Proceedings of Australasian Conference on Robotics and Automation, 2-4 Dec 2013, University of New South Wales, Sydney Australia

PWL Approximation for Dense Mapping and Associated Dijkstra Processes for the Concurrent Synthesis of Multiple Full Cost-to-Go Functions

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

Abstract

This work proposes and illustrates the synthesis of multiple cost-to-go functions based on PWL approximation of a dense environment. The goal is to create a data structure based on PWL quadtree that accurately describes the working space allowing optimisation algorithms to explore greater areas of the working space in an efficient manner. To illustrate the proposed approach, a Dijsktra algorithm will be used to compute cost-to-go functions in both, the working space and the data structure and the results will be presented. The methodology includes adaptations to the algorithms used in order to achieve higher efficiency of the computational cost.

1 Introduction

Generally, path planning algorithms represent the working space divided in two categories, obstacles and free spaces. However, for many applications it is important for the platform to be able to discriminate among the different degrees of traversability of the free space and plan accordingly. This is achievable by the use of dense contexts, but efficient data structures are needed to increase speed of such algorithms.

In the proposed approach the environment properties are approximated by QuadTree Piece Wise Linear (PWL) representations, which implies the optimisation process can also exploit the QuadTree partition to explore the configuration space. Provided that working environments of robotic platforms have cluttered spaces and large regions where the dense properties are smooth, QuadTree PWL representations are able to represent the context in a fraction of the cells that would need a homogeneous grid.

The optimality of the path can be quantitatively described by more than one parameter, such as time to travel, distance to travel or even safety of the platform. Furthermore, it is of interest of the present work to be able to use the computed data structure to plan for dense contexts. In many cases dense properties of the perceived environment are needed as the cost function depends on both, the control actions and these properties. For the case of route planning, dense properties are inherent to the terrain and usually are the result of a mapping or learning process, even a SLAM process such as DenseSLAM [Nieto *et al.*, 2004] and [Nieto *et al.*, 2006]. In this work the values of such dense properties are expressed as a function of a 2D geographical coordinate.

The present work is organised as follows, Section 2 presents a brief description of the used algorithms. In Section 3 the proposed approach is described. Section 4 shows the results obtained using different contexts. Finally Section 6 presents conclusions and future work.

2 Related research

Dense contexts are proposed in [Nieto *et al.*, 2006] where different sensory information is represented in a dense multi-layered map. DenseSLAM, i.e. the process of performing SLAM (Simultaneous Localisation and Mapping) in a dense environment, obtains and maintains a detailed environment representations. Each layer represents different properties of the environment in a Hybrid metric map representation [Nieto *et al.*, 2004]. With this rich representation it is possible to better assist in the navigation process as it is an effective way to model the environment.

Rapidly-Exploring Random Trees is a path planning algorithm aimed for higher-dimensional state spaces. It is an incremental sampling and searching approach that incrementally constructs a search tree that gradually improves resolution [Lavalle and Kuffner Jr, 2000]. It also performs kinodynamic planning [LaValle and Kuffner, 2001] that allows the planner to solve problems with high degrees of freedom and complicated system dynamics. However, it does not perform well in environments with narrow corridors [LaValle and Kuffner, 2001]; it does not work in dense contexts and the computation time varies

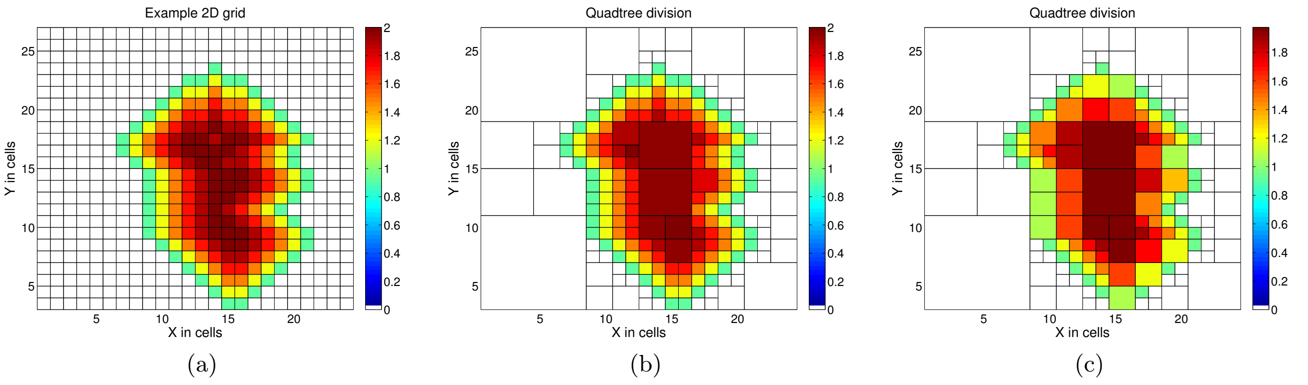


Figure 1: Specific area zoomed for comparison of the division process by the PWL QuadTree: (a) shows the original 2D grid map; (b) shows the QuadTree partition created with a small threshold to divide; and, (c) shows the QuadTree partition created with a large threshold to divide.

dramatically as the metri is varied, a problem described as difficult as the path planning problem itself.

In [Gennery, 1999] the exteroceptive capabilities of the robot are used to describe the environment's traversability. This information is used to compute a path. However it highlights that the amount of computation used increases roughly as the cube of the linear dimensions of the grid. Moreover, [Pereida and Guivant, 2013] proposes the use of a hybrid approach based on Dijkstra and PSO algorithms. But it needs to improve efficiency to plan in a higher dimensionality. Finally, [Yahja *et al.*, 2000] proposes a D* algorithm in framed QuadTrees for path planning.

However, in many cases, multiple full cost-to-go functions need to be synthesised in order to assist higher level planners, such as the ones required in logistics problems. In such cases, investing in any preprocessing that can simultaneously improve the performance of the multiple individual optimisation processes is of relevant benefit.

3 Description of the proposed approach

The current work is aimed at planning in dense contexts. In many applications the workspace of the platform will consist of large areas of similar characteristics, whether they are being occupied by obstacles, free or in a certain range in between. For this reason, a partition is proposed to describe the workspace of the platform. Particularly, for the case of dense properties, such as certain terrain characteristics, certain regions present values that can be approximated by a low number of patches of a PWC (Piece Wise Constant) or a PWL (Piece Wise Linear) representation.

For the current work the work space is a 2D grid map of the environment which assigns costs of traversability to each cell. Obstacles present the highest (infinite) costs, areas difficult to traverse are assigned high (but finite) costs while free space (e.g. easy to traverse terrain) has a low cost to be traversed. In order to describe the terrain (and other dense properties) in an efficient way, a PWL approximation is applied. In order to simplify the run-time processing requirements of synthesizing the approximating function, a less expensive version is applied, it is called QuadTree PWL. The QT-PWL has a partition of the function's domain that is constrained to respect a QuadTree structure, [Guivant *et al.*, 2012]. The PWL approximation for a 2D function is defined as follows:

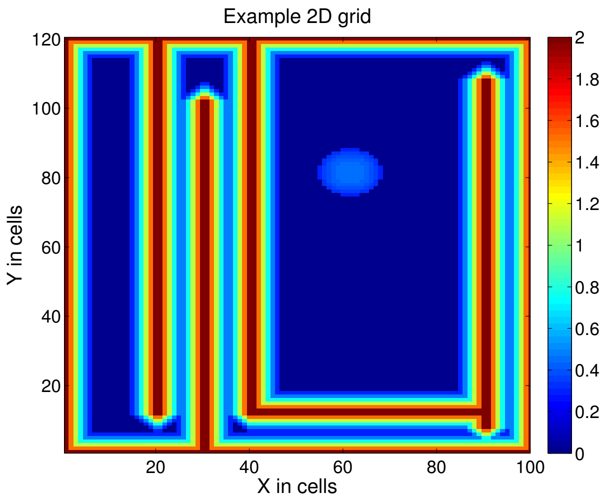
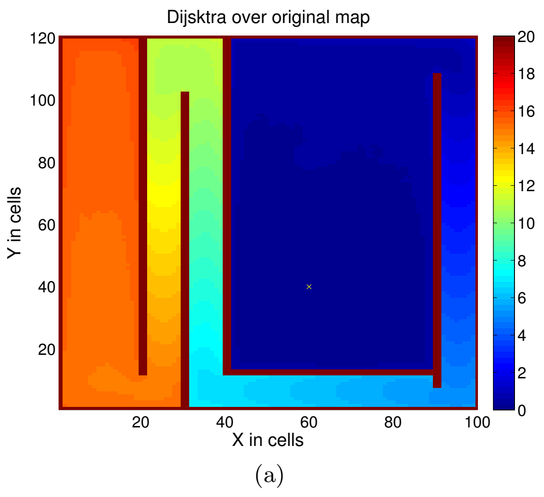
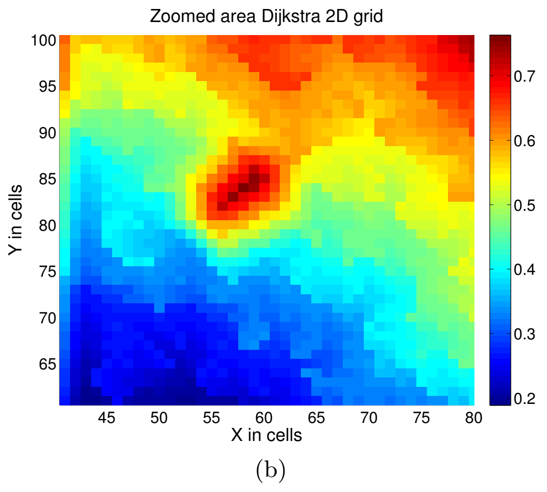
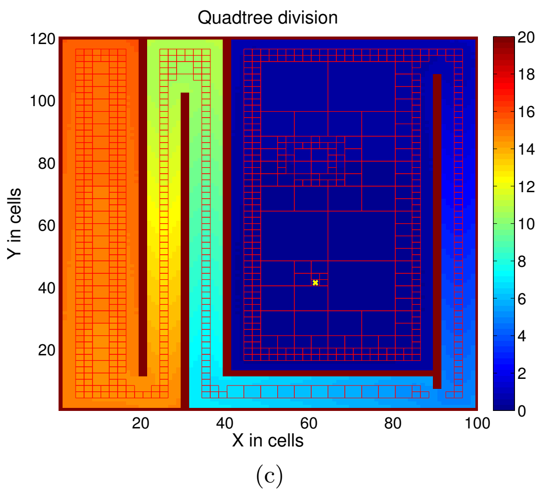


Figure 2: Example 2D grid map with an area of higher costs to traverse depicted as a light blue circle. The reason of this higher cost may be due to terrain characteristics (vegetation, surface quality, etc.) that imply higher cost of being traversed.

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







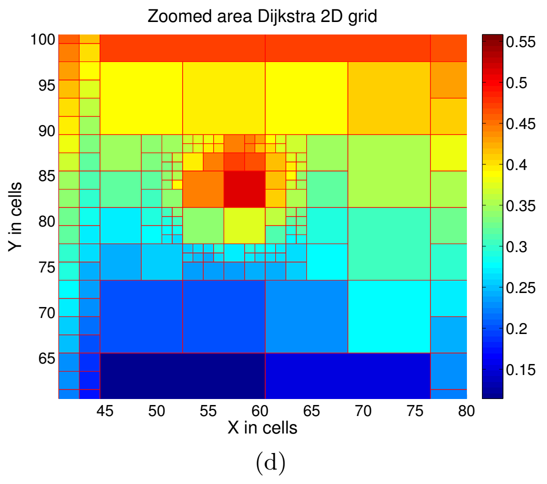


Figure 3: Figures (a) and (c) depict Dijkstra algorithm run over the 2D grid and QuadTree respectively. Red squares in figure (c) depict delimit nodes of size greater or equal to 2 cells. Figures (b) and (d) zoom in the area of higher depicted as a blue circle in Figure 2

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 defined as follows,

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

Because of the different traversability costs of the dense contexts, a threshold τ has to be determined in order to obtain a sufficiently accurate PWL approximation. As indicated in Equation 1, the error between the real value of the dense property and the approximating function has to be lower than the specified threshold. This threshold affects the resulting QuadTree partition. A higher threshold would allow the existence of cells presenting larger discrepancies between the approximating function and the real dense property. While this could sound advantageous, as the number of required nodes to describe the workspace would be reduced, the threshold should remain in a working range, in order to keep the PWL approximation realistic.

A typical area of a 2D grid map is shown in Figure 1a. Figure 1b and 1c show zoomed images around the same area depicting the different results obtained as a result of the PWL-QuadTree approximation. Figure 1b has a smaller threshold yielding a larger number of nodes but a closer representation of the original map. Figure 1c has a larger threshold depicting a coarser representation of the desired area.

For the optimisation process a Dijkstra approach is applied. In order to deal with the variable number and location of adjacent neighbours (due to the variable size

	2D grid	Quadtree
Explored nodes	11996	5091
Average node size	1	1.265
Average number of neighbours	8	8.0177

Table 1: Quantitative data obtained from running a Dijsktra process over the original 2D example grid and the computed QuadTree based on such grid.

of patches), those are treated according to the algorithm proposed in [Samet, 1984]. This algorithm consists of traversing the parent nodes up the tree and traversing down the tree in a specific direction. Further speed ups to the algorithm such as ropes and nets can be used. Once the neighbours to each node are computed, Dijkstra algorithm with Pseudo Priority Queue [Robledo *et al.*, 2010] is run in the leaf nodes of the QuadTree.

The cost of evaluating and maintaining the PWL representation (of the Dense Properties) is less relevant for the cases where the dense properties are the result of a mapping process performed simultaneously to the operation of the platform. As the mapping process evolves, the PWL approximation needs to be updated as well; however, those updates occur in localised regions, what implies that the PWL approximation does not need to be fully re-evaluated. Consequently the PWL approximation is gradually updated and also shared by the multiple optimisations (that generate the requested full cost-to-go functions).

4 Results

In order to illustrate the speed-up of the algorithm, a Dijkstra process was performed in both cases, a regular grid and a QT-PWL built based on the information of the regular grid. In Figure 2 an example involving a 2D grid is presented; Figure 3a presents the result of running the Dijkstra algorithm in the regular grid while Figure 3c depicts the Dijsktra algorithm applied to the QT-PWL case. Boundaries of cells with size over 2 cells are shown on red. Borders of small cells (sizes 1×1 and 2×2) are not shown, for the sake of symplifying the figures. It can be appreciated that a large area is covered by these nodes. Moreover, the shape of the cost-to-go function is similar in both cases, varying slightly only by the way the grid is partitioned. Figures 3b and 3d show a zoomed area around the area with higher cost to show that Dijkstra algorithm is assigning higher costs to that area in both, the 2D grid and the QuadTree.

However, a more substantial difference between the two approaches is shown by the quantitative differences illustrated in Table 1. The number of nodes to be explored by the Dijkstra algorithm is significantly reduced when using a QT-PWL while the average number of neighbours is close to eight as shown in Table 1 which assures the convenience of using QuadTree as a data structure. It is also important to observe that large areas of free space are represented by PWL patches which means large areas can be rapidly explored by optimisation algorithms given the fact that the area they enclose possesses similar attributes.

	2D grid	QuadTree
Explored nodes	632943	129512
Average node size	1	1.3294
Average number of neighbours	8	8.0319

Table 2: Quantitative data obtained from running a Dijsktra process over the real context and the computed QuadTree based on such grid.

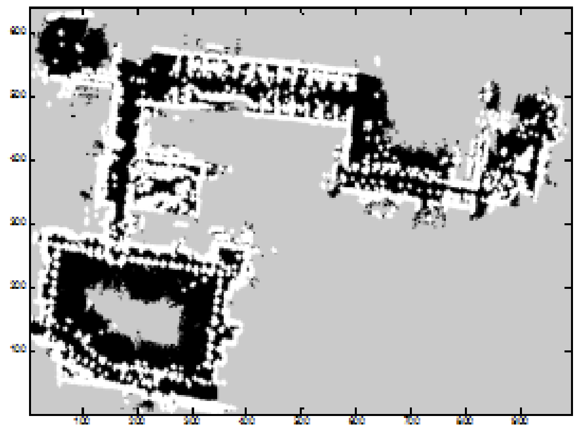


Figure 4: Local cost of OG. The cells are assigned a cost of being traversed based on the terrain characteristics. the cells representing the obstacles are dilated by adding a buffer zone considering the size of the platform. Dilated obstacles are shown in white (highest local cost, i.e. "infinite"); clear cells are shown in black and unknown cells are shown in grey.

The proposed algorithm was tested in a real context based on the UGV platform and methods presented in [Whitty *et al.*, 2010] and [Guivant *et al.*, 2012]. The platform has 3D imaging capabilities, associated data fusion and global localisation process detailed in [Guivant *et al.*, 2012] and [Robledo *et al.*, 2010]. The platform computes a 2D map composed by multiple properties i.e. layers. From this information the cost of traversing a cell is based on properties of the terrain such as inclination and discontinuities of the surface. A typical map generated by the platform is shown in Figure 4 and corresponds to part of the University's campus. The map is classified in obstacles, depicted in white, traversable space indicated in colors close to black and unknown cells in grey.

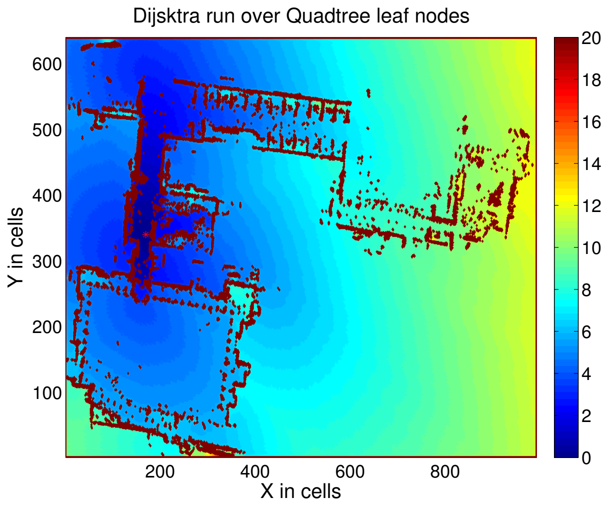


Figure 5: Dijkstra algorithm over 2D grid depicting part of the University's campus. Goal position is depicted by a light red asterisk (at point $(170, 340)$).

Figure 5 shows the result of running Dijkstra algorithm in this grid map. Figure 6 depicts the result of running Dijkstra algorithm over the QT-PWL representation that was evaluated based on the information of the grid map. Black lines delimit the area covered by leaf nodes of size equal or greater than 16 cells. It can be observed that the area covered by these cells is more than half of the area covered by smaller cells. Once more, the corresponding cost-to-go functions are similar, as shown bu the coloration of the environment in Figures 5 and 6. Furthermore, Table 2 shows a quantitative comparison of the results. The number of explored nodes drops significantly when using the PWL QuadTree. Moreover, the average number of neighbour nodes is maintained close to eight which assures that the computation cost of Dijsktra algorithm is not increased by an increased number of neighbours per node.

Finally, this algorithm is able to answer to different queries of goal nodes, being able to return different cost-to-go functions, one per goal node requested. The advantage in this approach is that the most computationally expensive task, which is the PWL approximation of the dense mapping properties, its associated QuadTree partition and the calculation of neighbours is performed only once. Later queries use the previously computed QuadTree in order to calculate the cost-to-go function. This is exemplified in Figure 7 where multiple requests are made and the generated cost-to-go functions are returned.

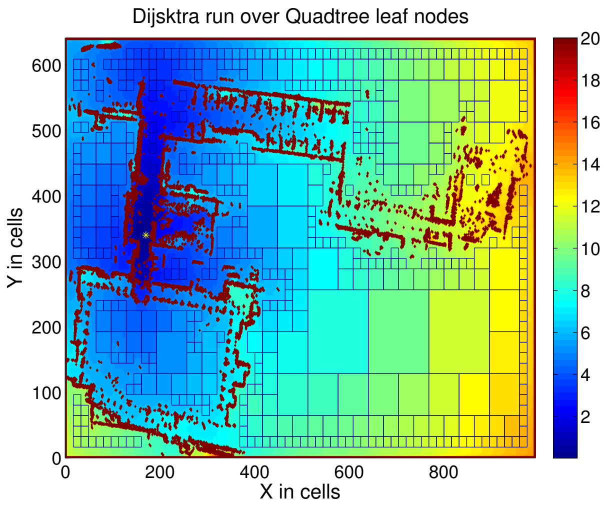


Figure 6: Dijkstra algorithm over QT-PWL partition computed based on the 2D grid depicting part of the University's campus. Goal position is depicted by a yellow asterisk (at position $(170, 340)$). Black lines delimit leaf nodes of size equal or greater to 16.

5 Conclusions and future work

As the above results showed, an effective representation of the workspace of a given platform allows substantial speed ups for the algorithms that run through them. The results obtained with QT-PWL are consistent with those obtained using 2D occupancy grids. The latter is because bigger nodes cover larger areas of cells with similar characteristics or smooth variations. This is particularly advantageous in open areas where the dense properties can be well approximated by large linear patches. This algorithm can generate multiple full cost-to-go functions, necessary for certain applications that need to solve logistic problems. Even though time has to be spent computing the QT-PWL and calculating the neighbours to each leaf node, it is still very effective as information can be reused through several processes, as shown in Figure 6. The proposed approach is able to work in dense contexts and, together with approaches such as the proposed in [Pereida and Guivant, 2013], will able to plan in higher dimensions.

Future work will be focused in dealing with higher dimensionality cases, such as including more Degrees of Freedom for the case of considering the heading and speed of the platforms.

References

[Gennery, 1999] D. Gennery. Traversability Analysis and Path Planning for a Planetary Rover. *Autonomous Robots*, 6(2):131–146, 1999.

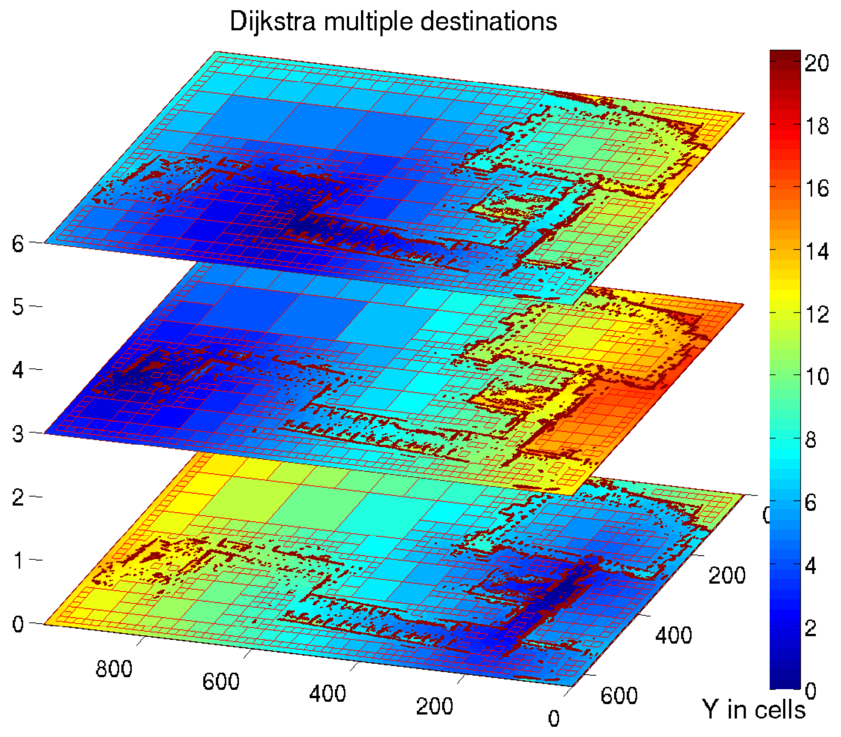


Figure 7: Three different cost-to-go functions based on different goal nodes calculated with the same QuadTree based on data obtained at the University's campus.

[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.

[Lavalle and Kuffner Jr, 2000] S. M. Lavalle and J. J. Kuffner Jr. Rapidly-Exploring Random Trees: Progress and Prospects. In *Algorithmic and Computational Robotics: New Directions*. Citeseer, 2000.

[LaValle and Kuffner, 2001] S. M. LaValle and J. J. Kuffner. Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[Nieto *et al.*, 2004] 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, 2004.

[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, 2013.

[Robledo *et al.*, 2010] 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*, 2010.

[Samet, 1984] Hanan Samet. The Quadtree and Related Hierarchical Data Structures. *ACM Comput. Surv.*, 16(2):187–260, June 1984.

[Whitty *et al.*, 2010] M. Whitty, S. Cossell, K.S. Dang, J. Guivant, and J. Katupitiya. Autonomous navigation using a real-time 3D point cloud. In *Australasian Conference on Robotics and Automation 2010*, 2010.

[Yahja *et al.*, 2000] Alex Yahja, Sanjiv Singh, and Anthony Stentz. An efficient on-line path planner for outdoor mobile robots. *Robotics and Autonomous Systems*, 32(2–3):129–143, 2000.