208-891-4

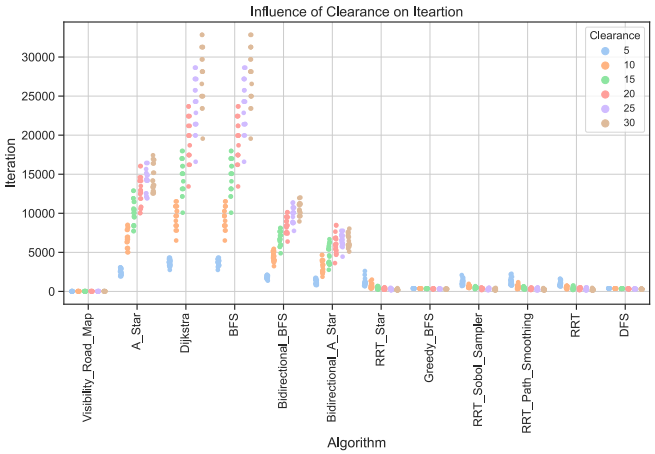


Figure. 6. Effects of the clearance over Iteration

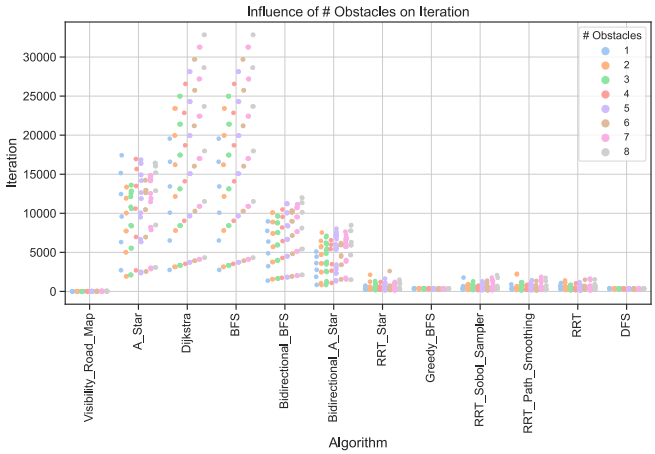


Figure. 7. Effects of the # Obstacles over Iteration

increasing the number of nodes in free space, therefore, we are increasing the number of nodes that these algorithms must evaluate. The DFS algorithm, for its part, uses a strategy that keeps the number of iterations independent of the clearance value, but pays the price by generating path lengths with values well above the rest of the algorithms evaluated. The Greedy BFS algorithm, due to its greedy nature, uses very few iterations since it only needs to have one path available between the start node and the target node. The Visibility Road Map algorithm keeps the number of iterations almost constant because, to find the solution, it creates a graph based on the number of visible points of the obstacles present and does not consider the number of nodes in the free space that we are varying with clearance.

With the probabilistic algorithms, we can notice instead that, as the clearance value increases, it becomes easier to find the solution, that is, they use fewer iterations. This is because, the narrower the corridor through which the samples must be taken, the more likely that the samples taken will overlap with the obstacle space. This sample must be discarded and a new one must be taken that matches the free space. This will make it necessary to do more iterations to be able to go through narrower paths. As can be seen, clearance has inverse effects on the number of iterations used between deterministic and probabilistic algorithms.

Regarding the effect of the number of obstacles on the metric of the number of iterations shown in Figure 7, it is well defined for the Bidirectional BFS, BFS and Dijkstra algorithms that there is no relationship with the position of the obstacles, but only the number of obstacles and the clearance value between them. On the other hand, probabilistic algorithms do show a dependency with the position of the obstacles. In case of the Visibility Road Map, when the number of obstacles increases, the number of iterations also increases. This is because of its nature of using the information of the obstacles to create the graph where the search will be done. However, there is no dependency of the clearance value on the number of iterations.

TABLE II
SUMMARY OF ALGORITHM IMMUNITY WITH THE CONSTRAINTS

	Clearance Immunity		# Obstacles Immunity	
Algorithm	**Path Length**	**Iteration**	**Path Length**	**Iteration**
Visibility Road Map	no	yes	no	no
A*	no	no	no	no
Dijkstra	no	no	no	no
BFS	no	no	no	no
Bidir BFS	no	no	no	no
Bidir A*	no	no	no	no
RRT*	no	no	no	no
Greedy Best First Search	no	no	no	no
RRT Sobol Sampler	no	no	no	no
RRT Path Smoothing	no	no	no	no
RRT	no	no	no	no
DFS	no	no	no	no

In Table II, based on the information from the four graphs in Figures 4, 5, 6 and 7, the immunity presented by the algorithms to the constraints evaluated has been summarized. As a result, we can see that Visibility Road Map is the only algorithm that really presents immunity with respect to the number of iterations versus the clearance value. This is because, regardless of where the obstacles are located or how distant they are from each other, this algorithm will create a graph with the same number of nodes and edges for the same group of obstacles if their shapes are maintained and, therefore, solving the search will always have the same number of nodes.

Table III shows a summary of the type of correlation among the constraints clearance and number of obstacles with the metrics path_length and iteration. The minus sign "-" represents a negative correlation, i.e., while one variable increase the other variable decrease. In this case, when clearance or number of obstacles increase, path_length or iteration decrease. Similarly, the plus sign "+" indicates a positive correlation, in other words, when the constraint variable increases, the metric variable also increases. The sign "x" means no correlation between variables can be defined.

V. CONCLUSION AND FUTURE WORK

In this work, the influence exerted by the clearance value and number of obstacles constraints on the generated path

TABLE III
TYPE OF CORRELATION AMONG (CLEARANCE, PATH_LENGTH, ITERATION) AND (# OBSTACLES, PATH_LENGTH, ITERATION). SIGN - IS NEGATIVE, + IS POSITIVE AND X NO CORRELATION

	Clearance		# Obstacles	
Algorithm	**Path Length**	**Iteration**	**Path Length**	**Iteration**
Visibility Road Map	-	x	-	+
A*	-	+	-	x
Dijkstra	-	+	-	+
BFS	-	+	-	+
Bidir BFS	-	+	-	+
Bidir A*	-	+	x	x
RRT*	-	-	x	x
Greedy Best First Search	-	-	-	-
RRT Sobol Sampler	-	-	x	x
RRT Path Smoothing	-	-	x	x
RRT	-	-	x	x
DFS	-	-	-	-

length and number of iterations metrics on a group of robotics path planning algorithms was explored. From the simulations carried out, and the analysis made to the data, it was possible to establish relationships between the metrics, the algorithms, and the restrictions. These results are shown qualitatively and were obtained using data analysis tools in Python language.

As an extension of this work, we intend to develop statistically validated indices that allow a quantitative approach and allow to generalize a prediction model of the behavior of the algorithms under different types of constraints.

VI. ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101007134. Also this work was in part funded by National Secretariat of Higher Education, Science, Technology and Innovation of Ecuador (SENESCYT).

REFERENCES

[1] A. Banerjee, C. Chakraborty, A. Kumar, and D. Biswas, "Emerging trends in iot and big data analytics for biomedical and health care technologies," In Handbook of data science approaches for biomedical engineering, pp. 121–152, 2020.

[2] S. Bonthu and K. H. Bindu, "Review of leading data analytics tools," International Journal of Engineering & Technology, vol. 7, no.3, pp. 10–15, 2017.

[3] X. Cao, X. Zou, C. Jia, M. Chen, and Z. Zeng, "Rrt-based path planning for an intelligent litchi-picking manipulator," Computers and electronics in agriculture, vol. 156, pp. 105–118, 2019.

[4] I. Chaari, A. Koubaa, H. Bennaceur, A. Ammar, M. Alajlan, and H. Youssef, "Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments," International Journal of Advanced Robotic Systems, vol. 14, no.2, pp. 1–15, 2017.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms," MIT press, 2009.

[6] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische mathematik, vol. 1, no.1, pp. 269–271, 1959.

[7] C. Frăsinaru and B. Răschip, "Greedy best-first search for the optimal-size sorting network problem," Procedia Computer Science, vol. 159, pp. 447–454, 2019.

[8] 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.

[9] M. Korkmaz and A. Durdu, "Comparison of optimal path planning algorithms," In 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET), pp. 255–258, IEEE, 2018.

[10] J. C. Latombe, "Robot motion planning," Springer Science & Business Media, vol. 124, 2012.

[11] S. LaValle, "Planning algorithms," Cambridge university press, 2006.

[12] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept. Oct., vol. 98, no. 11, 1988

[13] E. F. Moore, "The shortest path through a maze," In Proc. Int. Symp. Switching Theory, pp. 285–292, 1959.

[14] G. Nannicini, D. Delling, D. Schultes, and L. Liberti, "Bidirectional a* search on time-dependent road networks," Networks, vol. 59, no. 2, pp. 240–251, 2012.

[15] C. Nissoux, T. Simeon, and J.-P. Laumond, "Visibility based probabilistic roadmaps," In Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients, vol. 3, pp. 1316–1321, IEEE, 1999.

[16] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: a survey and future directions," Int. J. Adv. Comput. Sci. Appl, vol. 7, no. 11, pp. 97–107, 2016.

[17] S. Panicucci et al., "A cloud-to-edge approach to support predictive analytics in robotics industry," Electronics, vol. 9, no. 3, pp. 492, 2020.

[18] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, "Pythonrobotics: a python code collection of robotics algorithms," arXiv preprint arXiv:1808.10703, 2018

[19] A. Shojaie, A. Jauhiainen, M. Kallitsis, and G. Michailidis, "Inferring regulatory networks by combining perturbation screens and steady state gene," Plos One, vol. 9, no. 2, pp. 1–16, February 2014.

[20] I. M. Sobol, "On the distribution of points in a cube and the approximate evaluation of integrals," Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, vol. 7, no. 4, pp. 784–802, 1967.

[21] K. Ueno, T. Suzumura, N. Maruyama, K. Fujisawa, and S. Matsuoka, "Efficient breadth-first search on massively parallel and distributed memory machines," Data Science and Engineering, vol. 2, no. 1, pp. 22–35, 2017.

[22] S. Zaheer, M. Jayaraju, and T. Gulrez, "Performance analysis of path planning techniques for autonomous mobile robots," In 2015 IEEE international conference on electrical, computer and communication technologies (ICECCT), pp. 1–5, IEEE, 2015.

[23] M. Quigley et al., "ROS: an open-source Robot Operating System," ICRA workshop on open source software, vol. 3, no. 3.2 pp. 5, Kobe, Japan, 2009.

[24] W. McKinney, "pandas: a foundational Python library for data analysis and statistics," Python for high performance and scientific computing, vol. 14, no. 9, pp. 1–9, 2011.

[25] M. L. Waskom, "Seaborn: statistical data visualization," Journal of Open Source Software, vol. 6, no. 60, pp. 3021, 2021.

 ISBN: 978-1-61208-891-4