PWL Approximation for Dense Mapping and Associated Hybrid PSO-Dijkstra Processes for Path Planning

Karime Pereida, Jose Guivant and Anton Lohr
University of New South Wales, Australia
{k.pereidaperez@student.unsw, j.guivant@unsw, anton@unsw}.edu.au

Abstract

This work proposes and illustrates an efficient path planning algorithm for nonholonomic platforms in dense contexts. The aim is to compute optimal paths in a map described by dense properties and including the robot's constraints. The proposed algorithm integrates a low dimensional Dijsktra cost-to-go function, calculated over a QT-PWL approximation, with Particle Swarm Optimisation (PSO) minimisation, performed in the higher dimensional configuration space. The methodology includes adaptations to reduce the processing cost and increase the optimality of the planned path. Experimental results show the performance of the algorithm.

1 Introduction

Planning is an essential capability for autonomous robots. The goal is to plan a trajectory that joins an initial configuration to a goal configuration in an optimal manner according to certain criteria, such as time to travel and distance to travel. For many applications it is important that the platform describes the perceived environment in terms of dense properties as the cost of operation depends on both, the control actions and the properties of the context of operation. In recent years, such applications are becoming important as robots are being introduced to less controlled and dynamic environments. Dense properties, in applications such as route planning, are inherent to the terrain and are usually the result of a mapping or learning process, even a SLAM process such as DenseSLAM [Nieto *et al.*,], [Nieto *et al.*, 2006]. The values of such dense properties are expressed as a function of 2D geographical coordinates. Efficient data structures are needed to increase the speed of algorithms that work with dense contexts.

Few path planning approaches, for example [Gennery,], have addressed the problem of including the traversability of the terrain as part of the planning process. An efficient planner that specifies the control inputs that will take the robot from an initial configuration to a goal configuration in a dense environment while including the constraints of the platform is needed. There are two main difficulties in this approach. First, the admissible motion is constrained by obstacles, holonomic constraints and nonholonomcic constraints imposed by the kinematics of the system. The constraints in the feasible configurations thin the feasible subspace in some directions to a lower-dimensional manifold or variety. Second, the use of dense contexts to plan as the environment is represented in a detailed manner that describes situations that are more or less desirable for the platform. In the case of a ground vehicle these situations could be exemplified by paved areas (more desirable areas) and grassy areas (less desirable areas).

The approach in this work proposes a set of adaptations to the environment to improve the processing cost of the algorithm. It is a hybrid approach that combines dynamic programming and Particle Swarm optimisation in order to plan for a platform in a dense context. The paper is organised as follows. In Section 2 the related research is presented. Section 3 describes the proposed algorithm. Section 4 shows the results obtained with various platforms. Finally, Section 5 presents the conclusions and future work.

2 Related Research

The Dijkstra algorithm is one of the most efficient implementations for solving certain Dynamic Programming optimisation processes. The result of the Dijkstra algorithm is the computation of the shortest path for a given source s to a destination d based on a directed graph with non-negative edge weights [Bellman, 1957]. The complexity of the Dijkstra algorithm for vertex-based cost functions is $O(|E| + |V|log|V|)$ using a binary heap implementation for the priority queue [Barbehenn, 1998]. Further efficiency can be provided if certain characteristics of the valid transition are exploited as is the case of

the Untidy Priority Queue [Yatziv *et al.*, 2006] and the Pseudo Priority Queue [Robledo *et al.*,].

Nonholonomic planning techniques have been a rich area of research. This topic was introduced by [Laumond, 1987]. Nonholonomic systems are more complex as they introduce inequality constraints to the system; hence, reducing the set of achievable velocities. In the present research it is of interest to plan for a ground vehicle, this problem becomes more complex with a platform such as the tractor-trailer system. The tractor-trailer kinematics and constraints have been studied in [Yan *et al.*,]. A common approach to nonholonomic path planning is to use a holonomic planed path and modify the trajectory to satisfy the nonholonomcic constraints. This approach is presented in [Sekhavat *et al.*, 1998] where two techniques for these transformations are defined.

The DenseSLAM algorithm obtains and maintains detailed environment representations, i.e. it is the process of performing SLAM (Simultaneous Localisation and Mapping) while obtaining a dense environment representation. This process was introduced by [Nieto *et al.*, 2006] where different sensory information is presented in dense multi-layered maps. Each layer represents different properties of the environment that could be occupancy, traversability, elevation, among others or each layer can describe the same property using different representations. The HYbrid metric maps were introduces as a form of DenseSLAM representation by [Nieto *et al.*,]. Its framework allows he combination of efficient feature-based SLAM algorithms with occupancy grid maps among other representations. The importance of a rich representation lies in its potential to assist in the navigation process.

3 Description of the proposed approach

3.1 Map adaptation

The present research is aimed at planning in dense contexts where the workspace is a 2D grid map of the environment that assigns costs of traversability to each cell. Nontraversable obstacles, such as walls, are assigned the highest (infinite) cost-to-traverse; areas that are difficult to traverse, such as grassy areas, are assigned a high (but finite) cost-to-traverse; and areas that are easy to traverse, such as paved areas, are assigned a low cost-to-traverse. This research is focused on planning for a car-like vehicle and for a tractor pulling trailers. Figure 1 shows a synthetic example of an environment described by dense contexts. Dark red areas represent nontraversable obstacles with infinite cost-to-traverse; green areas have a high cost-to-traverse; and white areas have a low cost-to-traverse.

The sensory information provided by the platform needs to be adapted in order to include the various constraints and requirements of the planner. The first adaptation is the expansion of the nontraversable obstacles. This adaptation imposes the platform a high cost to traverse areas near nontraversable obstacles. The amount by which the obstacles are expanded is determined by the geometry of the platform. Figure 2 shows the enlargement of the nontraversable obstacles in the synthetic example presented before.

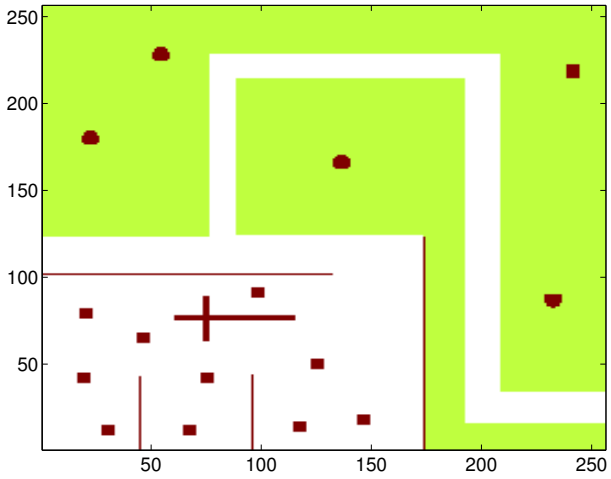


Figure 1: Synthetic map with dense properties. Red areas are nontraversable obstacles that have an infinite cost-to-traverse; green areas have a high cost-to-traverse; and white areas have a low cost-to-traverse.

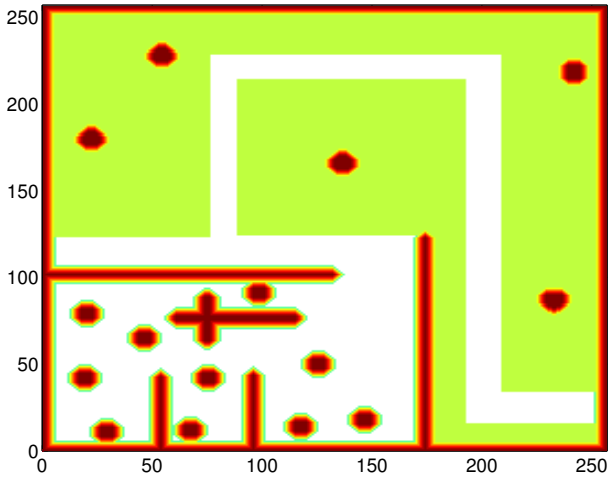


Figure 2: Synthetic map with dense properties with expanded nontraversable obstacles added to prevent the planner from colliding with a nontraversable obstacle.

The present research aims to plan a path for a robot that operates in a dense context. A Dijkstra algorithm is used to compute the cost-to-go function of the environment as it takes into account the dense context. Furthermore, the robot operates in a 2D context, but is

required to navigate from an initial to a final configuration specified by (x, y, ϕ). Adding one more constraint (e.g. heading) to the problem could be solved in different manners, such as increasing the dimensionality of the configuration space; however, this significantly increases the processing cost of the Dijkstra algorithm. In [Pereida and Guivant, 2013] the virtual alley is introduced to be able to specify the heading constraint in a 2D context. This adaptation is used in the present work as it creates a significant global minimum that aids the associated Particle Swarm Optimisation (PSO) process to converge to the final configuration.

Numerous applications are characterised by a workspace with sizeable areas of similar or identical dense attributes, e.g. free space. Certain regions of dense properties present values that can be approximated by a lower number of patches of a Piece Wise Constant (PWC) or a Piece Wise Linear (PWL) representation. This work utilises a PWL approximation to describe the terrain and other dense properties in an efficient manner. The PWL approximation for a 2D function is defined as:

$$|\hat{f}(u,v) - f(u,v)| < \tau \quad \forall (u,v) \in \Omega \tag{1}$$

where the function $f(u,v)$ is evaluated in a domain Ω and can be approximated by a Piece Wise Multi Linear function $\hat{f}(u,v)$. The approximating function is,

$$\begin{aligned} &\hat{f}(u,v) = a_k + b_k \cdot u + c_k \cdot v \; \forall (u,v) \in \Omega_k \\ &\{\Omega_k\}_{k=1}^{N} \setminus \cup_{k=1}^{N} \Omega_k = \Omega \end{aligned} \tag{2}$$

The threshold τ determines the granularity of the approximation. A higher threshold would allow the existence of cells presenting larger discrepancies between the approximating function and the real dense property (generating larger patches). In order to keep the PWL approximation realistic, the threshold τ should remain in a working range. To simplify the run-time processing requirements of synthesising the approximating function, a less expensive version called QuadTree PWL (QT-PWL) is applied. The QT-PWL has a partition of the function's domain that is constrained to respect a QuadTree structure [Guivant *et al.*, 2012]. Figure 3 depicts the differences in the resulting PWL approximation with a different τ. Figure 3a shows a typical area of a 2D grid map. Figure 3b depicts the result of the QT-PWL approximation with a small threshold; thus, yielding more patches with a close representation of the original map. Figure 3c shows the result for a larger threshold where a coarser representation is obtained.

In order to illustrate the map adaptations described, the goal configuration is set to $(34, 23, 0)$. Figure 4 depicts the result of adapting the synthetic map presented earlier. Nontraversable obstacles are expanded, the virtual alley adaptation is included and the QT-PWL approximation is applied. In this case the contours of patches size 4 and bigger are shown.

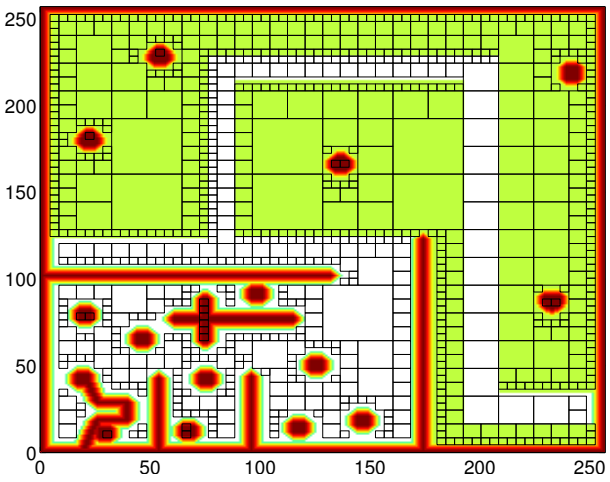


Figure 4: QT-PWL approximation of the synthetic map with dense properties. Areas with identical or similar properties are grouped in patches. The contour of patches size 4 and bigger are shown. Most of the area is covered by big patches.

The Dijkstra algorithm is applied over the adapted environment. The first task is to locate the QT-PWL leaf start patch S, which is the region of the workspace that contains the specified start point. The cost-to-go function f is then calculated for every patch c as:

$$f(c) = g(p) + \tilde{g}(p,c) \tag{3}$$

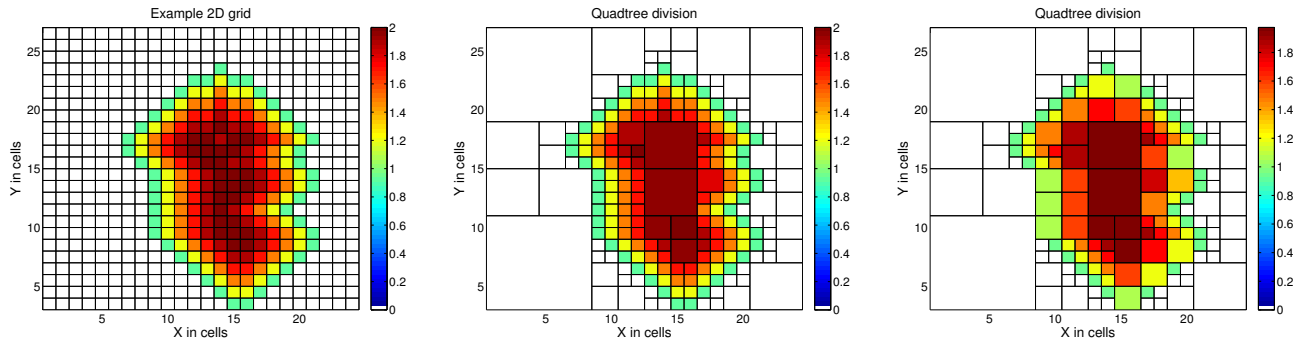
where $g(p)$ is the cost of the path from S to p, which is the predecessor of c, and $\tilde{g}(p,c)$ is the cost of the path segment between p and c. It is defined as

$$\tilde{g}(p,c) = D(p,c) + \alpha \cdot d(c) \tag{4}$$

where $D(p,c)$ represents the distance between nodes p and c; $d(c)$ represents the cost of including node c in the path, i.e. the associated dense property; and α is a weight related to the effect of the dense properties in the cost-to-go function. This cost takes into account the QT-PWL approximation cost associated with nodes p and c. The Pseudo-Priority Queue [Robledo *et al.*,] is utilised to reduce the processing cost of the implementation.

3.2 Minimisation process

Once the map has been properly adapted to include some of the constraints of the planner, the minimisation process is applied in order to find a suitable path to the destination. To achieve this result the Particle Swarm Optimisation (PSO) algorithm is implemented. The PSO



(a) (b) (c)

Figure 3: Specific area zoomed for comparison of the division process by the PWL QuadTree: (a) shows the original 2D grid map; (b) shows the QuadTree partition created with a small threshold to divide; and, (c) shows the QuadTree partition created with a large threshold to divide.

is initialised with a number (e.g. 100) two-dimensional particles. The particles specify values for the inputs of the system, in this case, speed and steering (v, ϕ) as the tractors are assumed to have no steering. These values are also constrained to lie in the set of achievable inputs, i.e. $v \in [0, v_{max}]$ and $\phi \in [\phi_{min}, \phi max]$. In this work the particles are initialised to two possible initial speeds (50% and 100% of v_{max}), and steering spanning evenly through the range of achievable values. With these values, the known initial configuration and the kinematic model, it is possible to calculate the next configurations of the system.

The set of states obtained by the application of the proposed input controls are used to calculate their corresponding fitness value. The fitness value is a metric that determines the best control function that minimises the desired parameters. In this work the fitness function aims to find a short path while avoiding steep turns and sudden changes in speed. Because the environment is represented by patches of different sizes, the cost-to-go $g(x, y)$ is obtained using,

$$g(x, y) = c_b(x, y) + \frac{D(b, p) * (c_c(x, y) - c_b(x, y))}{D(b, c)} \quad (5)$$

where $c_b(x, y)$ is the cost of the neighbouring patch b (which has the lowest cost-to-go) to the patch where the vehicle is c; $D(b, p)$ is the distance between the current position of the vehicle p and the neighbouring patch with the lowest cost-to-go b; $c_c(x, y)$ is the cost-to-go of the patch where the vehicle is c; and $D(b, c)$ is the distance between current patch c and its neighbouring patch with the lowest cost-to-go b. This approach incorporates the size and cost of the patches involved in the calculation of the cost of the state $g(x, y)$.

Each approximated patch $p(x, y)$ has a cost-to-go $p_g(x, y)$ referred to the centre of such patch assigned by the Dijkstra algorithm. Furthermore, each patch has a set of neighbours. The neighbouring patch with the smallest cost-to-go is chosen to interpolate the cost. Interpolation takes into account the cost-to-go of the current patch (the one that includes the x,y coordinate of the current pose), the cost-to-go of its neighbour with the lowest cost-to-go and the cost of transition.

4 Results

4.1 Synthetic example

The Dijsktra algorithm is applied to the adapted map to obtain the cost-to-go function. Figure 5 depicts the result of computing the cost-to-go function over the approximated map with the described adaptations. It is clear that the cost-to-go increases rapidly in areas with a high cost-to-traverse. This is noticed by the rapid change in colour. Areas with similar or identical dense attributes have been grouped in larger patches, which reduces the processing cost of the Dijkstra algorithm. Table 1 shows the number of patches in the uniform grid and the QT-PWL approximation. The reduction in the number of patches is significant, the QT-PWL approximation has less than 20% of the number of patches that patches that the regular grid has. Moreover, each patch has the same number of neighbours, on average. Table 2 shows detailed information on the number of patches in the QT-PWL approximation. It is clear that most of the area is covered by large patches; more than 50% of the area is covered by patches size 8 and bigger which reduces the processing cost of the Dijkstra algorithm.

In this example the initial configuration is $(x, y, \phi) = (200, 200, \pi/2)$ and the goal configuration is $(34, 23, 0)$. Figure 6 depicts the planned path for this platform. The shortest path, from the initial configuration traverses the high cost-to-traverse area. Because the planner is able to

Parameter	Uniform Grid	Quadtree-PWL
Number of cells	65,536	11,836
Average number of neighbours	12	11.99
Time to compute (ms)	98	23

Table 1: Comparison of different parameters on the normal grid and the QT-PWL approximation. It is clear that the QT-PWL approximation has a much better performance than the normal grid representation.

Patch size	Number of patches	% of area covered
1 × 1	9,220	14.07 %
2 × 2	1,635	9.98 %
4 × 4	651	15.89 %
8 × 8	263	25.68 %
16 × 16	60	23.44 %
32 × 32	7	10.94 %

Table 2: Patch size and percentage of the total area covered by patches of the specified size.

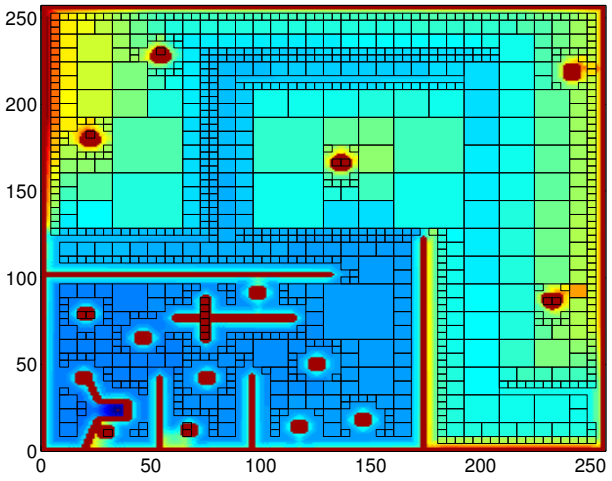


Figure 5: Dijkstra algorithm over QT-PWL approximation of the synthetic map with dense properties. The cost-to-go function agrees with the dense properties. The cost-to-go increases at a lower rate over the low cost-to-traverse areas. Goal position is at $(34, 23)$.

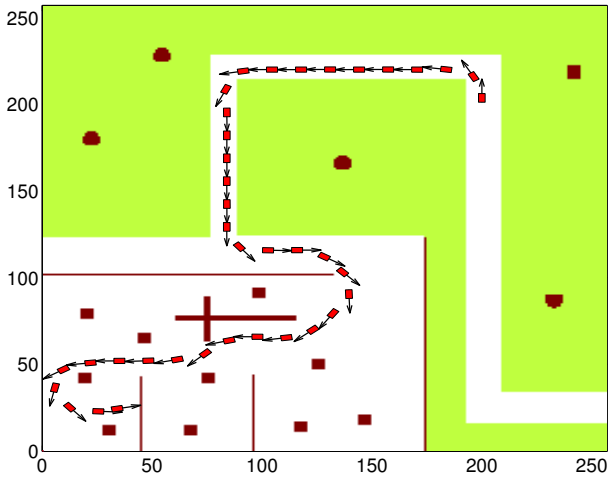


Figure 6: Planned path for a car with initial position at (200,200,$\pi/2$) and goal position at (34,23,0). The planner computes a longer path in order to avoid traversing the high cost-to-traverse area. The platform is able to include the heading constraint at the destination.

consider the dense areas, the platform traverses a longer path but remains in the low cost-to-traverse area. The cost-to-go function synthesised by the Dijkstra algorithm creates a global minimum that favours the goal heading. The PSO includes this knowledge from the cost-to-go function as the platform traverses a longer path in order to achieve the desired heading.

The case with trailers

The algorithm was tested in a more complex platform, a tractor pulling two trailers. The addition of the two trailers increases by two the number of degrees of freedom in the platform $(x, y, \phi_0, \phi_1, \phi_2)$. These degrees of freedom are controlled by the speed and steering of the tractor only. This platform represents more challenges as the number of achievable configurations decreases. In this example the final configuration is kept, assigning the trailers the same heading as the tractor, while the initial configuration changes to (50, 200, $\pi/2$, $\pi/2$, $pi/2$). Figure 7 depicts the planned path for this platform. The platform starts in a high cost-to-traverse area and quickly turns to remain in a low cost-to-traverse area. The path taken by the platform is not the shortest path, but it is the one that traverses most of the path over a low cost-to-traverse area. The platform takes a longer

path in order to achieve the desired final heading.

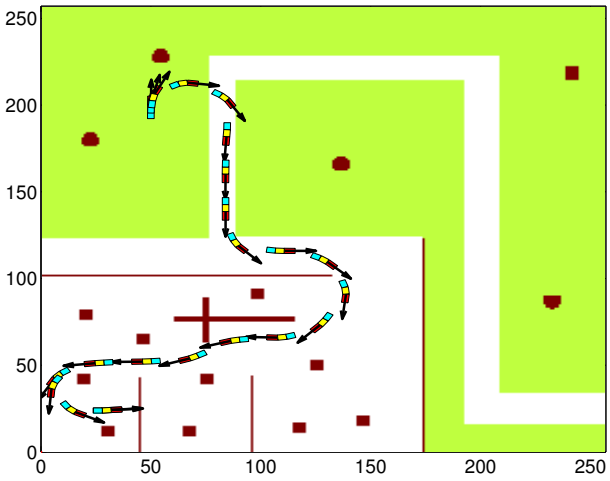


Figure 7: Planned path for a car pulling two trailers with no steering with initial position at (500,200,$\pi/2$,$\pi/2$,$\pi/2$) and goal position at (34,23,0,0,0). The planner soon pushed the platform from the high cost-to-traverse area to the low cost-to-traverse area. The platform is able to include the heading constraint at the destination.

5 Conclusions and Future Work

The approach presented in this work is based on dynamic programming and Particle Swarm Optimisation. It includes adaptations to the different algorithms in order to achieve a reduction in the processing cost of the planner. The QT-PWL approximation is used to reduce the processing cots. This approximation groups areas with identical or similar dense properties. The PSO approach includes the nonholonomic constraints of the platform in the planned path. The proposed algorithm was tested in different situations and with different platforms. Results show that this approach is able to plan in a dense context for platforms with different number of degrees of freedom.

In the present approach the QT-PWL approximation is recalculated on its entirety. Future work involves the incremental change in the existing approximation when the environment is changed allowing for an efficient replanning when needed. Also, the QT-PWL approximation could summarise patches that have been traversed or that are far away from the robot in order to further reduce the processing cost.

References

[Barbehenn, 1998] M. Barbehenn. A note on the complexity of dijkstra's algorithm for graphs with weighted vertices. *Computers, IEEE Transactions on*, 47(2):263, 1998.

[Bellman, 1957] R. Bellman. Dynamic programming, princeton. *NJ: Princeton UP*, 1957.

[Gennery,] D. Gennery. Traversability analysis and path planning for a planetary rover. *Autonomous Robots*.

[Guivant *et al.*, 2012] Jose Guivant, Stephen Cossell, Mark Whitty, and Jayantha Katupitiya. Internet-based operation of autonomous robots: The role of data replication, compression, bandwidth allocation and visualization. *Journal of Field Robotics*, 29(5):793–818, 2012.

[Laumond, 1987] J. P.l Laumond. Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1120–1123, 1987.

[Nieto *et al.*,] J. I. Nieto, J. E. Guivant, and E. M. Nebot. The hybrid metric maps (hymms): a novel map representation for denseslam. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 1, pages 391–396 Vol.1.

[Nieto *et al.*, 2006] J.I. Nieto, J. E. Guivant, and E. M. Nebot. Denseslam: Simultaneous localization and dense mapping. *The International Journal of Robotics Research*, 25(8):711–744, 2006.

[Pereida and Guivant, 2013] Karime Pereida and Jose Guivant. Hybrid dijkstra-pso algorithm for motion planning of non-holonomic multiple-trailer platforms in dense contexts. In *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, pages 13–18. IEEE, 2013.

[Robledo *et al.*,] A. Robledo, J. E. Guivant, and S. Cossell. Pseudo priority queues for real-time performance on dynamic programming processes applied to path planning. In *Australasian Conference on Robotics and Automation*.

[Sekhavat *et al.*, 1998] S. Sekhavat, P. Svestka, J. P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *The International Journal of Robotics Research*, 17(8):840–857, 1998.

[Yan *et al.*,] D. Yan, S. He-xu, L. Zuo-Jun, and B. Zhi-Yuan. Kinematics and constraint analysis of tractor-trailer mobile robot. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 3, pages 1380–1384 Vol. 3.

[Yatziv *et al.*, 2006] L. Yatziv, A. Bartesaghi, and G. Sapiro. O(n) implementation of the fast marching algorithm. *Journal of Computational Physics*, 212(2):393–399, 2006.