Path Planning: A 2013 Survey

(presented at the 5^{th} IESM Conference, October 2013, Rabat, Morocco)

Omar SOUISSI*, Rabie BENATITALLAH*, David DUVIVIER*, AbedlHakim ARTIBA*,
Nicolas BELANGER† and Pierre FEYZEAU†
*University of Valenciennes and Hainaut-Cambresis
LAMIH UMR 8201, 59313 France
†EUROCOPTER
Aéroport International Marseille Provence, 13700 Marignane

***Abstract*—**

In the recent decades an impressive progress was done in automation and robotic fields. Projects such as Google driverless car with "sense and avoid" capabilities or intelligent Unmanned Air Vehicle (UAV) are examples of interesting research works targeting a high degree of autonomy. In this context we underline the unavoidable requirement for improving path planning algorithms. Indeed, path planning is one of the essential tasks in the automation process of a system that moves in the environment while avoiding obstacles and respecting various constraints. In this paper we present a state-of-the-art of path planning in the field of automation, robotics and video games. Furthermore, we focus on widespread algorithms that can satisfy real-time constraints and dynamic re-planning.

I. Introduction

In the last two decades the continuous improvement in robotics and video games led to extraordinary intelligent systems. The key requirement for progress in such areas is undoubtedly the path planning. Indeed, the ability to generate an efficient path from a given initial point to a final destination in real-time conditions is still one of the biggest challenges.

The programming of an algorithm which is complete, deterministic and able to generate an acceptable path in real-time will permit to achieve a high level of autonomy. Which means for example that you will be able to read the newspaper when your car will autonomously drive you to your job.

The remainder of this paper is organized as follows. Section 2 presents the path planning problem and the state-of-the-art of the different algorithms studied in the last decades. Section 3 describes the different methods of environment modeling. Section 4 focuses on solutions adapted with real-time environment for a high level of autonomy. Finally, Section 5 summarizes the state-of-the-art presented in this paper and exposes the perspectives of our future research.

II. Path Planning

Several contributions on path planning have emerged especially from the 60s. Some of these researches have permitted to resolve several challenges in different areas such as the trajectory calculation allowing a robot to move from one point to another, computing walking movements and manipulation incorporating kinematic and dynamic constraints in humanoid robotics, studying the movements of molecules in bio-informatics, etc.

As described in Figure 1 at the highest level, we can classify the path planning problems into 3 types:

- Holonomic problems
  In robotics, a platform is called holonomic when all degrees of freedom are controllable. This ability allows to control the robot easily because all movements are possible, simplifies the problem of path planning. For instance, in the case of a robot moving on a plane, two translations and one rotation are the three degrees of freedom. From a given position, a holonomic platform will be able to move forward, to the side and turn on itself. Kinematic constraints impose a relationship between the configuration of the system and its velocity. Thus, the constraints may be written mathematically only in function of position parameters. Indeed, holonomic constraints do not involve the velocity of the moving object.
- Nonholonomic problems
  The term nonholonomic comes from the mechanic field and refers to differential constraints that cannot be fully integrated to remove time derivatives of the state variables. Many simple platforms are not holonomic. This is the case for example of cars, in which we are forced to maneuver in order to perform some trajectories such as parking operation. For nonholonomic vehicles motion planning, a popular local planning strategy is the Dubins optimal path.
- Kinodynamic problems
  The term Kinodynamic was first introduced in [1] to designate two types of constraints: Kinematic and Dynamic. Avoiding obstacles and velocity bounds are respectively examples of Kinematic and Dynamic constraints of a Kinodynamic problem.

The second level of path planning problems classification is the differentiation between two strategies used to execute the path planning. Indeed, for some algorithms like Dijkstra and A star (A*) [2], the first essential step is to model the environment before searching the optimal or feasible path, while for some other algorithms like Potential Fields [3] and RRTs (Rapidly exploring Random Trees) [4] the environment modeling is not necessary.

In the third level, we separate between the algorithms which may be efficient on-line and a second category of algorithms which may only be used off-line. In general, the on-line algorithms are those without environment modeling step.

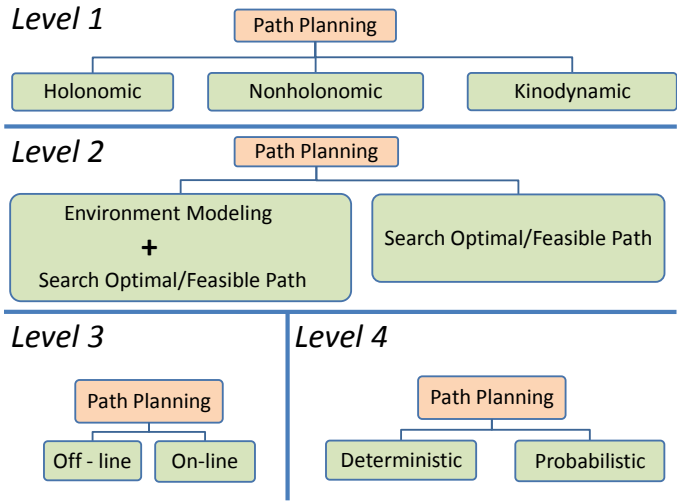

Fig. 1. Classification of path planning levels

But this is not always the case. Indeed, when we can model the environment in pre-processing step, some of the algorithms of the second category may be potential candidate for the on-line optimization.

In the fourth level, we distinguish between two categories of path planning: deterministic and probabilistic. As deterministic methods, we quote Mathematical Programming approach, Dijkstra and A* algorithms applied with grids and visibility Graphs. These deterministic methods allow to achieve the same result in each execution with the same initial conditions. Given the complexity of the algorithms cited above, these methods are most of the time not effective in practice when they are used in a real-time environment. Thus, in several works, such as [5] and [6], the authors consider a receding horizon in order to reduce the computation time. Also, other research works consider only the search of a feasible/optimal path whereas the environment modeling is achieved in a preprocessing step. But the most used solutions to overcome the problematic of real-time environment are the probabilistic methods such as Particle Swarm Optimization [7], Ant Colony [8], Probabilistic Road Mapping [9], Randomly exploring Random Trees [4] and mutlti-agent path planning [10]

III. Environment modeling

As exposed in the precedent section, the environment modeling is unavoidable step in order to execute some types of path planning algorithms. In this section we will present most of the environment modeling methods explored in the state-of-the-art. There are mainly two categories of methods: ***The first category*** are the Cell Decompositions methods, which may be done in different ways as regular and irregular grids or navigation meshes. The cells allow to represent the topology of the free space. The methods of ***the second category*** use the pre-calculation of paths between points distributed in the modeled environment as Visibility Graphs [11] or diagrams [12]. In the following paragraphs we will present the methods cited above in details:

- Regular grids:

They are the most used in environment modeling especially in robotics and video games. Indeed, regular grids have several advantages such as the ease of implementation and the simplicity of updates. Because regular grids always have the same number of nodes and edges regardless the number of obstacles that may exist. Three types of cells are explored in the state-of-the-art which are Square, Triangular and Hexagonal grids as described respectively in Figure 2-a, b and c. Also shifting cells as shown in Figure 2-e is sometimes used in order to escape from some situations in which we may be trapped between obstacles, see Figure 2-d. Even though regular grids are still the privileged environment modeling technique. It has two main drawbacks, the first is that obstacles modeling is likely to be less accurate, and the second is the memory overhead when increasing the resolution.

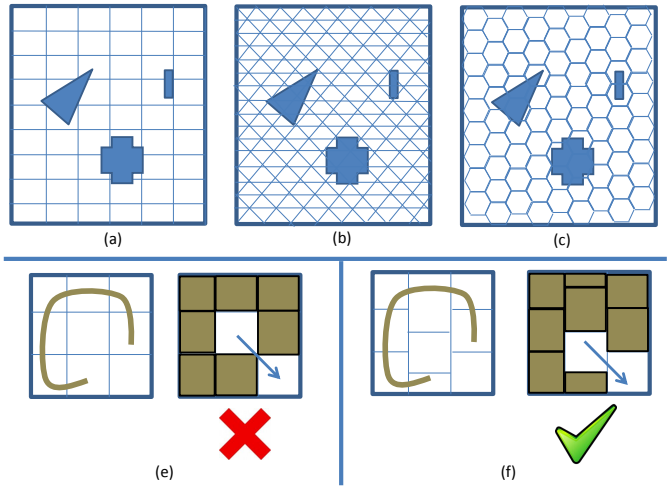

Fig. 2. Regular Grids

- Irregular grids:
In 1974, Finkel and Bentley introduced in [13] a new technique of grids realization called Quadtree. This grid is irregular since the cells do not have the same dimensions. The research about the quadtree has been strongly driven by the work of H. Samet in [14]. The use of quadtree has become common in several areas such as image processing or GIS (Geographic Information System) or 2D collision detection. The key idea of quadtree as shown in Figure 3-d is to subdivide the map into 4 squares and repeat this process for each square containing an obstacle. Hence, with an irregular grid we must obtain a roadmap with a large areas without obstacles represented by one node, and in the opposite we will have much thinner cells near obstacles. The first advantage of this method is reducing memory consumption comparing to a regular grids. Furthermore this method accelerates the graph exploration carried out by the path planning algorithm. The main disadvantage of quadtrees is that the quality of the path may be reduced in some cases. Another drawback is related with the high density of obstacles, indeed in this case, the use of quatrees loses

all meaning. An extension of quadtrees to 3D called octree is also widespread in image processing, see [15] and [16].

- Navigation mesh:
  As illustrated in Figure 3-c a navigation mesh represents the walkable areas of a map. The meshes may be represented by any type of polygon, in the given example the meshes are triangles. At first glance the visibility graph and the navigation meshes, see respectively Figure 3-a and c, may seem similar. But in practice the complexity of the navigation meshes is much less than the visibility graph one. The most often when there are used, the navigation meshes are applied for video games. More detailed explanations about these methods are given in a Phd thesis published in 2012, which compare the grids, the visibility graphs and the navigation meshes, see [11].

- Visibility graph:
  The principle of the visibility graph method as shown in Figure 3-a, is to model all the topology of the environment by linking the obstacles vertices between themselves and with the initial and final nodes. Thus, while exploring the visibility graph we may obtain the shortest possible path. Indeed, the shortest path between two nodes is either the straight line, or tangential to the obstacles. The visibility graph, therefore, provides the optimal solution when applied with Dijkstra or A* algorithms in 2D space, which make it one of the most widely environment modeling method used in path planning.
  The visibility graph technique was first introduced by Hart and Nilson [2] in 1969 and applied to the Shakey robot as reported by Lozano-Prez and Wesley [17] in 1979. Several works allowed to reduce the complexity of visibility graphs algorithms. Indeed, the first algorithm runs in $O(n^3)$ time. Several studies have been made since that, in order to reduce the complexity of the visibility graph as $O(n^2\ log(n))$ time algorithm which was presented by Lee [18] and later in 1986 by M. Sharir and A. Schorr [19]. Other works have presented faster algorithms as [20], [21] and [22].
  Despite all the efforts made to improve the visibility graph, it remains a realizable solution only in 2D space modeling environment and with an off-line resolution. Indeed as proved by Canny [23], the path planning problem in 3D is NP hard.

- Voronoï diagram:
  The first leading works on roadmaps were based on the Voronoï diagram. A comprehensive bibliography on Voronoï diagrams is given in [24]. The first informal use of Voronoï diagrams is due to Descartes in 1644, 2D and 3D cases were also studied by Dirichlet in 1850. The generalization to multiple dimensions was achieved by the mathematician Georgy Fedosievych Voronoï, which explains the method name. The literature contains several practical uses of Voronoï diagram such as [25] which aimed to optimize collision detection. The Voronoï diagram gives in a 2D polygonal environment, lines of equal distance to the polygonal obstacles. This procedure is illustrated in Figure 3-b. The lines among obstacles allow theoretically to navigate safely. The use of Voronoï diagrams is sometimes preferred in order to reduce the collisions chances, when optimality is not a critical requirement.
  Note that for navigation mesh, the visibility graph, and the Voronoï diagram, the complexity plainly relies on the number and form of obstacles and generally on the constraints of the space studied.

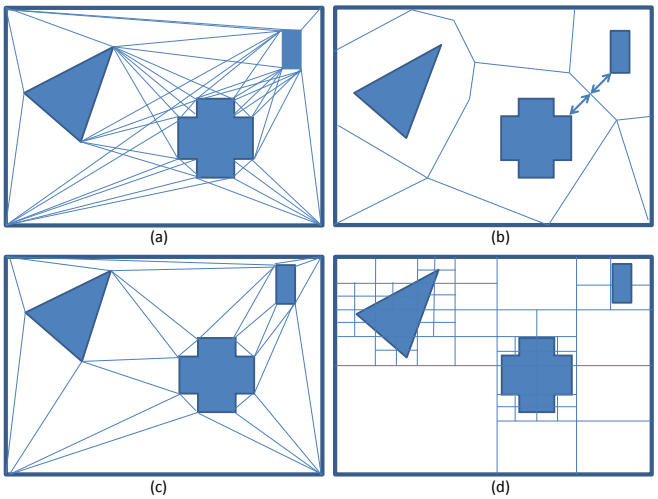

Fig. 3. Different types of roadmaps

IV. Search The Optimal/Feasible Path

Several applications and real-time systems like autonomous robots and video games lead to the increasing demand on the performance related with path planning. Also, there are still many challenges in path planning such as the capacity to deal with a high level of autonomy. The leading work in the path planning field is the one proposed by E. W. Dijkstra [26] in 1959 which exposed the way to find the path of minimum length between two nodes on a graph. Nowadays "Dijkstra's algorithm" is still one of the most well-known graph traversal algorithms.

In this section, we will introduce briefly the different types of used algorithms in path planning. Then we will delve into details for three types of algorithms: the A* with its variants, the RRTs with its variants, and finally the Potential Fields algorithm.

Ten years after the Dijkstra's algorithm, a great contribution was published by Hart and Nilson in 1969. It is the well-known A* algorithm which was used for many applications in that time, and is still nowadays one of the most used algorithms.

The main disadvantage of the basic A* algorithm is the execution time. On the contrary, the Potential Fields algorithm which was proposed in 1986 [3] is considered one of the fastest methods for path planning.

In the 80's, the variety and the number of published path planning algorithms has grown significantly. Since it were a parallel need due to the new challenges in robotic automation and video games. The common objective of a large part of the research works during the last two decades is minimizing

the execution time. Especially after the publication of Canny's article [23] in 1987 about the complexity of the path planning problematic.

Several metaheuristics are proposed in the literature such as Ant Colony [8], Simulated annealing (see [27] and [28]), Genetic Algorithms (see [29], [30] and [31]). The main advantage of these algorithms is the efficiency of the obtained solution. The main disadvantage of metaheuristics is the difficulty to reproduce the same solution since they are stochastic. Also in many cases the metaheuristics applied for path planning are unable to meet the real-time constraints.

In 1998, the most effective metaheuristic and one of the fastest methods of path planning ever proposed was published by Steven Lavalle and called "RRTs: Rapidly exploring Random Trees". The RRTs method is probabilistically complete since it guarantees to find a feasible solution when it exists, but not necessarily the optimal one.

In some fields in real-time environments, the metaheuristics like RRTs algorithms are considered as unbeatable. Indeed, they give a good trade-off between performance and acceptable quality solutions. But they are still impractical for certain industrial fields such as helicopters and UAVs where the solution should be at least, if not deterministic, bounded in terms of time and memory in order to get the certification.

In the remainder of this paper we will present in details respectively A*, RRTs and Potential Fields and their variants, since they are three among the most used methods for path planning.

A. A star and its variants

A*, pronounced as "A star", is the most well-known pathfinding algorithms. It was developed by Peter E. Hart, Nils Nilsson and Bertram Raphael in 1968 [2]. The authors extended the famous "Dijkstra's algorithm". The main principle of the A* algorithm is to reduce the number of the graph nodes explored by using a heuristic leading to the goal node. Hence, A* algorithm is generally much faster to run than Dijkstra's algorithm, and in the worst case it degenerates. The A* algorithm gives optimality when applied with the visibility graph as shown in Figure 5, while it offers an acceptable solution when applied with a grid, see Figure 6. For some fields such as avionic industry, determinism and completeness may be two indispensable properties in order to certificate the use of some algorithms. Both of these characteristics are verified by the A* algorithm independently if it is used with a visibility graph or a grid.

Given the importance of this algorithm, several researchers tried to improve the original version especially on the two last decades. In what follows, we will present some variants of the A* which are the most prominent in the literature review. As exposed in Figure 4, we may distinguish between four categories of the A* variants.

1) Dynamic variants:

- **D* (1994):** The name of the algorithm, D*, was chosen because the algorithm has the same basic construction of A* and has an additional characteristic since it is dynamic. Indeed, the idea as exposed by the author [32], is to react quickly when there is a change on the studied map, for instance an onset of a new obstacle. The main drawback of the D* algorithm is the high cost of the required memory. Also, D* has the reputation to be very complicated to understand and to implement.
- **The Focussed D* (1995):** This is an improvement of the D* algorithm, indeed it keeps the same algorithmic but reduces the computation cost. As claimed by Antony Stentz [33] which is the same author of the D* algorithm, with Focussed D* the runtime is reduced by a factor of two to three.
- **LPA*(2002):** LPA* (Lifelong Planning A*), is an incremental version of A*. On the first run, it is exactly the same as A*. But all subsequent searches are much faster since it reuses the previous search in order to reduce the number of nodes which need to be examined. Advantages of LPA* are proven by experimental results in [34]. LPA* differs from D* in that it always finds a path from the initial start point to the initial target. Consequently this approach is not used when the start point coordinates may change.
- **D* Lite (2002):** it uses the same navigation strategy as D* but is algorithmically different as stated in [35]. The authors applied their own algorithm LPA* in order to develop the D* Lite algorithm which is able to plan dynamically the shortest path from the current node to the target. D*-Lite is considered much simpler to understand and implement than D*. Also it is proven in [35] that D* Lite always runs at least as fast as D*.

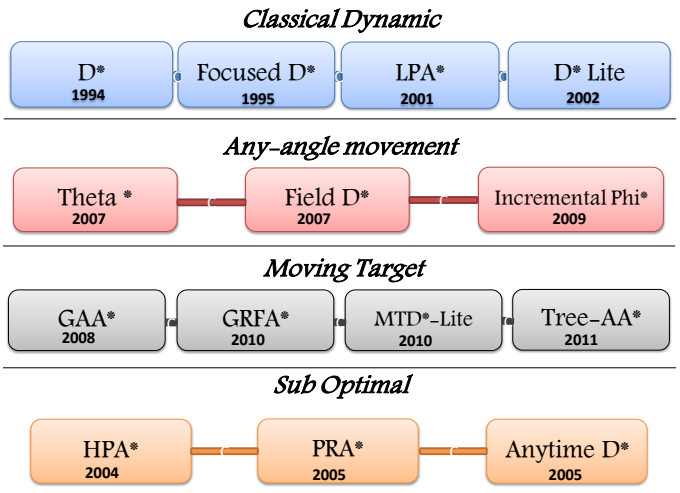

Fig. 4. A star variants

2) Any-angle movement: Note that an excellent survey of any angle path planning methods is given in the PhD dissertation of Alex Nash published in 2012 [36].

- **Field D* (2007):** A variant of D*-Lite which does not constrain movement to a grid; the best path can have the unit moving along any angle and not just 45- (or 90-) degrees between grid-points. This method was used by NASA to pathfind for the Mars rovers.

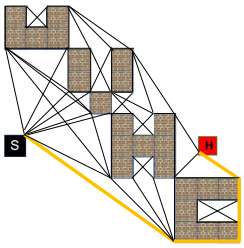

Fig. 5. Example of a A* on a Visibility Graph

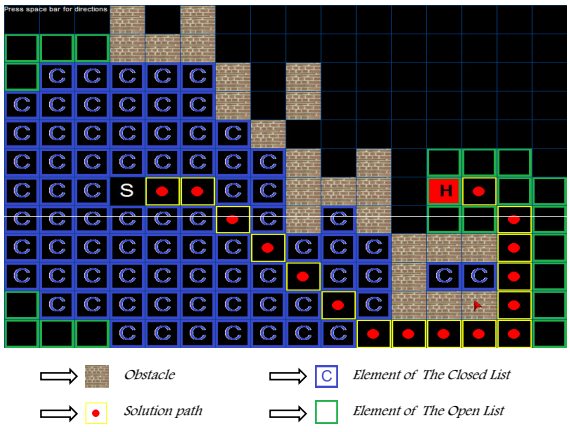

Fig. 6. Example of a A* on a regular Grid

- **Theta* (2007):** As introduced in [37], Theta* is a variant of A* that gives shorter paths than both A* and Field D*. However, theta* does not have the fast-replanning capabilities since it is based on A* unlike Field D* which is based on D*-Lite.
- **Incremental Phi* (2009):** Combines the benefits of Theta* and D* Lite. Indeed we may consider algorithm as an incremental version of Theta*, see [38].

3) Moving Target Points: Moving target search is an important issue for many robotic applications. Many works were published in the last decade in order to deal with this problem as the Generalized Adaptive A* (GAA* 2008), Moving Target D* Lite (MTD* Lite 2010), Generalized Fringe-Retrieving A* (GRFA* 2010) and Tree Adaptive A* (Tree-AA* 2011), see respectively [39], [40], [41] and [42].

4) Anytime path planning: For several case studies, the time available to take a decision is very limited. For example to get out a helicopter from a "No Fly Zone", we just need an extremely fast solution in order to avoid an accident and not specially to generate a new path to fly until the final target. In order to get a solution for similar problems a new category of A* variants was explored which we can call "Anytime path planning for suboptimal solutions", also called "Hierarchical path planning".

- **Hierarchical Path-Finding A* (HPA* 2004):** in order to satisfy to real-time constraints the authors of HPA* [43] had the clever idea to compose the general problem into hierarchical subproblems. They demonstrate that this approach is effective for reducing problem complexity in pathfinding on grid-based maps.
- **Anytime D* (2005):** as mentioned in [44], this approach combines the benefits of anytime and incremental methods in order to provide efficient solutions to complex, dynamic search problems.
- **Partial Refinement A* (PRA* 2005):** the effectiveness of this algorithm in the domain of realtime strategy (RTS) games was proven in [45].
- **Hierarchical Annotated A* (HAA* 2008):** is one of the most advanced algorithms in hierarchical path planning category. It has the property to deal with heterogeneous multi-terrain environments, see [46].

B. RRTs and its variants

The RRTs heuristic was introduced by Steven M. LaValle [4] in 1998. But it quickly became one of the most widely used methods, indeed, it proved to be very innovative and effective compared to existing methods. The RRTs method does not require an upstream environment modeling, and it has a considerable advantage in terms of computation time compared to A* and its variants which require an environment modeling. As shown in Figure 7, in a given space X with obstacles X_{obs} the RRTs strategy is to expand a tree by randomly sampling the space and growing from a starting point X_{init} until the tree is sufficiently close to a known target X_{goal}. In each iteration the tree is extended by adding a new vertex. The vertex chosen to be extended is the nearest one to the generated random vertex, see Figure 7. Once the nearest vertex is chosen we advance then with some predefined metric in the direction of the random vertex. An example of RRTs algorithm application is given in Figure 7.

Hence the RRTs based algorithms combine the environment construction phase, with the pathfinding phase. The RRTs approach allows to explore very quickly the state space which make it one of the fastest existing methods. Also RRTs is probabilistically complete. That means when a solution exist, the probability to produce it tends to "1".

The main disadvantage of the RRTs algorithms consists in their failure to stop execution and report when no possible solution exists. In the last decade several works are aiming to improve the basic version of RRTs approach.

In what follows, we expose the mainly proposed RRTs variants:

- **Bi-directional RRT:** The clever idea proposed by the authors in [47] was to launch two trees, one initialized in the start node and the other in the target. This method may be very interesting when using a multiprocessors architecture for the execution.
- **RRT*:** is an incremental sampling based algorithm which finds an initial path as the basic RRTs and later optimizes using the triangle inequality. Indeed it optimizes the connection between two non successive

GENERATE_RRT($x_{init}, K, \Delta t$)
1 $\mathcal{T}$.init(x_{init});
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 $\mathcal{T}$.add_vertex(x_{new});
8 $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
9 Return $\mathcal{T}$

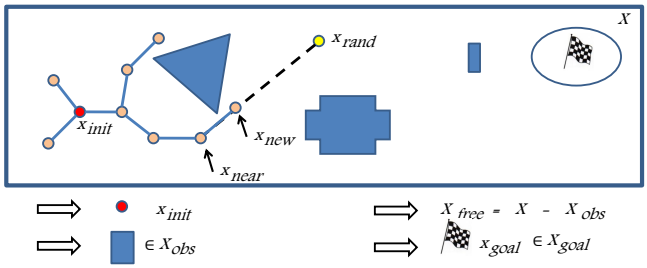

Fig. 7. RRT, Rapidly-exploring Random Trees

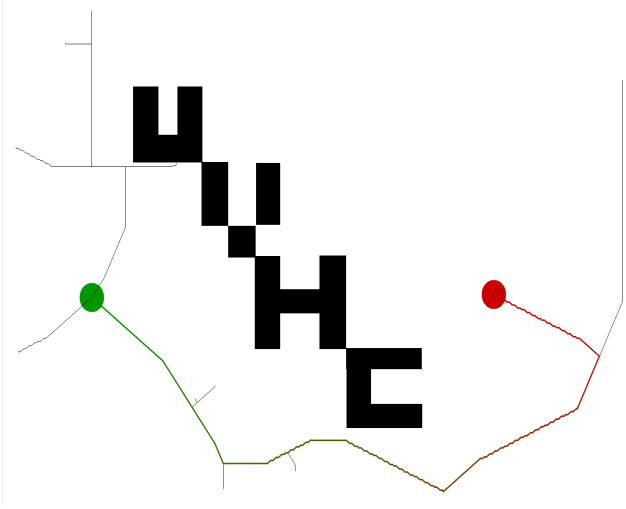

Fig. 8. Example application of the algorithm RRT

nodes. The authors of RRT* has claimed that their solution is asymptotically optimal, see [48].

- **RRT* Smart grid:** it has a similar principle as the Visibility Graph. Indeed, instead of generating a purely random exploration trees, it performs an informed exploration of the search space, and generates the random nodes in the obstacles region. As mentioned in [49], RRT*-Smart is an extension of the RRT* algorithm which aims to accelerate its rate of convergence in order to obtain an optimal or near optimal solution at a reduced execution time.

C. Potential Fields

The Potential Fields method, called also Artificial Potential Fields, was first introduced by O. Khatib in [3]. The author had the clever idea to assimilate the robot to a particle moving along the current lines of a potential created according to the target and the environment obstacles perceived by the robot. The mobile robot is immersed in potential fields (see eq. 1) which results from the superposition of an attractive potential (see eq. 2) linked to the target to be achieved, and an amount of a repellent potentials related to obstacles (see eq. 3).

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{1}$$

$$U_{att}(q) = \frac{1}{2} * k_{att} * \| q - q_{goal} \|^2 \tag{2}$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2} * k_{rep} * (\frac{1}{r(q)} - \frac{1}{r_0}) & if \ r(q) \leqslant r_0 \\ 0 & if \ r(q) \geqslant r_0 \end{cases} \tag{3}$$

The gradient of the potential U (see eq 4) provides in each point the moving direction of the robot and makes it evolve according to the decreasing gradient of the potential function. Then the algorithm iterates until the distance between the robot represented by q and the target represented by q_0 is inferior to some predefined constant noted δ in the equation 5.

$$\begin{cases} F(q) = F_{att}(q) + F_{rep}(q) \\ F(q) = \nabla U(q) \end{cases} \tag{4}$$

$$\| q - q_{goal} \| \leqslant \delta \tag{5}$$

The potential field method provides a simple technique for path planning, easy to implement and suitable for navigation with real-time requirements. However, the simplicity of the potential field method should not hide its major drawback which are the local minimum risks. Indeed the total potential influencing the mobile robot is the sum of an attractive and repulsive potentials of opposite signs. Hence, for some obstacles provision in the studied environment, this function can have a number of local minimum in which the robot will be trapped, as shown in Figure 9. Thus the potential field method can be very limited in congested environments.

Already in the late 80's, several research efforts were done in order to remove the Potential fields limitations due to the local minimum. One of the most significant works was proposed by Barraquand and Latombe in 1989, indeed, they presented and compared four techniques in order to escape the local minimum problem: "best-first motion", "random motion", "valley-guided motion" and "constrained motion".

As reported in a literature review of path planning [50] published in 2012, the potential fields method still holds interest of researchers. Indeed, it is still considered one of the best methods for on-line path planning, see [51], [52] and [53] in which some variants of the potential field method were published in the last decade.

V. Conclusion

In this paper we have presented a literature review on path planning methods. We described several methods and exposed their advantages and drawbacks. Hence, we showed the value of using each method depending on the case study considered.

We distinguished between two main categories of path planning methods. The first category involves approaches in which environment modeling is necessary. These methods are

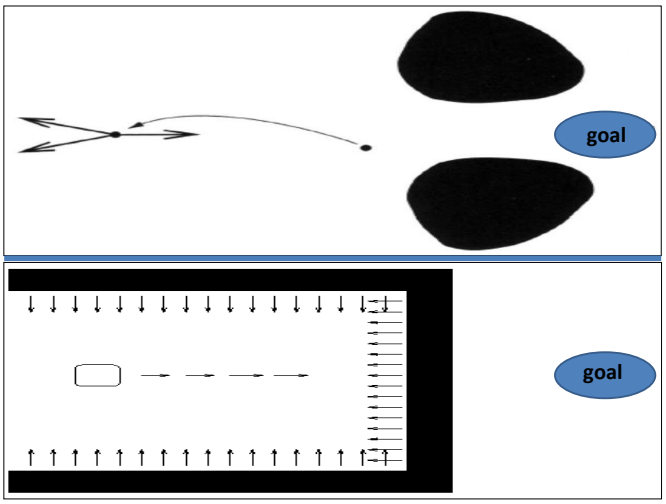

Fig. 9. Two Examples of local minima

most likely to give optimal or near optimal solutions. While the second category considers approaches in which search path may be done without modeling environment upstream. These methods are more suitable in order to deal with real-time requirements. In our study, we focused especially on three families of algorithms which are A*, RRTs, Potential Fields and their variants.

The 3D path planning problem has been shown to be NP hard [23], however some good approximation algorithms have been published in the last decade which aim to generate 3D paths as [54], [55] and [56]. But these algorithms cause a loss on performance or efficiency and consume a lot of memory. Thus they cannot be used for real-time problems.

In future works, we expect to develop a new 3D algorithm inspired from the A* family variants, and which meet the requirements of real-time constraints and dynamic path re-planning.

References

[1] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *J. ACM*, vol. 40, pp. 1048–1066, 1993.

[2] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Rob. Res.*, vol. 5, pp. 90–98, 1986.

[4] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," *Computer Science Dept. Iowa State University*, 1998.

[5] J. Bellingham, A. Richards, and J. P. How, "Receding horizon control of autonomous aerial vehicles," *American Control Conference (ACC)*, vol. 5, pp. 3741–3746, 2002.

[6] Y. Kuwata and J. How, "Three dimensional receding horizon control for uavs," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004.

[7] M. Yarmohamadi, H. H. S. Javadi, and H. Erfani, "Improvement of robot path planning using particle swarm optimization in dynamic environments with mobile obstacles and target," *Advanced Studies in Biology*, vol. 3, pp. 43–53, 2011.

[8] M. Brand, M. Masuda, N. Wehner, and X.-H. Yu, "Ant colony optimization algorithm for robot path planning," *International Conference on Computer Design and Applications (ICCDA)*, vol. 3, pp. 436–440, 2010.

[9] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

[10] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," *Algorithmic Foundations of Robotics X*, pp. 157–173, 2013.

[11] K.-H. A., *Representations for Path Finding in Planar Environments*. Aarhus Universitet, Datalogisk Institut, 2012.

[12] J. M. Kang, "Voronoi terminology," *Encyclopedia of GIS*, pp. 1241–1246, 2008.

[13] R. Finkel and J. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, pp. 1–9, 1974.

[14] H. Samet, "Using quadtrees to represent spatial data," Ph.D. dissertation, 1983.

[15] A. Knoll, "A short survey of octree volume rendering techniques," *Proceedings of 1st IRTG Workshop*, June 2006.

[16] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, pp. 129–147, 1982.

[17] T. L. Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, pp. 560–570, 1979.

[18] D. T. Lee, "Proximity and reachability in the plane," Ph.D. dissertation, 1978.

[19] M. Sharir and A. Schorr, "On shortest paths in polyhedral spaces," *SIAM Journal on Computing*, vol. 15, pp. 193–215, 1986.

[20] M. Overmars and E. Welzl, "New methods for constructing visibility graphs," *Proc. 4th ACM Symposium on Computational Geometry*, pp. 164–171, 1988.

[21] S. K. Ghosh and D. M. Mount, "An output sensitive algorithm for computing visibility graphs," *SIAM Journal on Computing*, vol. 20, pp. 888–910, 1991.

[22] S. Kapoor and S. N. Maheshwari, "Efficiently constructing the visibility graph of a simple polygon with obstacles," *SIAM Journal on Computing*, vol. 30, pp. 847–871, 2000.

[23] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pp. 49–60, 1987.

[24] A. F., "Voronoi diagramsa survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, pp. 345–405, 1991.

[25] M. L. Gavrilova and J. O. N. Rokne, "Collision detection optimization in a multi-particle system," *International Journal of Computational Geometry & Applications*, vol. 13, pp. 279–301, 2003.

[26] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[27] H. Martinez-Alfaro and S. Gomez-Garcia, "Mobile robot path planning and tracking using simulated annealing and fuzzy logic control," *Expert Systems with Applications*, vol. 15, pp. 421–429, 1998.

[28] D. P. Garga and M. Kumarb, "Optimization techniques appliedto multiple manipulators for path planning andtorque minimizationl," *Engineering Applications of Artificial Intelligence*, vol. 15, pp. 241–252, 2002.

[29] A. Elshamli, H. Abdullah, and S. Areibi, "Genetic algorithm for dynamic path planning," *Electrical and Computer Engineering*, vol. 2, pp. 677–680, 2004.

[30] H. Burchardt and R. Salomon, "Implementation of path planning using genetic algorithms on mobile robots," *IEEE World Congress on Computational Intelligence (WCCI), Congress on Evolutionary Computation (CEC)*, pp. 1831–1836, 2006.

[31] O. Castillo and L. Trujillo, "Multiple objective optimization genetic algorithms for path planning in autonomous mobile robots," *International Journal of Computers, Systems and Signals*, vol. 6, pp. 48–63, 2005.

[32] A. Stentz, "Optimal and efficient path planning for partially-known environments," *IEEE International Conference on Robotics and Automation*, pp. 3310–3317, 1994.

[33] ——, "The focussed D* algorithm for real-time replanning," *Proceedings of the 14th international joint conference on Artificial intelligence*, vol. 2, pp. 1652–1659, 1995.

[34] K. S. and L. M., "Incremental A*," *In Advances in Neural Information Processing Systems (NIPS)*, pp. 1539–1546, 2002.

[35] ——, "D* lite," *In Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, pp. 476–483, 2002.

[36] A. Nash, "Any-angle path planning," Ph.D. dissertation, 2012.

[37] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Advanced Studies in Biology*, vol. 3, pp. 43–53, 2007.

[38] A. Nash, S. Koenig, and M. Likhachev, "Incremental phi*: incremental any-angle path planning on grids," *Proceedings of the 21st international jont conference on Artifical intelligence*, pp. 1824–1830, 2009.

[39] S. K. Xiaoxun Sun and W. Yeoh, "Generalized adaptive A*," *Proceedings of 7th International Conference on Autonomous Agents*, pp. 469–476, 2008.

[40] W. Y. Xiaoxun Sun and S. Koenig, "Moving target D* lite," *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, vol. 1, pp. 67–74, 2010.

[41] ——, "Generalized fringe-retrieving A*: Faster moving target search on state lattices," *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1081–1088, 2010.

[42] X. S. Carlos Hernndez and S. Koenig, "Tree adaptive A*," *Proceedings of 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 123–130, 2011.

[43] A. Botea, M. Muller, and J. Schaeffer, "Near optimal hierarchical path-finding," *Journal of Game Development*, vol. 1, pp. 7–28, 2004.

[44] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," *In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS*, pp. 262–271, 2005.

[45] N. Sturtevant and M. Buro, "Partial pathfinding using map abstraction and refinement," *Proceedings of the 20th national conference on Artificial intelligence*, vol. 3, pp. 1392–1397, 2005.

[46] D. Harabor and A. Botea, "Hierarchical path planning for multi-size agents in heterogeneous environments," *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, pp. 258–265, 2008.

[47] S. M. Lavalle and J. J. Kuffner, "Randomized kinodynamic planning," *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 473–479, 1999.

[48] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.

[49] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "Rrt * - smart: Rapid convergence implementation of rrt * towards optimal solution," *International Conference on Mechatronics and Automation (ICMA)*, pp. 1651–1656, 2012.

[50] R. P. and P. S., "Optimal path planning of mobile robots: A review," *International Journal of Physical Sciences*, vol. 7, pp. 1314–1320, 2012.

[51] V. P, L. TH, and X. L, "Application of evolutionary artificial potential field in robot soccer system," *Proceedings of the Joint ninth IFSA World Congress and twentieth NAFIPS International Conference*, pp. 2781–2785, 2001.

[52] G.-C. Luh and W.-W. Liu, "An immunological approach to mobile robot reactive navigation," *Applied Soft Computing*, vol. 8, pp. 30–45, 2012.

[53] G. Li, A. Yamashita, H. Asama, and Y. Tamura, "An efficient improved artificial potential field based regression search method for robot path planning," *IEEE International Conference on Mechatronics and Automation*, 2012.

[54] J. Carsten, D. Ferguson, and A. Stentz, "3d field d: Improved path planning and replanning in three dimensions," *International Conference on Intelligent Robots and Systems IEEE*, pp. 3381–3386, 2006.

[55] L. Filippis, G. Guglieri, and F. Quagliotti, "Path planning strategies for uavs in 3d environments," *J. Intell. Robotics Syst.*, vol. 65, pp. 247–264, 2012.

[56] K. Y. Seng Keat Gan and S. Sukkarieh, "3d path planning for a rotary wing uav using a gaussian process occupancy map," *Australasian Conference on Robotics and Automation (ACRA)*, 2009.