A general purpose appraoch for global and local path planning combination

Luca Bombini, Alessandro Coati, Javier Sanchez Medina, Daniele Molinari, Andrea Signifredi

Abstract— **Path planning is a key and complex element for every unmanned ground vehicle development. Once the 3D reconstruction of the environment is completed and the objective configuration (desired position and pose) is defined, there has to be a careful path planning algorithm. That path is subject to many restrictions: it has to be time optimal; we have limited degrees of freedom to work with since the vehicle is a non-holonomic robot; we have limited computational power and real-time constraints regarding on-board equipments; and finally the vehicle's mechanical limitations, like the maximum curvature.**

In this paper we present a new methodology for the path planning calculation. It was meant to be a one for all methodology, useful for different scenarios (automotive, industrial applications, mining, etc.) and different platforms (car-like vehicles, forklift trucks, etc.).

This paper splits the problem in two stages. The first one faces the problem of reaching the goal with an a priori knowledge of the position affected by noise. The second approach develops a system capable of reaching the goal, enhancing the precision using a detection system, mainly based on computer vision. Particular focus is given to the interaction between the two methods proposed.

I. INTRODUCTION

In the autonomous vehicle field there is a lot of research studies regarding different path planning methods. These researches are classified according to kinematic model of the vehicles, real time processing constraints and precision of manoeuvres.

Typically, each of these algorithms are focused on solving one single problem like precision or complexity.

What we want to study in this paper is a new path planning method, that combine different approaches in order to obtain a new real-time method with good precision in applications that require it.

Our system is based on different kinds of input data: GPS position, an approximate map with fixed obstacles position, data from a detection system. This system provides information about obstacles around the vehicle and moreover it detect the target when possible.

Our work is aimed to have a single method for different kinds of scenario, like automotive, industrial applications and mining. It is also utilized for different kinds of vehicles like cars, forklift trucks or trucks. Furthermore, this project is sponsored by a big car manufacturer company. For this reason we are not able to describe some implementation details.

Sec. II covers the state of the art related to the path planning algorithms, while Sec. III and IV provide a detailed description of our global and local path planning method and their combination.

Finally, Sec. V and VI provide some performance considerations and our future developments roadmap for different applications.

II. STATE OF THE ART

There is a huge corpus of literature on autonomous robot, collision-free, time-optimal path planning. One of the first and most influential was [1] back in 1979. For most of them a high number of Degrees of Freedom (DoF) is required in the experimental platforms. Classical methods search for the mathematically proven optimal, if it exists. On the other hand, there are other traditional very widely used heuristic methods like A* ([2]), D* ([3]), Probabilistic Roadmaps ([4]), Rapidly-exploring Random Tree (RRT ([5]), potential field ([6]). There are also many examples on Genetic Algorithms, Neural Networks, Ant Colonies, Simulated Annealing, Particle Swarm Optimization and other Natural Computing optimization techniques, usually in a multi-objective fashion.

However, for this section, we will focus specifically on a particular type of robot. We will focus on non-holonomic[1] Unmanned Ground Vehicles (UGVs) path planning developments, where DoF are dramatically restricted. In other words, we will focus on automated car-like vehicles.

Path planning is approached as a two steps problem, as we have introduced in the presented methodology: first, the global planning from point A to B, to create a close to time optimal path from origin to destination, avoiding static obstacles. Second, the local planning to control the vehicle when the target is in sight, sending a sequence of feasible (drivable) control commands, making trajectories smooth and time optimal, and sometimes even avoiding smaller obstacles, avoiding less traversable patchs.

Now we will list the most frequent methodologies used for Global Path Planning and Local Path Planning.

A. Global Planning

For Global Planning, the most frequently used methodologies are as follows:

L. Bombini, A. Coati, D. Molinari, A. Signifredi are with VisLab - Department of Information Engineering, Università degli Studi di Parma, Parma, Italy{bombini, coati, dmolina, a.signi} @vislab.it, Javier Sanchez Medina is with Innovation Center for the Information Society (CICEI) - University of Las Palmas de Gran Canaria, Canary Islands, Spain, javier.sanchez.medina@ieee.org

[1] vehicles with a restricted DoF, e.g. cars. Holonomic robots are those who are assumed to have all the 6 DoF.

- A* ([7]) combines the advantages of uniform-cost and greedy searches using a fitness function. Its fitness function is basically composed of two parts, one to calculate the cost from the origin to the current position, and the second to estimate, through an heuristic, the cost from the evaluated position to the target position. Obviously, the choice of a good heuristic is essential to have an efficient and robust method.
- D*, standing for Dynamic A* Search, ([8]) is a quite similar to A* algorithm which is believed to be faster. It expands path nodes from destination. It is called dynamic because arc costs can change during the algorithm execution. In comparison to A*, D* has the completeness and optimality properties. It can also be more efficient. However, according to [9], A* seems to be more suited to passive or static environments, like it is in the case of the present work.
- Dijkstra ([10]) is a good path-finding algorithm to find high-quality paths. It only considers the cost from origin to each evaluated position. Hence a greedy search is needed, meaning it has a much higher computational cost than methods using heuristics and future cost estimations.
- Rapidly-Exploring Random Trees (RRTs) and relatives like RRT*, MARRT, etc. ([11]) . RRT is a data structure and algorithm that is designed for efficiently searching non-convex high-dimensional spaces by building a space-filling tree. Basically, it does a biased search into the largest Voronoi regions of a graph. It works by creating recurrently collision free branches of a expanding tree until it gets to the goal position.

B. Local Planning

The second step is Local Planning. The focus is on generating a smooth curve that can be run by the control system of the vehicle. This curve should connect the vehicle with the position of the target given by a detection system. For local planning there are traditionally two big categories: Functional Methods and Polynomial Approximation Methods.

1) Functional Methods: Functional methods work by combining mathematical functions to express the path. This first category can also be divided in two, as follows:

a) Closed-Form Functional Methods: These are functional methods whose coordinates have a closed-form expression, like spline based methodologies ([12]), quintic polynomials ([13]) and Bézier based curves ([14]).

According to [15] the main weaknesses of this group of methodologies are that there is no efficient way to optimize the path lengths nor the minimum radium to fit the vehicle mechanical restrictions.

b) Parametric Functional Methods: Functional Methods whose curvature is defined as a parametric curve, which is a function of their arc length. Some frequent functions used are:

- Clothoids ([16])
- Cubic spirals ([17])
- Quintic G^2-splines ([18])

This family of methodologies started with Dubins curves ([19]). The original 1957's method is quite limited, since it basically combines only two kind of functions: lines and circle segments, taking the maximum rate of change of turn of the specific vehicle. The main drawback of Dubins curves' based methods is the discontinuities that may arise between two different circle segments – where a real system would need to stop because the linear and angular velocities need to be continuous in real world. There has been a number of efforts to cancel those discontinuities, e.g. using clothoids instead of circles ([16]) or even spirals ([17]).

Clothoid pairs provide smooth transitions with continuous curvature, meaning that a non-holonomic vehicle could actually follow them. They can also be optimized to minimum-length for a given limit on jerk ([20]).

The main two drawbacks of this kind of methodologies are that it is not possible to theoretically proof completeness (with the exception of [21]) and that having no closed-form expression for position along the clothoid path means that the vehicle control needs to be built on approximations to that path and/or look-up tables. Anyway, both limitations should not be a problem with modern equipments and computers.

In the proposed methodology in this paper, our local path planning methodology is in this group. In particular, we are using the method presented in [18] because it provide a flexible tool to generate simple trajectories with the possibility to impose some contraints.

2) Polynomial Interpolation Methods: Some argue that these methods have limitations regarding providing an optimal path for a set of random points and serious scalability problems, due to the Runges phenomenon ([22]) and its computational cost.

A remarkable work in this category is [23], where a very interesting Quadratic Polynomial Interpolation is shared.

III. GLOBAL METHOD FOR PLANNING

As said before, the global method for planning aims at solving the problem of estimating a trajectory to navigate from the current position of the vehicle to a specific GPS configuration (position and orientation) of the place of interest, before that its exact position and orientation is being observed and calculated by the detection system.

We based the first part of our work on [24] because A* is one of the most commonly used search techniques in motion planning, and the variation proposed by Dolgov et al. permits to search generating feasible steering actions. This allows to generate trajectory almost ready to be actuated.

The output of the system is a trajectory that neither collides with any fixed obstacle nor goes outside the working area. It must have a maximum curvature, restricted by the vehicle's maximum feasible curvature – taking into consideration the non-holonomic nature of the vehicle. Furthermore, the trajectory has to be as short as possible for obvious reasons.

As in the implementation of Dolgov et al. [24], the input of the algorithm is an obstacle map, that contains all the

position of known obstacles. This map is implemented as a grid divided into cells. At first, each cell with a part of an obstacle on it, is set as occupied. After all the obstacles have been evaluated, we perform a morphological dilation, taking into account the space occupied by the vehicle. The dimension of the vehicle is used to expand the obstacles. After this phase we can consider the vehicle as a point.

Another input of our algorithm is the configuration that have to be reached. This data is collected via GPS for the position and with a magnetometer for the orientation. We must take into consideration that this input may be affected by noise such as drift for GPS, bias related to temperature, etc., and there may be also noise due to the fact that the data has been collected from another vehicle or by a human operator with devices that may have a slightly different calibration from the ones mounted on our vehicle.

The algorithm is a variant of the Hybrid-state A* Search proposed in the first part of the work of Dolgov et al. [24]. Each node of our implementation has only three continuous coordinates (x,y,θ) which represent position and orientation of the node in the world. We choose not to take into consideration the direction of the motion of the vehicle (forward or reverse) for several reasons. One reason is that the system must be able to respect real-time specifications without neither the use of high speed processor nor dedicated computer. Through reducing the coordinates from four to three, we reduced the computational cost of the algorithm. Another very important reason is that the vehicle has a vision obstacle detector and a laser scanner obstacle detector that are mounted on the front of the vehicle. For this reason we opt for limiting the reverse manoeuvres. We go in reverse only when we have recently seen the area where we have to go through, for example when local planner is running and we are too close to the goal and we cannot calculate a good trajectory.

For these and some other reasons we also modify the Analytic Expansion part of the algorithm described by Doglov et al. We decided to use Dubins curves ([19]) instead of the curves based on ReedShepp model. These two curves are very similar: both are segment of circle with maximum curvature and straight line. The difference is that ReedShepp curves contemplate reverse movements (e.g. a ReedShepp curve can be composed by circle to the right, reverse circle to the left and a circle to the right again, on the other hand Dubins curve are composed only by forward movements).

Another difference with Doglov's implementation of Analytic Expansion is that when the distance between the node and the goal is under a chosen threshold, we randomly try to connect the node that have to be expanded with the goal by a Dubins curve. We didn't pick any of the possible Dubins curves that connect the two positions. Instead we took out all the curves that pass on a obstacle and we also rejected the curves with circle parts that exceed $\pi/2$ each. This last condition is important, especially when we are near the goal, because matematically the positions can be reached anyway but with a long circular path that is uncomfortable and inefficient.

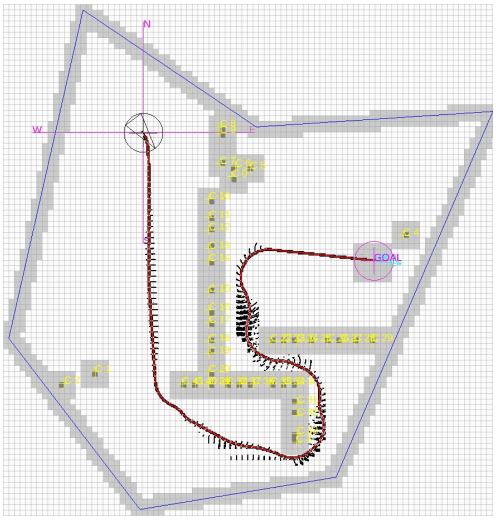

Fig. 1. Result of global planning. The background contains the grid, and the grey cells represent obstacles and perimeter. The triangle represents the vehicle. The trajectory is shown in red while the black dots show the states explored by our algorithm.

We also changed the non-holonomic-without-obstacles heuristic using the Dubins curves ([19]). We calculate online the length of the shortest Dubins curve that link the node with the goal and use that measure as the heuristic. We left unchanged the holonomic-with-obstacles heuristic.

A required feature of the trajectory is that the last segment has to be straight in order to give enough space to the local planner to calculate a new smooth trajectory after the goal is detected. This space allows the vehicle to reach the goal without going in reverse when the position and orientation error is low. This feature is required because the detection system is based mainly on a stereo system which sees only on a visual cone in front of it.

This feature is difficult to be fulfilled in the standard implementation of A* and in Hybrid-state A*. To fulfil it, our algorithm starts to expand the A* search from the goal toward the current position of the vehicle. For the first N meters of the tree, a weight is summed to the costs of each node that is not straight. The weight decreases as we get away from the first node. With this modification, the search will try to go straight for N meters and, if it can't (e.g. to avoid the collision with an obstacle), it will go straight as much as it can and then turn. This change doesn't affect the speed of convergence of the algorithm if the weights are big enough, because it works as a choice of the best node from where the normal execution starts.

We left unchanged the rule to expand the tree. From each node we can go straight, turn max curvature to the right or turn max curvature to the left. In fig. 1 we can see an example of computed trajectory.

To follow the generated trajectory we use a Pure Pursuit tracker [25]. We decided to use it because it is easy to understand and simple to implement and moreover it is robust and reliable. Furthermore its computational complexity is appropriate for real-time applications. It has proved to be able

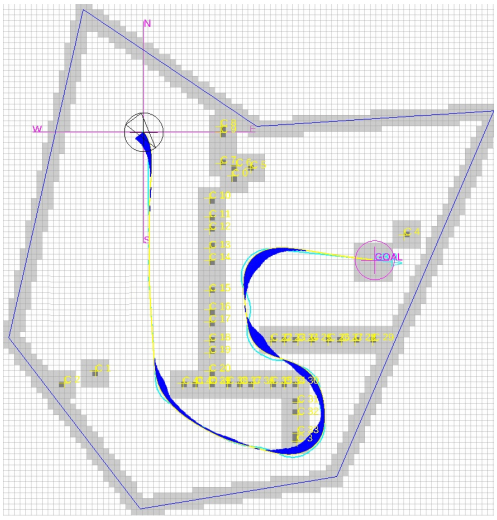

Fig. 2. Simulation of the tracking system. In yellow the simulated path and in blue a measure of the simulated curvature setpoint. The thickness is proportional to the curvature.

to follow the trajectory correctly, rejecting the error caused by position sensor errors and by delays.

As we can see in fig. 2, another effect of using a Pure Pursuit tracker is that the final path has a continuous curvature, even though our trajectory has a discontinuous curvature. This is a necessary requirement for real system because the curvature is directly proportional to the steering angle which can not change instantaneously.

In this implementation, when the vehicle finds a dynamic or unexpected obstacle, it stops and it waits that the path is freed.

IV. LOCAL METHOD FOR PLANNING

The local method for planning aims at solving the problem of estimating a feasible trajectory to navigate from the current position of the vehicle and the position and orientation of the target place observed and calculated by the detection system.

We based the second part of our work on [18] because we need a motion planning primitive with overall second-order geometric (G^2) continuity. Piazzi et al. present a method that permits the generation of trajectories that satisfy all the conditions and restrictions of our problem.

The detection system gives the relative coordinates of the goal respect to its center. The rototraslation that converts from the coordinate system centered in the detection system to the GPS system is known. First, we convert the point in GPS coordinates in order to consistently store data even if the detection system misses some frames or to cope with the navigation system working at a different frequency, which is expected. If everything works perfectly we should have a not moving point in GPS coordinates because the area detected is not moving. Obviously, this is not a realistic situation because the measure given by the detection system is subject to errors due to miscalculations or approximations. The GPS position and the orientation are also affected by the noise coming from the GPS and the magnetometer.

Taking into account these problems we implemented an algorithm that is focused on robustness, reliability and noise tolerance. Moreover, the algorithm should provide a smooth navigation through the goal.

The input of the algorithm is the GPS position of the vehicle, the current curvature and the GPS position of the goal. The system tries to compute a correct trajectory each time the vehicle moves or the goal position is updated.

The algorithm is composed by two parts: the first calculates a trajectory to the goal and the second part follows this trajectory and implements a controller to reject errors.

The first part calculates a quintic spline trajectory defined as follows:

$$\begin{cases} x = a*u^5 + b*u^4 + c*u^3 + d*u^2 + e*u + f \\ y = g*u^5 + h*u^4 + i*u^3 + l*u^2 + m*u + n \end{cases}$$

As we can see, we have a maximum of twelve constraints to be applied because the number of constrains is equal to the number of free variables in the system.

The main constraints are:

- Start position equal to current vehicle position (two constraints)
- Start orientation equal to current vehicle orientation (one constraint)
- End position equal to detected goal position (two constrains)
- End orientation equal to detected goal orientation (one constraint)
- Start curvature equal to current curvature (two constraints)
- End curvature equal to zero (two constraints)

The formula of the curvature is:

$$K = \frac{x' * y'' - y' * x''}{(x'^2 + y'^2)^{\frac{3}{2}}}$$

To set the curvature in the beginning and in the end, we calculate the first derivatives. Then we set one of the two second derivatives to zero. In this way, after having reversed the equation, we have the formula to set the curvature.

The first six constraints make the trajectory go from current position to goal considering the orientation. We forced the starting curvature to be equal to the current one to avoid any curvature discontinuity. The path must be curvature continue, in order to provide a smoother and more comfortable control. Otherwise, the vehicle should have to stop to maintain angular and linear momentum.

The curvature at the end of the trajectory is forced to zero because we want that the vehicle completes the bend before reaching the goal. In this way the curvature, as well as the angle of the turning wheels, changes continuously, reaching zero in the end.

With this configuration we have ten constraints and twelve free parameters, allowing us to vary the last two parameters to generate many feasible trajectories.Between these trajectories, we choose the one that has the best combination of these features:

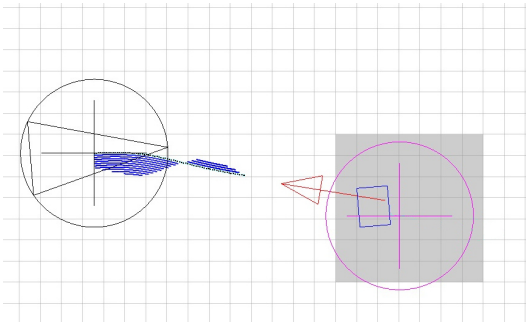

Fig. 3. Result of local planning. In black the trajectory and in blue a measure of the curvature. The thickness is proportional to the curvature. The arrow represents the output of the detection system. The trajectory ends in front of the goal point because we have to reach the goal with the front of the vehicle.

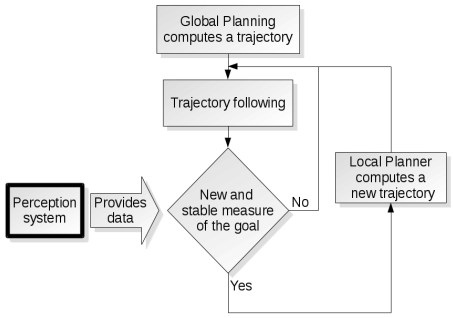

Fig. 4. Flowchart of the planning system.

- The maximum curvature less then or equal to the maximum curvature feasible by the vehicle
- It must not touch any obstacles
- Minimize the trajectory length
- Minimize the maximum derivative of the curvature

The first two features are mandatory. We reject immediately any trajectory that breaks one of these two constraints without checking the other two. Out of the surviving trajectories, we choose the one that minimizes both the length and the maximum derivative of the curvature. In fig. 3 it is possible to see an example of output.

Path following

Once provided the trajectory calculated at the previous step and a GPS position and orientation, we developed an algorithm that has as output a set-point of curvature that permits the vehicle to navigate on the trajectory or, in case the vehicle deviate from it, to get close to the trajectory.

Unfortunately, it is not possible to stay perfectly on the trajectory for multiple reasons: wheel slips, delay in the actuators, delay in the communication modules. Another source of errors is the detection system, because during the path its output may change both in position and orientation. This can be caused by bad interpretation of the object seen, or because the measure is imprecise in the distance and when the vehicle get close the measure change because it becomes more accurate.

These errors can be grouped in two effects to be corrected:

- The vehicle is not on the planned trajectory, so we want to bring it back on it
- The end of the trajectory is not on the detected point, so if we continue on it we miss the target

We calculate the combination of these errors and correct them simultaneously. The basic idea is that if the errors are both zero we can reach the goal by applying as output the curvature of the trajectory in the point where we are. We simulate the path that we will follow if we don't use any corrections. Starting from the current position, the simulator moves the vehicle by applying the curvature of the nearest point of the trajectory to the simulated position. When the nearest position is the end of the trajectory, we assume that the simulator has reached the position that the vehicle will reach without correction. We use the euclidean distance between the points just calculated and the last point given by the detection system as the error of the tracking system.

The output curvature of the path follower is the result of the combination of: the curvature of the nearest point of the trajectory as feed-forward component and the output of a PID controller that takes as input the calculated error.

TABLE I

AVERAGE OMPUTATIONAL TIMES OF EACH MODULE.

Task	Average computational time
Global Pianification	$532ms$
Pure pursuit tracker	$89\mu s$
Local Pianification	$14ms$
PID tracker	$77\mu s$

V. RESULTS

In this section we present some qualitative results.

Figure 1 and figure 2 show that the planner is able to plan a trajectory in complex environments. Moreover, it is able to follow this trajectory generating a smooth sequence of curvature set-points. During the experiments the system successfully reached the goal avoiding the fixed obstacles.

The local planner has proven to be able to reach the target with high precision. We obtained good results regarding both position and orientation. We tested a scenario where the vehicle had to reach a box shaped target with the front of the vehicle. In this way when the vehicle leans against the goal, it is possible to measure the errors. During the experiment we obtained a mean error of approximately ten centimeters in regards to position and ten degrees in regards to orientation.

As shown in figure 4, the switching between the two systems is done when the detection system provides a stable measure. When the information about the goal is provided for some consecutive cycles, the local planner try to calculate a trajectory. When the trajectory is ready, the control system seamlessly switches between global and local planner. That switching is jerk free. There are no curvature discontinuities because the starting curvature of the local planner trajectory is set equal to the last curvature actuated.

In table I are shown the average computational time of each module described before on a pc with a i7-4800MQ.

Fig. 5. Autonomous Piaggio Porter. It mounts several stereo vision systems, with different baselines, that look in different directions. It also has four laser scanner that cover the area all around the vehicle

For global pianification we considered only non-trivial configuration for the time measurement.

VI. CONCLUSION

This paper describes a new secure method to approach a target and reach it with high precision using the information provided by a detection system composed by a stereo vision system and a laser system. The paper firstly presents a global planning algorithm to go near the target and then it describes a local planning method that takes control of the vehicle when the target is detected.

Through experiments on real application, we can see that the combination algorithms proposed in this paper satisfies the requirement of real-time navigation. Furthermore it generates a G^2 control sequence. This feature becomes crucial in transporting people or dangerous goods providing high comfort and safety.

A new scenario where this algorithm will be applied is the path planning for the vehicle shown in figure 5. This is an electric vehicle equipped with sensors and actuators, already used for autonomous driving tasks. We will use the system proposed in this paper to add some interesting tasks:

- Reaching a platform to let people get inside the vehicle. For example to transport old people or people with disability.
- Reaching an autonomous battery chargeing station, where the vehicle can go and charge its batteries or replace them if needed.
- Driving the vehicle in a parking lot after being utilised. After the passengers get off the autonomous vehicle, the vehicle will autonomously park itself, if no one else need it.

REFERENCES

[1] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

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

[3] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent Unmanned Ground Vehicles*, ser. The Springer International Series in Engineering and Computer Science, M. Hebert, C. Thorpe, and A. Stentz, Eds. Springer US, 1997, vol. 388, pp. 203–220.

[4] L. E. K. P. Svestka and J.-C. L. M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces."

[5] S. M. LaValle, "Rapidly-exploring random trees a ew tool for path planning," 1998.

[6] Y. Hwang and N. Ahuja, "A potential field approach to path planning," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 1, pp. 23–32, Feb 1992.

[7] V.-D. Hoang, D. Hernandez, J. Hariyono, and K.-H. Jo, "Global path planning for unmanned ground vehicle based on road map images," in *Human System Interactions (HSI), 2014 7th International Conference on*, June 2014, pp. 82–87.

[8] A. Stentz, "The focussed dˆ* algorithm for real-time replanning," in *IJCAI*, vol. 95, 1995, pp. 1652–1659.

[9] H. J. Woo, H. J. Woo, S. B. Park, S. B. Park, J. H. Kim, and J. H. Kim, "Research of the optimal path planning methods for unmanned ground vehicle in darpa urban challenge." IEEE, 2008, pp. 586–589. [Online]. Available: www.summon.com

[10] P. Konkimalla and S. M. Lavalle, "Efficient computation of optimal navigation functions for nonholonomic planning," in *In Proc. First IEEE Intl Workshop on Robot Motion and Control*, 1999, pp. 187–192.

[11] J. Yoon and C. Crane, "Path planning for unmanned ground vehicle in urban parking area," in *Control, Automation and Systems (ICCAS), 2011 11th International Conference on*, Oct 2011, pp. 887–892.

[12] K. Komoriya and K. Tanie, "Trajectory design and control of a wheel-type mobile robot using b-spline curve," in *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, Sep 1989, pp. 398–405.

[13] A. Takahashi, T. Hongo, Y. Ninomiya, and G. Sugimoto, "Local path planning and motion control for agv in positioning," in *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, Sep 1989, pp. 392–397.

[14] J. wung Choi, R. Curry, and G. Elkaim, "Path planning based on bezier curve for autonomous ground vehicles," in *World Congress on Engineering and Computer Science 2008, WCECS '08. Advances in Electrical and Electronics Engineering - IAENG Special Edition of the*, Oct 2008, pp. 158–166.

[15] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 561–568, June 2010.

[16] T. Fraichard and A. Scheuer, "From reeds and shepp's to continuous-curvature paths," *Robotics, IEEE Transactions on*, vol. 20, no. 6, pp. 1025–1035, Dec 2004.

[17] Y. Kanayama and B. I. Hartman, "Smooth local path planning for autonomous vehicles," 1989, pp. 1265–1270 vol.3. [Online]. Available: www.summon.com

[18] A. Piazzi, C. G. Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic g2-splines for the iterative steering of vision-based autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 27–36, 2002. [Online]. Available: www.summon.com

[19] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, pp. 497–516, 1957.

[20] W. Nelson, "Continuous-curvature paths for autonomous vehicles," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, May 1989, pp. 1260–1264 vol.3.

[21] A. Scheuer and T. Fraichard, "Collision-free and continuous-curvature path planning for car-like robots," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 1, Apr 1997, pp. 867–873 vol.1.

[22] J. F. Epperson, "On the runge example," *Amer. Math. Monthly*, vol. 94, no. 4, pp. 329–341, 1987.

[23] U.-Y. Huh and S.-R. Chang, "A g2 continuous path-smoothing algorithm using modified quadratic polynomial interpolation," *International Journal of Advanced Robotic Systems*, vol. 11, no. 1, 2014. [Online]. Available: www.summon.com

[24] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010. [Online]. Available: www.summon.com

[25] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, January 1992.