J Intell Robot Syst (2013) 71:303–317
DOI 10.1007/s10846-012-9794-2

Fast Path Re-planning Based on Fast Marching and Level Sets

Bin Xu · Daniel J. Stilwell · Andrew J. Kurdila

Received: 25 February 2012 / Accepted: 18 October 2012 / Published online: 11 November 2012

Abstract We investigate path planning algorithms that are based on level set methods for applications in which the environment is static, but where an a priori map is inaccurate and the environment is sensed in real-time. Our principal contribution is not a new path planning algorithm, but rather a formal analysis of path planning algorithms based on level set methods. Computational costs when planning paths with level set methods are due to the creation of the level set function. Once the level set function has been computed, the optimal path is simply gradient descent down the level set function. Our approach rests on the formal analysis of how value of the level set function changes when the changes in the environment are detected. We show that in many practical cases, only a small domain of the level set function needs to be re-computed when the environment changes. Simulation examples are presented to validate the effectiveness of the proposed method.

B. Xu · D. J. Stilwell (✉)
The Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA
e-mail: stilwell@vt.edu

B. Xu
e-mail: bxu@vt.edu

A. J. Kurdila
The Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA
e-mail: kurdila@vt.edu

Keywords Path planning · Autonomous vehicle · Autonomous navigation

1 Introduction

We consider an autonomous vehicle navigating towards a predefined target in a static environment for which an a priori map is available. Due to the inconsistency between the a priori map and the actual environment, path replanning is required in order to avoid collisions with obstacles that do not appear on the a priori map as well as to search for a better paths than those indicated by the a priori map.

Our principal contribution is not a new path planning algorithm. Instead, we seek to develop rigorous justification for path planning algorithms based on level set methods. The result is a specific algorithm that is very similar to E* [20], which is also based level set methods. The proposed method does not account for full vehicle dynamics or specific requirements such as maximum turn-rate. Thus, intuitively speaking, our approach is well-suited to vehicles that operate at sufficiently slow speeds such that they can follow arbitrary prescribed paths. This class of vehicles includes certain autonomous surface vehicles, which motivate our work, but also includes classes of ground vehicles and ground hovercrafts. In all cases, we presume that a low-level path-following controller onboard the vehicle enables the vehicle to follow the path generated by the planning algorithm. We presume that the environment is represented by a map that consists of a two dimensional uniform-sized occupancy grid [5] which is initially generated from a priori knowledge of the environment. Obstacles in the environment are detected by onboard sensors that have a limited range.

Path planning for autonomous vehicles has been studied for decades. Excellent references that survey the current literature can be found in [14, 15]. These methods can be grouped into two categories: local and global replanning. Local planning methods, including [2, 8, 12, 13, 21], among many others, are fast, but they can fail in trap scenarios for which progress toward the desired endpoint is impossible due to the lack of global knowledge about the environment. Global replanning can avoid these problems but can be computationally expensive (see e.g. [11, 24]). In [6], a group of global replanning methods are introduced. These methods share some common attributes. They are variants of A* search (see e.g. [14, p. 604]). The map is modeled by a set of nodes and the corresponding set of costs to traverse between two adjacent nodes. Thus, finding an optimal path is treated as a minimal cost path searching problem in graph [25]. When the environment changes, the costs to traverse the corresponding nodes will change. The overall minimal cost and the path to travel from a given node to the goal are consistently updated.

The proposed approach to path planning is based on level set methods, which can be used to compute minimum risk paths. The level set function is the solution of a PDE known as the Eikonal equation. The value of the level set function at any point is the cost to traverse from that point to the goal location, and the optimal path is simply gradient descent of the level set function [3, 9, 10]. The solution of the Eikonal equation can be approximated by the fast marching method (FMM) [23]. This method has been successfully applied to path planning when the environment is known, for example, in [7, 9, 17, 19]. For path planning in uncertain environments, D* which is a varient of A* is proposed in [6], and E* Lite which is based on level set methods is proposed in [20]. A qualitative comparison between A* search and level set methods can be found in [1].

Our work is strongly inspired by deployment of the E* algorithm, reported in [20]. E* is a path planning approach based on principals from A* and D* applied to a level set framework. In implementation, E* differs only in relatively minor ways from the algorithm reported herein. Indeed, the principal contribution of our work is to provide a formal derivation and analysis of a path planning algorithm based on level set methods rather than a completely new set of algorithms. Our formal approach not only yields the algorithm reported herein and attendant performance guarantees, but it can be useful for formal analysis of other planning methods that are derived from level set methods. We propose a dynamic fast marching method which builds upon the original fast marching method [23] such that the new paths are replanned in many cases without computing the entire level set functions. We present rigorous and formal analysis of how changes in the environment produce corresponding changes in the level set function, and how these changes can be used to reduce computational burden of using level set methods for path planning. Our algorithm addresses separately the case in which new obstacles are detected and the case in which new empty areas are detected. For the case that new obstacles are detected, the proposed method reduces computational expenses in two aspects. First, we show that if obstacles are not on the vehicle's current optimal trajectory and if there are no unexpected empty areas, the trajectory remains optimal. This observation allows us to delay computation of the level set function update if the original trajectory is still feasible. Second, if an unexpected obstacle intersects the current path, we show that only a portion of the level set values needs to be recomputed. For the case that new empty areas are detected, we again show that only a portion of the level set values usually needs to be computed. After we discuss the two difference cases separately, we then propose an algorithm that deals with both new empty areas and new obstacles simultaneously.

A preliminary report of this work appears in [26], where we addressed only the case that obstacles are added to the map. In this paper, we address the complete case that obstacles are added and removed from the map.

The paper is organized as follows. In Section 2, we introduce preliminary results, and in Section 3, we define the obstacle detection and formulate the path replanning problem. In Section 4, we show the development of the proposed replanning strategy when new obstacles are detected. In Section 5, we propose the algorithm that updates the level set values when new empty areas are detected. In Section 6, we present the algorithm that deals with the mixture of both new obstacles and new empty areas simultaneously. The computation efficiency of the proposed method is investigated in Section 7. Simulation results are illustrated in Section 8. Concluding remarks are presented in Section 9.

2 Preliminaries

We first review path planning using level set methods as originally presented in [9, 18, 23].

2.1 Eikonal Equation and Level Set Methods

Consider an autonomous vehicle with position denoted by $\mathbf{x} \in \mathbb{R}^2$ in a global Cartesian reference frame, navigating in the closure $\overline{\Omega}$ of a connected and bounded open set $\Omega \subset \mathbb{R}^2$. The vehicle is modeled as a point mass under the assumption that its characteristic size is small relative to $\overline{\Omega}$. The task for the vehicle is to travel along an obstacle free path with minimum risk such that the vehicle can reach a predefined goal $\mathbf{z} \in \overline{\Omega}$.

For each point in $\overline{\Omega}$, we associate a risk for the vehicle to traverse a path, quantified by a cost function $g \in C^1(\overline{\Omega}; \mathbb{R})$, which is positive everywhere except at the target $\mathbf{z}$ for which $g(\mathbf{z}) = 0$. For any $\boldsymbol{\xi} \in \overline{\Omega}$, we define a function $Q(\boldsymbol{\xi})$ which represents the minimal cumulative cost to travel from $\boldsymbol{\xi}$ to $\mathbf{z}$

$$Q(\boldsymbol{\xi}) = \min_{\mathbf{c}} \int_0^1 g(\mathbf{c}(p)) \left\| \mathbf{c}'(p) \right\| dp \qquad (1)$$

where $\mathbf{c} \in \mathrm{Lip}([0,1]; \overline{\Omega})$ is a Lipschitz continuous parameterized path with, $\mathbf{c}(0) = \boldsymbol{\xi}$ being a current starting point, and $\mathbf{c}(1) = \mathbf{z}$ being the goal, and $\mathbf{c}$ is differentiable almost everywhere in $[0,1]$ (see e.g. [16, p. 116]).

From Eq. 1 and from the definition of g, a direct application of the fundamental theorem of calculus of variation shows that $Q(\boldsymbol{\xi})$ is the viscosity solution [23] of the following Eikonal equation [9],

$$\|\nabla Q(\boldsymbol{\xi})\| = g(\boldsymbol{\xi}), \quad Q(\mathbf{z}) = 0. \qquad (2)$$

The value $Q(\boldsymbol{\xi})$ is the overall minimal risk to travel from the point $\boldsymbol{\xi}$ to the goal, and the optimal paths are along the gradient of $Q(\boldsymbol{\xi})$. Figure 1 shows an example of a priori map, and Fig. 2 shows the corresponding level set function. Figure 3 illustrates that a minimum-risk path progresses in the gradient direction down the level set function.

2.1.1 Finite Difference Scheme and its Properties

The Eikonal Eq. 2 often cannot be solved analytically. In [23], a first order update scheme is proposed which approximates the viscosity solution of Eq. 2.

Let the goal location be $\mathbf{z} = [z_1, z_2]^T$. In order to discretize $\overline{\Omega}$, we define a set $\Psi \in \mathbb{Z} \times \mathbb{Z}$ that is composed of grids with mesh size Δx, where $\mathbb{Z}$ is the set of integers. We denote the approximate value of Q by $\mathcal{Q} : \Psi \to \mathbb{R}^1$ satisfying

$$\mathcal{Q}(i, j) \simeq Q(i\Delta x + z_1, j\Delta x + z_2). \qquad (3)$$

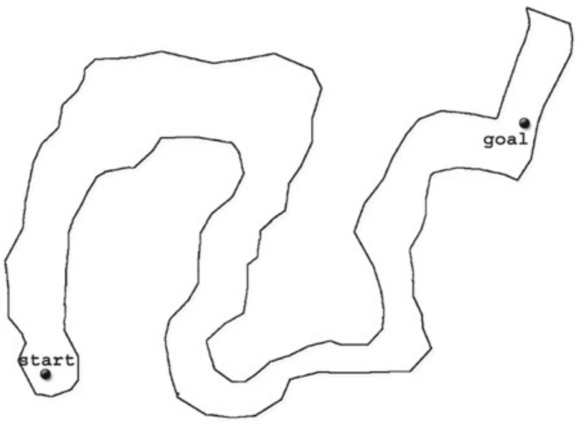

Fig. 1 An a priori map $\overline{\Omega}$ for a riverine environment

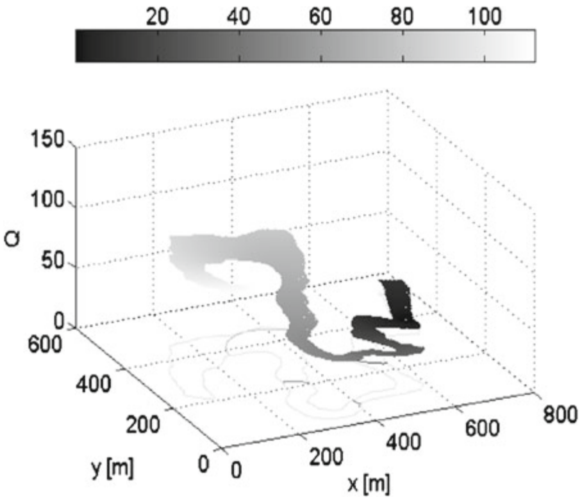

Fig. 2 The level set function for the a priori map

Correspondingly, we approximate the function g with $\mathbf{g} : \Psi \rightarrow \mathbb{R}^1$ satisfying

$$\mathbf{g}(i, j) = g(i\Delta x + z_1, j\Delta x + z_2). \tag{4}$$

We define the neighbors of a grid element $(i, j) \in \Psi$ to be the set of grid elements $(i+1, j)$, $(i-1, j)$, $(i, j+1)$ and $(i, j-1)$. If grid element (i, j) satisfies $[i\Delta x + z_1, j\Delta x + z_2]^T \notin \overline{\Omega}$, we set value $\mathcal{Q}(i, j)$ to be a very large number. Otherwise, if grid element (i, j) satisfies $[i\Delta x + z_1, j\Delta x + z_2]^T \in \overline{\Omega}$, the numerical approximation $\mathcal{Q}(i, j)$ satisfies the following conditions

$$\mathcal{Q}(0, 0) = 0 \tag{5}$$

$$\max\left(\frac{\mathcal{Q}(i, j) - \min(\mathcal{Q}(i-1, j), \mathcal{Q}(i+1, j))}{\Delta x}, 0\right)^2 + \max\left(\frac{\mathcal{Q}(i, j) - \min(\mathcal{Q}(i, j+1), \mathcal{Q}(i, j-1))}{\Delta x}, 0\right)^2 - \mathbf{g}^2(i, j) = 0, \quad \forall (i, j) \neq (0, 0) \tag{6}$$

which converges to the continuous solution as $\Delta x \rightarrow 0$. As remarked in [22], $\mathcal{Q}$ in Eqs. 5 and 6 exhibits a first order accuracy of order Δx.

The discrete approximation solution $\mathcal{Q}$ is found as follows. We define

$$a := \min(\mathcal{Q}(i+1, j), \mathcal{Q}(i-1, j)),$$
$$b := \min(\mathcal{Q}(i, j+1), \mathcal{Q}(i, j-1)). \tag{7}$$

The approximate solution $\mathcal{Q}(i, j)$ is computed by identifying two cases,

Case 1 If $|a - b| \geq \mathbf{g}(i, j)\Delta x$, then

$$\mathcal{Q}(i, j) = \min(a, b) + \mathbf{g}(i, j)\Delta x \tag{8}$$

Case 2 If $|a - b| < \mathbf{g}(i, j)\Delta x$, then $\mathcal{Q}(i, j)$ is selected as the larger solution of the quadratic equation

$$(\mathcal{Q}(i, j) - a)^2 + (\mathcal{Q}(i, j) - b)^2 - \mathbf{g}^2(i, j)\Delta x^2 = 0 \tag{9}$$

that is

$$\mathcal{Q}(i, j) = \left(a + b + \sqrt{2\mathbf{g}^2(i, j)\Delta x^2 - (a - b)^2}\right) \Big/ 2 \tag{10}$$

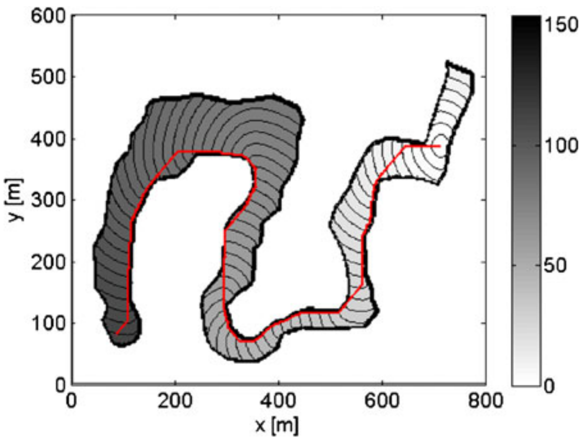

Fig. 3 The level sets and the optimal path for the a priori map

Both Eqs. 8 and 10 show that for each grid element (i, j), the value $\mathcal{Q}(i, j)$ depends on the smaller values of the neighbors. This is called upwind property indicating that the values of $\mathcal{Q}$ propagate from smaller values to larger values. In addition, the scheme admits no local minima. Indeed, if $\mathcal{Q}(i, j)$ would be lower than its neighbors, and given that $\mathbf{g}(i, j) > 0$ for all $(i, j) \neq (0, 0)$, the left-hand side of Eq. 6 would be negative.

2.1.2 Fast Marching Method Algorithm

The fast marching method proposed in [23] can efficiently compute the solution for Eq. 6. Indeed, the fast marching method solves Eqs. 5 and 6 in $O(N \log N)$ where N is the number of grid elements in $\overline{\Omega}$. Making use of the upwind property, the fast marching method builds the solution outward from smaller values of $\mathscr{Q}$ to larger values. The reader is referred to [23] for details.

2.1.3 Directed Graph

As proposed in [20], when computing the approximate solution $\mathscr{Q}$ of the Eikonal equation, one can use a directed graph to explicitly represent upon which neighbor grid elements the value of $\mathscr{Q}(i, j)$ depends. We denote by Σ the graph whose nodes corresponds to the grid elements in Ψ and whose directed edges represent the computational dependance between the value of $\mathscr{Q}$ at each node. As an example depicted in Fig. 4, we illustrate the construction of Σ with respect to the value dependence of the grid element in the center. For notational simplicity, we define the center grid element by E and its four neighbors by A, B, C and D. For Case 1, since the value $\mathscr{Q}(E)$ is determined by its smallest neighbor, say node A, Σ would contain a directed edge from A to E as in Fig. 4b. For Case 2, $\mathscr{Q}(E)$ depends on two nodes, say A and B. Then Σ would contain directed edges from A to E and from B to E, as in Fig. 4c.

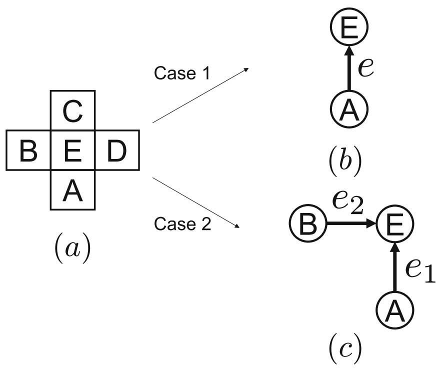

Fig. 4 **a** Grid element E and its neighbors; **b** Graph Σ for Case 1; and **c** Graph Σ for Case 2

If there is a directed edge from a node (i, j) to another node (k, m), then (k, m) is said to be a direct child of (i, j), and (i, j) is said to be a direct parent of (k, m). In general, if a path leads from (i, j) to (p, q), then (p, q) is said to be a child of (i, j) and (i, j) is said to be a parent of (p, q).

3 Problem Formulation

We let the open, connected and bounded set $\Omega \subset \mathbb{R}^2$ be a domain that is large enough such that any new obstacles and any new empty areas are detected in $\overline{\Omega}$. Based on the a priori map information, we choose g_0 as the initial risk for traversal such that g_0 is much larger for the areas that are marked as obstacles than the area that are marked as empty.

3.1 Obstacles and Empty Areas Detection

Since the onboard sensor has finite range, changes to the map are local to the vehicle. We assume that the sensor obtains measurements periodically with period h. Let the sensor sampling time be $t_k = t_0 + kh$, where $k = 0, 1, \cdots$. In order to model the environmental changes that are induced by detection of obstacles, we denote the cost functions at time instant t_k by g_k. If the sensor detects no changes the map at time t_{k+1}, then we write $g_k = g_{k+1}$ to denote that $g_k(\boldsymbol{\xi}) = g_{k+1}(\boldsymbol{\xi})$ for any point $\boldsymbol{\xi} \in \overline{\Omega}$. If a point $\boldsymbol{\xi} \in \overline{\Omega}$ is detected in an obstacle domain at time t_{k+1}, then $g_k(\boldsymbol{\xi}) < g_{k+1}(\boldsymbol{\xi})$. Conversely, if an obstacle on the map is determined not to exist by the local sensor at a point $\boldsymbol{\xi} \in \overline{\Omega}$ at time t_k, then $g_k(\boldsymbol{\xi}) > g_{k+1}(\boldsymbol{\xi})$. Thus, the new Eikonal equation that is induced by new cost functions g_{k+1} at time t_{k+1} is

$$\|\nabla Q_{k+1}(\boldsymbol{\xi})\| = g_{k+1}(\boldsymbol{\xi}),\ \ Q_{k+1}(\mathbf{z}) = 0. \tag{11}$$

where $\boldsymbol{\xi} \in \overline{\Omega}$.

3.2 Problem Statement

One way to update the solution of Eikonal equation (Eq. 11) is to recalculate level set values over the entire domain $\overline{\Omega}$ with respect to the new cost function g_{k+1}. For the purpose of reducing the computation cost, we propose a new path replanning method such that the solutions of Eikonal equations can be more efficiently updated upon the environmental changes.

In the following sections, to facilitate the analysis, we separately address the case for which new obstacles are detected and the case for which a priori known obstacles are found to not exist. When we address the path replanning in the presence of new obstacles, we assume that g_k monotonically increases with the time sequence t_k, and vice versa. We find, in our analysis, that the two cases are reversible in some situations.

4 Optimality of Trajectories in the Presence of New Obstacles

In this section, we discuss the case that only new obstacles are detected. Thus, we assume that g_k monotonically increases with the time sequence t_k. We show that an optimal path remains optimal when a new obstacle is detected so long as the obstacle does not intersect the path. Due to this fact, we are required to update the values of the level set function only when a new obstacle intersects the path. For notational simplicity, without further specification, $g_{k+1} \geq g_k$ means that for all $\xi \in \overline{\Omega}$, $g_{k+1}(\boldsymbol{\xi}) \geq g_k(\boldsymbol{\xi})$. Correspondingly, similar meaning can be deduced for $\mathbf{g}_{k+1} \geq \mathbf{g}_k$ in the discrete case.

Proposition 1 *Suppose Q_k and Q_{k+1} are the solutions of Eq. 11 for the cost functions g_k and g_{k+1}, respectively. Denote the obstacle detected at time t_{k+1} by a bounded open set $O(k+1) \subset \overline{\Omega}$. Assume $g_{k+1} \geq g_k$. Then,*

(a) *For all $\boldsymbol{\xi} \in \overline{\Omega}$, $Q_{k+1}(\boldsymbol{\xi}) \geq Q_k(\boldsymbol{\xi})$.*
(b) *If $\mathbf{c}^*$ is the optimal path associated to g_k and if $O(k+1)$ does not intersect $\mathbf{c}^*$, then $\mathbf{c}^*$ is still one of the optimal paths associated to the new cost function g_{k+1}.*

Proof of Proposition 1 We prove (a) first and then (b) as an immediate implication of (a).

Let $I = [0, 1]$. Given the cost function g_k, denote the cumulative cost along an arbitrary differentiable parameterized path $\mathbf{c}(p) \in \mathrm{Lip}(I; \overline{\Omega})$ by

$$J_k(\mathbf{c}) = \int_I g_k(\mathbf{c}(p)) \left\| \mathbf{c}'(p) \right\| dp \tag{12}$$

where $\mathbf{c}(0) = \boldsymbol{\xi}$ and $\mathbf{c}(1) = \mathbf{z}$. Thus, by definition,

$$Q_k = \min_{\mathbf{c} \in \mathrm{Lip}(I; \overline{\Omega})} J_k(\mathbf{c}), \tag{13}$$

and

$$Q_{k+1} = \min_{\mathbf{c} \in \mathrm{Lip}(I; \overline{\Omega})} J_{k+1}(\mathbf{c}). \tag{14}$$

Denote with L the intersection of curve $\mathbf{c}(p)$ and obstacle $O(k+1)$. Thus, L satisfies

$$L = \{p \in I : \ \mathbf{c}(p) \subseteq \overline{O(k+1)}\}. \tag{15}$$

Since for all $\boldsymbol{\xi} \in \overline{\Omega}$, $g_{k+1}(\boldsymbol{\xi}) \geq g_k(\boldsymbol{\xi})$, subtracting $J_k(\mathbf{c})$ from $J_{k+1}(\mathbf{c})$ yields

$$\begin{aligned} \Delta J(\mathbf{c}) =& J_{k+1}(\mathbf{c}) - J_k(\mathbf{c}) \\ =& \int_{I \setminus L} (g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))) \left\| \mathbf{c}'(p) \right\| dp \\ &+ \int_L (g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))) \left\| \mathbf{c}'(p) \right\| dp \\ =& \int_L (g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))) \left\| \mathbf{c}'(p) \right\| dp \end{aligned} \tag{16}$$

By inspecting the above equation, we conclude that (i) $\Delta J(\mathbf{c}) = 0$ when $L = \varnothing$, and (ii) $\Delta J(\mathbf{c}) \geq 0$ when $L \neq \varnothing$. This implies

$$J_{k+1}(\mathbf{c}) \geq J_k(\mathbf{c}) \tag{17}$$

for any differentiable parameterized path $\mathbf{c}$ connecting $\boldsymbol{\xi}$ and $\mathbf{z}$. Since Q_k and Q_{k+1} are the optimal values for $J_k(\mathbf{c})$ and $J_{k+1}(\mathbf{c})$, respectively, we conclude that $Q_{k+1} \geq Q_k$.

We now prove (b). Considering Eq. 16, since $L = \varnothing$, for the path along $\mathbf{c}^*$

$$J_{k+1}(\mathbf{c}^*) = J_k(\mathbf{c}^*) \tag{18}$$

Since $\mathbf{c}^*$ is the optimal path given g_k, $J_k(\mathbf{c}^*) \leq J_k(\mathbf{c})$. Together with Eq. 17, we conclude that for an arbitrary curve $\mathbf{c}$,

$$J_{k+1}(\mathbf{c}^*) = J_k(\mathbf{c}^*) \leq J_k(\mathbf{c}) \leq J_{k+1}(\mathbf{c}) \tag{19}$$

The inequality 19 indicates the path $\mathbf{c}^*$ is the optimal for the cost function $\mathbf{g}_{k+1}$. □

We analyze changes in the discretized solutions of the Eikonal equation that result when the value of the cost function $\mathbf{g}_k$ increases. Using the result of our analysis and employing the directed graph introduced in Section 2.1.3, we show that level set function does not necessarily need to be updated everywhere, and we identify the group of grid elements whose level set values should be updated.

4.1 The Approximation of the Discrete Level Set Function

Proposition 1 indicates that the values of level set function monotonically increases if the cost function monotonically increases. We now investigate the corresponding property for the discrete approximation of the level set function. Although the update scheme 6 introduces approximation errors, the following proposition ensures monotonicity of the discrete approximation $\mathcal{Q}_k$ with respect to the risk $\mathbf{g}_k$. This proposition will be used, in part, to show that if the sequence of discrete cost functions satisfy $\mathbf{g}_k \leq \mathbf{g}_{k+1}$, then discrete approximation of the level set value satisfies $\mathcal{Q}_k \leq \mathcal{Q}_{k+1}$.

Proposition 2 *Given cost functions $\mathbf{g}_k$ and $\mathbf{g}_{k+1}$, let $\mathcal{Q}_k$ and $\mathcal{Q}_{k+1}$ be the discrete solutions to Eq. 11. If $\mathbf{g}_{k+1} \geq \mathbf{g}_k$, then the approximation satisfies $\mathcal{Q}_{k+1}(i,j) \geq \mathcal{Q}_k(i,j)$ for all grid element $(i,j) \in \Psi$.*

Proof of Proposition 2 Using the monotonicity of the scheme 6, we show the proof by contradiction. From the scheme 6, we define a function

$$\begin{aligned} &F(\mathcal{Q}(i,j), \Phi_{ij}(\mathcal{V}), \mathbf{g}(i,j)) \\ &:= \max\left(\frac{\mathcal{Q}(i,j) - \min(\mathcal{V}(i-1,j), \mathcal{V}(i+1,j))}{\Delta x}, 0\right)^2 \\ &\quad + \max\left(\frac{\mathcal{Q}(i,j) - \min(\mathcal{V}(i,j+1), \mathcal{V}(i,j-1))}{\Delta x}, 0\right)^2 \\ &\quad - \mathbf{g}^2(i,j) \end{aligned} \tag{20}$$

where the second argument in $F(\cdot,\cdot,\cdot)$ is defined as the set

$$\Phi_{ij}(\mathcal{V}) := \{\mathcal{V}(i-1,j), \mathcal{V}(i+1,j), \mathcal{V}(i,j+1), \mathcal{V}(i,j-1)\}.$$

Note that when the scheme 6 holds, we have, for example,

$$0 = F(\mathcal{Q}_k(i,j), \Phi_{ij}(\mathcal{Q}_k), \mathbf{g}_k(i,j)). \tag{21}$$

We denote by Σ_{k+1} the graph corresponding to the value dependence of the solution $\mathcal{Q}_{k+1}$. Assume that there exists a node (i,j) where

$$\mathcal{Q}_{k+1}(i,j) < \mathcal{Q}_k(i,j). \tag{22}$$

Given that $\mathbf{g}_k \leq \mathbf{g}_{k+1}$, from Eqs. 21 and 22, we have

$$\begin{aligned} 0 &= F(\mathcal{Q}_k(i,j), \Phi_{ij}(\mathcal{Q}_k), \mathbf{g}_k(i,j)) \\ &\geq F(\mathcal{Q}_k(i,j), \Phi_{ij}(\mathcal{Q}_k), \mathbf{g}_{k+1}(i,j)) \\ &> F(\mathcal{Q}_{k+1}(i,j), \Phi_{ij}(\mathcal{Q}_k), \mathbf{g}_{k+1}(i,j)) \end{aligned} \tag{23}$$

From the definition of Eq. 20, $F(\mathcal{Q}_{k+1}(i,j), \Phi_{ij}(\mathcal{V}), \mathbf{g}_{k+1})$ monotonically decreases as the value of a neighbor of (i,j), say $\mathcal{V}(i+1,j)$, increases. Thus, from the inequality 23, we infer that there is at least one direct parent of the node (i,j) in the graph Σ_{k+1}, say $(i+1,j)$, satisfying

$$\mathcal{Q}_{k+1}(i+1,j) < \mathcal{Q}_k(i+1,j). \tag{24}$$

Repeating the same analysis, we always infer that Eq. 24 holds for the parents of (i,j) by tracing backward along the graph Σ_{k+1}, which eventually leads us to the conclusion that

$$\mathcal{Q}_{k+1}(0,0) < \mathcal{Q}_k(0,0) = 0.$$

The contradiction completes the proof. □

By using the directed graph, we identify the nodes that need to be updated. Assume that at time t_{k+1}, the vehicle detects new obstacles. Then, the cost functions of these grids are such that $\mathbf{g}_{k+1} > \mathbf{g}_k$. Consider a graph Σ_k that indicates dependence of the value of the level set function on other neighbor nodes at time t_k. Using Σ_k, Proposition 3, and Corollary 1, we show that when the cost function $\mathbf{g}_k$ increases to $\mathbf{g}_{k+1}$ level set values do not necessarily need to be updated at all grid elements, and we can identify the nodes for which update of the level set function is necessary.

Proposition 3 *Denote the computational dependence between nodes by the graphs Σ_k and Σ_{k+1} for times t_k and t_{k+1} respectively. Suppose that for every node $\mathbf{g}_{k+1} \geq \mathbf{g}_k$. For a grid element (i, j), if $\mathbf{g}_{k+1}(i, j) = \mathbf{g}_k(i, j)$ and, for all direct parents nodes of (i, j) denoted by (l, m), $\mathcal{Q}_k(l, m) = \mathcal{Q}_{k+1}(l, m)$, then $\mathcal{Q}_{k+1}(i, j) = \mathcal{Q}_k(i, j)$ and the direct parents of (i, j) in Σ_k are direct parents of (i, j) in Σ_{k+1}.*

Proof of Proposition 3 We only address Case 1 since the proof for Case 2 follows the similar procedure. For notational simplicity, we define $E := (i, j)$ and neighbors of E by A, B, C and D as shown in Fig. 4a. We assume that A is the only direct parent of E. Thus from Eq. 6,

$$\mathcal{Q}_k(A) \leq \mathcal{Q}_k(C), \tag{25}$$

From Eq. 6 and the assumption that node A is the only direct parent of E,

$$\mathcal{Q}_k(E) - \mathcal{Q}_k(A) = \Delta x \mathbf{g}_k(E).$$

Since $\min(\mathcal{Q}_k(B), \mathcal{Q}_k(D)) > \mathcal{Q}_k(E)$, we obtain

$$\min(\mathcal{Q}_k(B), \mathcal{Q}_k(D)) - \mathcal{Q}_k(A) \geq \mathbf{g}_k(E)\Delta x. \tag{26}$$

By Proposition 2, $\mathcal{Q}_{k+1} \geq \mathcal{Q}_k$ for all the neighbor nodes B, C and D. By hypothesis, $\mathcal{Q}_{k+1}(A) = \mathcal{Q}_k(A)$ since node A is a direct parent of node E in Σ_k, and $\mathbf{g}_{k+1}(E) = \mathbf{g}_k(E)$. From Eq. 25,

$$\mathcal{Q}_{k+1}(A) \leq \mathcal{Q}_{k+1}(C), \tag{27}$$

and from Eq. 26

$$\min(\mathcal{Q}_{k+1}(B), \mathcal{Q}_{k+1}(D)) - \mathcal{Q}_{k+1}(A) \geq \mathbf{g}_{k+1}(E)\Delta x. \tag{28}$$

Recalling a and b from Eq. 7,

$$\begin{aligned} a &= \min(\mathcal{Q}_{k+1}(B), \mathcal{Q}_{k+1}(D)) \\ b &= \min(\mathcal{Q}_{k+1}(A), \mathcal{Q}_{k+1}(C)) = \mathcal{Q}_{k+1}(A) \end{aligned}$$

and that note that from Eq. 28, $\min(a, b) = \mathcal{Q}_{k+1}(A)$. Thus from Eq. 8

$$\begin{aligned} \mathcal{Q}_{k+1}(E) &= \mathcal{Q}_{k+1}(A) + \mathbf{g}_{k+1}(E)\Delta x \\ &= \mathcal{Q}_k(A) + \mathbf{g}_k(E)\Delta x \\ &= \mathcal{Q}_k(E) \end{aligned} \tag{29}$$

which completes the proof for Case 1. □

Applying Proposition 3 from the node corresponding to the goal location $\mathbf{z}$, the following corollary follows.

Corollary 1 *Define the set*

$$\Upsilon := \{(l, m) \in \Psi : \mathbf{g}_{k+1}(l, m) > \mathbf{g}_k(l, m)\}. \tag{30}$$

For a node (i, j) and given the graph Σ_k, if $(i, j) \notin \Upsilon$, and if (i, j) is not a child of an element in Υ with respect to the graph Σ_k, then $\mathcal{Q}_{k+1}(i, j) = \mathcal{Q}_k(i, j)$.

Proof of Corollary 1 From Proposition 3, for any grid element $(i, j) \in \Psi$, $\mathcal{Q}_k(i, j) = \mathcal{Q}_{k+1}(i, j)$ if $(i, j) \notin \Upsilon$ and if $\mathcal{Q}_k(l, m) = \mathcal{Q}_{k+1}(l, m)$ for all direct parents (l, m) of (i, j). Starting from node $(0, 0)$, repeatedly applying Proposition 3 forwards through Σ_k, we conclude that if $(i, j) \notin \Upsilon$, and if (i, j) is not a child of the nodes in Υ, then $\mathcal{Q}_k(i, j) = \mathcal{Q}_{k+1}(i, j)$ which yields the desired conclusion. □

An example of the application of the corollary is shown in Fig. 5. This figure shows the graph Σ_k. The black node is detected as an obstacle. The white nodes are not the children of the black node and thus the corresponding values of $\mathcal{Q}_{k+1}$ do not need to be recalculated.. Since the grey nodes are

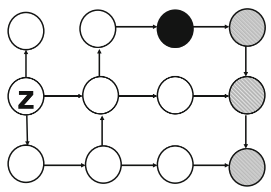

Fig. 5 Given the graph Σ_k, the nodes that need to be recomputed are the black node and its children

the children of the obstacle, the corresponding values of $\mathcal{Q}_{k+1}$ do need to be computed.

4.2 Algorithm

We now describe the proposed method to update the level set function. From Corollary 1, we only need to recalculate the nodes that are children of the node whose cost has increased. Therefore, the first step is to identify these nodes, using, for example, the depth first search (see e.g. [4, p. 477]). The second step is to recompute $\mathcal{Q}_{k+1}$ values for all children nodes. We employ the same principle of the fast marching method that propagates $\mathcal{Q}_{k+1}$ from smaller to larger values. Once the node corresponding to the location of the vehicle has been calculated, then an updated optimal path exists and further update of the level set value is not needed. Since obstacles are detected near the autonomous vehicle, the number nodes that must be recalculated is often very small.

5 Update of the Level Set Function when Empty Areas are Detected

We address the case that empty areas are detected in the places where obstacles were previously assumed. Thus, we assume that g_k monotonically decreases with the time sequence t_k. Following the discussion of Propositions 1 and 2, we can immediately conclude that the continuous level set value satisfies $Q_{k+1} \leq Q_k$ and that the corresponding discrete approximation satisfies $\mathcal{Q}_{k+1} \leq \mathcal{Q}_k$.

As in Section 4, we seek to identify the nodes that need to be recomputed. In parallel to Corollary 1, we infer the following.

Corollary 2 *Suppose* $\mathbf{g}_k \geq \mathbf{g}_{k+1}$. *Define the set*

$$\Theta := \{(l, m) \in \Psi : \mathbf{g}_{k+1}(l, m) < \mathbf{g}_k(l, m)\}.$$

Denote by Σ_{k+1} *the new graph representing the dependence of new level set values* $\mathcal{Q}_{k+1}$. *For a node* (i, j), *if* $(i, j) \notin \Theta$, *and if* (i, j) *will not become a child of an element in* Θ *with respect to the graph* Σ_{k+1}, *then* $\mathcal{Q}_{k+1}(i, j) = \mathcal{Q}_k(i, j)$.

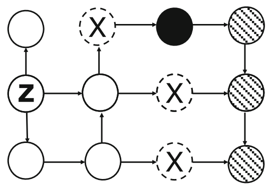

Fig. 6 The black node is detected as a new empty area and the grey nodes are its children in Σ_{k+1}

From Corollary 2, the nodes that need to be recomputed are the children of the nodes in Θ in the new value dependency graph Σ_{k+1}, as illustrated in Fig. 6. We assume that an obstacle disappears at the black node at time t_{k+1}. If the graph from Fig. 6 corresponds to the new value dependency graph Σ_{k+1}, the level set value of both black and grey nodes need to be recomputed. We also infer, from Corollary 2, among all the nodes (i, j) for which $\mathcal{Q}_k(i, j) > \mathcal{Q}_{k+1}(i, j)$, the node with the smallest value of $\mathcal{Q}_{k+1}$ is in Θ. Thus, if we update the level set function in the upwind direction we shall start recomputing the $\mathcal{Q}_{k+1}$ values for the nodes in Θ. In Fig. 6, the black node is recomputed first. The hurdle that prevents us from directly identifying the nodes that will become the children of Θ is that the new graph Σ_{k+1} remains unknown before we re-compute the new level set function. The following proposition allows us to start with $\mathcal{Q}_k$ and recompute the nodes that will become the children of Θ in the upwind directions.

Proposition 4 *Suppose* $\mathbf{g}_k \geq \mathbf{g}_{k+1}$, *and let* Σ_{k+1} *be the graph of the computational dependence corresponding to* $\mathcal{Q}_{k+1}$. *Consider a node* $(i, j) \in \Psi$ *satisfying*

$$\mathcal{Q}_{k+1}(i, j) < \mathcal{Q}_k(i, j).$$

Let (m, n) *be any neighbor of* (i, j). *We define the intermediate term* $\mathcal{V}(m, n)$ *satisfying*

$$\begin{cases} \mathcal{V}(m, n) = \mathcal{Q}_{k+1}(m, n), & \textit{if } (m, n) \textit{ is a direct parent of } (i, j) \textit{ in } \Sigma_{k+1}, \\ \mathcal{V}(m, n) \geq \mathcal{Q}_{k+1}(m, n), & \textit{otherwise}. \end{cases} \tag{31}$$

Then, the solution $\mathscr{Q}_{k+1}(i, j)$ corresponding to $\mathbf{g}_{k+1}$ *can be computed by solving $\mathscr{Q}_{k+1}(i, j)$ satisfying the following equation*

$$\max\left(\frac{\mathscr{Q}_{k+1}(i, j) - \min(\mathscr{V}(i-1, j), \mathscr{V}(i+1, j))}{\Delta x}, 0\right)^2 + \max\left(\frac{\mathscr{Q}_{k+1} - \min(\mathscr{V}(i, j+1), \mathscr{V}(i, j-1))}{\Delta x}, 0\right)^2 - \mathbf{g}^2_{k+1}(i, j) = 0, \quad \textit{if } (i, j) \neq (0, 0). \tag{32}$$

From Proposition 4, as long as the intermediate terms $\mathscr{V}$ satisfies Eq. 31 for each grid element, we can always correctly compute $\mathscr{Q}_{k+1}(i, j)$. To define such $\mathscr{V}$ values, we initially choose $\mathscr{V} = \mathscr{Q}_k$ for each node. During the re-computation in the upwind direction, for any grid element (i, j), we always replace $\mathscr{V}(i, j)$ with $\mathscr{Q}_{k+1}(i, j)$ as soon as it is available. This guarantees that the inequality in Eq. 31 always holds since $\mathscr{V}(i, j)$ value of a grid element is either $\mathscr{Q}_k(i, j)$ or $\mathscr{Q}_{k+1}(i, j)$. The equation in Eq. 31 holds too because in the upwind direction if a node (l, m) becomes a parent of (i, j), $\mathscr{Q}_{k+1}(l, m)$ value must be recomputed before $\mathscr{Q}_{k+1}(i, j)$. Thus, $\mathscr{V}(l, m)$ must be equal to $\mathscr{Q}_{k+1}(l, m)$ as we keep $\mathscr{V}$ values up to date. Therefore, from Eqs. 31 and 32, we infer that we can start from initially setting $\mathscr{Q}_k$ as $\mathscr{V}$ in Eq. 31 and recompute the nodes according to Eq. 32 in the upwind directions.

Proof of Proposition 4 The proof is indeed a direct application of the upwind property of the first order scheme 6 which supports the principle of the conventional fast marching method. We only detail the proof for Case 1. The proof for Case 2 follows a similar procedure. For notational simplicity, we define $E := (i, j)$ and neighbors of E by A, B, C and D as shown in Fig. 4a. Thus, from the first order scheme 6, we have

$$\max\left(\frac{\mathscr{Q}_{k+1}(E) - \min(\mathscr{Q}_{k+1}(A), \mathscr{Q}_{k+1}(C))}{\Delta x}, 0\right)^2 + \max\left(\frac{\mathscr{Q}_{k+1}(E) - \min(\mathscr{Q}_{k+1}(B), \mathscr{Q}_{k+1}(D))}{\Delta x}, 0\right)^2 - \mathbf{g}^2_{k+1}(E) = 0 \tag{33}$$

We assume that A is the only direct parent of E in the graph Σ_{k+1}. Thus, the solution of $\mathscr{Q}(E)$ satisfies

$$\mathscr{Q}_{k+1}(E) = \mathscr{Q}_{k+1}(A) + \mathbf{g}_{k+1}(E)\Delta x. \tag{34}$$

From Eq. 33, we infer

$$\mathscr{Q}_{k+1}(A) \leq \mathscr{Q}_{k+1}(C), \tag{35}$$

and

$$\mathscr{Q}_{k+1}(E) - \min(\mathscr{Q}_{k+1}(B), \mathscr{Q}_{k+1}(D)) \leq 0. \tag{36}$$

We must also have $\mathscr{Q}_{k+1}(A) < \mathscr{Q}_{k+1}(E)$, since A is the parent of E. Thus, according to Eq. 31, we obtain

$$\mathscr{Q}_{k+1}(A) = \mathscr{V}(A). \tag{37}$$

For the other neighbor nodes B, C and D, we infer, by the definition of $\mathscr{V}$ in Eq. 31, that

$$\mathscr{Q}_{k+1}(F) \leq \mathscr{V}(F), \quad \forall\, F \in \{B, C, D\}. \tag{38}$$

Thus, from Eqs. 35, 37, and 38, we infer that

$$\mathscr{V}(A) = \mathscr{Q}_{k+1}(A) \leq \mathscr{Q}_{k+1}(C) \leq \mathscr{V}(C). \tag{39}$$

From Eq. 39,

$$\max\left(\frac{\mathscr{Q}_{k+1}(E) - \min(\mathscr{V}(A), \mathscr{V}(C))}{\Delta x}, 0\right)^2 = \max\left(\frac{\mathscr{Q}_{k+1}(E) - \mathscr{V}(A)}{\Delta x}, 0\right)^2 \tag{40}$$

From Eqs. 36 and 38,

$$\begin{aligned}&\mathscr{Q}_{k+1}(E) - \min(\mathscr{V}(B), \mathscr{V}(D))\\ &\quad\leq \mathscr{Q}_{k+1}(E) - \min(\mathscr{Q}_{k+1}(B), \mathscr{Q}_{k+1}(D))\\ &\quad\leq 0.\end{aligned} \tag{41}$$

Given Eqs. 40 and 41, we conclude that the solution satisfying Eq. 32 is

$$\mathscr{Q}_{k+1}(E) = \mathscr{V}(A) + \mathbf{g}_{k+1}(E)\Delta x. \tag{42}$$

From Eqs. 37 and 34, we conclude the assertion. □

5.1 Algorithm

Following the above discussion, we now state the principal steps for efficiently updating the level set function. Note that from Corollary 2, the nodes need to be recomputed are either in Θ corresponding to the nodes detected as new empty area or will become the future children of Θ. Unlike the algorithm proposed in Section 4.2, we have to identify these nodes during the re-computation. Initially, we let the intermediate values $\mathscr{V} = \mathscr{Q}_k$ everywhere. Then, we start recomputing new intermediate values of $\mathscr{Q}_{k+1}$ for the nodes in Θ and list them in a queue. Selecting the node with the smallest intermediate value in this queue, we start re-computation in the upwind direction. Values of $\mathscr{V}$ are updated throughout the computation and we assign new values for $\mathscr{Q}_{k+1}$ as in Eq. 32. Due to Corollary 2, after we recompute the new value of a node, we check if this node becomes a child of Θ. If not, we do not propagate its value to the neighbors. In so doing, we stop propagating the new values of level set function at the nodes that are on the border between those who are the children of Θ and those who are not. For example, as shown in Fig. 6, the algorithm will stop propagating the level set value for the node marked as "X".

6 Algorithm in the Presence of Both New Obstacles and Empty Areas

In this section, we present the algorithm that addresses new obstacles and new empty areas simultaneously. Thus, we assume that $\mathbf{g}_k < \mathbf{g}_{k+1}$ for some nodes, and $\mathbf{g}_k > \mathbf{g}_{k+1}$ for some other nodes. The algorithm consists of two main steps respectively. Let Σ_k be the directed graph before the level set function is updated. First, we process the re-computation as if there were new obstacles only. Due to Corollary 1, we trace along the graph Σ_k from the nodes detected as obstacles ($\mathbf{g}_k < \mathbf{g}_{k+1}$) and set the intermediate values for both the new obstacle nodes and their children to an extremely large value. Instead of immediately recomputing the new level set values for new obstacles case, we use the intermediate values after the first step as an intermediate result. Since the intermediate values of both the new obstacle nodes and their children are very high, we assign the risk for traversal for these nodes to be extremely large but then decreased to $\mathbf{g}_{k+1}$. Thus, in the second step, we recompute the new level set values using the algorithm for the new empty areas in Section 5 by accounting for new empty areas nodes ($\mathbf{g}_k > \mathbf{g}_{k+1}$) and virtually treating the new obstacle nodes and their children as new empty areas from the intermediate step. Proposition 4 ensures that this algorithm gives correct solutions to the new Eikonal equation.

7 Computation Efficiency

Let N be the total number of nodes of a map. The computation cost for dynamic fast marching method $O(N \lg N)$ (see e.g., [23]). Although it is the same as that for the fast marching method [23], since the dynamic fast marching method only updates a portion of the entire map, the actual execution time is, in general, much smaller. We will show the difference in computational cost between the two methods in the next sections with some simulations.

One may compare the computation efficiency between the proposed method in this paper and the E* algorithm in [20]. Note that in terms of implementations, E* algorithm also identifies the portion of the nodes that requires re-computation and then employs the idea of the fast marching method for recomputing the new level set values. Therefore, the computation efficiency of both proposed method and E* algorithm are $O(N \lg N)$.

8 Illustrations

To illustrate the principal conclusions in this paper, we show two examples. First, in several canonical examples, we show how the level set function is influenced by unexpected obstacles, new empty areas, and when both cases happen. Second, we show a simulation result for an autonomous surface vehicle (ASV) navigating in a riverine environment. Note that there are two ways to find the optimal trajectory. The first is to compute the gradient direction by using the numerical solutions of level set function and the optimal path is along that gradient direction. The second method is to simply search for the smallest neighbor. The optimal path in all simulations in this paper is generated by searching the smallest neighbor.

8.1 Canonical Examples

To illustrate how an obstacle can influence the level set function, we show a canonical example for which an autonomous surface vehicle is navigating in an open area. Consider a 1000 m × 1000 m open area which is represented by a map whose grid size is 1 m × 1 m. We assume that the goal location is at the center of the square and that the cost function is $\mathbf{g}_0 = 1$ for each grid element except at the goal where $\mathbf{g}_0 = 0$. Figure 7 show the corresponding level set function $\mathcal{Q}_0$ before the detection of the new obstacles. We suppose that at time t_1 an obstacle is detected at location $[250\text{ m}, 250\text{ m}]^T$. Then, $\mathbf{g}_1 = 10^7$ at that location and $\mathbf{g}_1 = \mathbf{g}_0$ elsewhere. Figure 8 shows the difference between $\mathcal{Q}_0$ and $\mathcal{Q}_1$. The area included in the dash lines are where the level set function is recomputed because the corresponding nodes are children of the node that corresponds to the obstacle. Note that changes in value of the level set function are extremely small except in the area for which the obstacle occludes the target. It is this area in which an path would need to altered due to presence of the obstacle. It is worth noting that a deleted obstacle occurring at the same location as the obstacle in Fig. 8 causes same area of the level set function to be changed as that due to the added obstacle. This is because comparing Corollaries 1 and 2, the number of the nodes that are recomputed when a node is detected as new obstacle is equivalent to when the same node is known as an a priori obstacle but detected as empty.

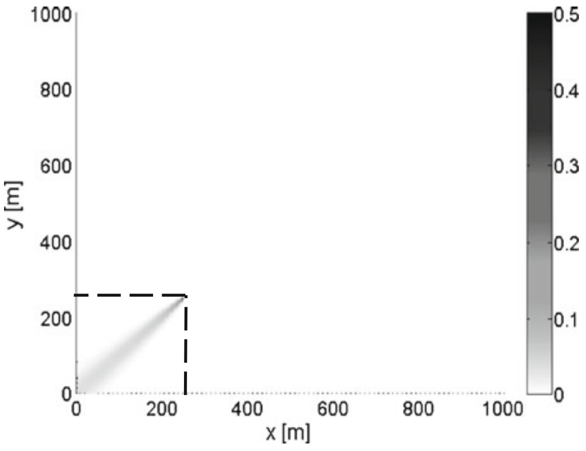

Fig. 8 The difference in the value of the level set function before and after the addition of an obstacle at location $[250\text{ m}, 250\text{ m}]^T$

In order to further illustrate the consequences of obstacle additions, removals, as well as when both cases happen simultaneously, we show a maze example. Figure 9 shows the original map of a maze and the corresponding trajectory from the start to the goal. Figures 10 and 11 show the cases of obstacle addition and removal respectively. The grey area in both picture corresponds to the domain for which the level set values are recomputed after using algorithms proposed in Sections 4 and 5 respectively. In comparison, Fig. 12 shows the case when both obstacle addition

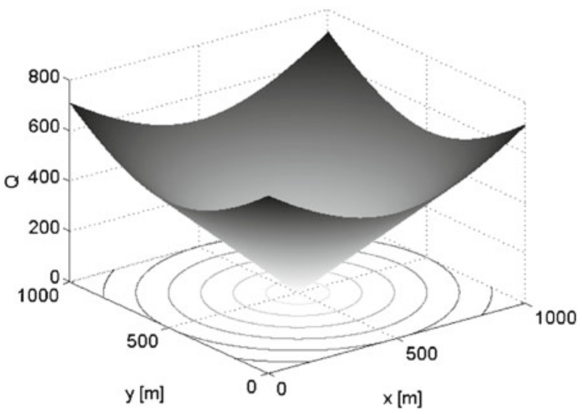

Fig. 7 Level set function before addition of an obstacle

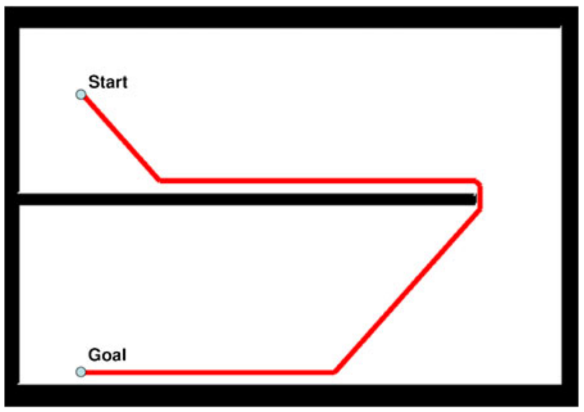

Fig. 9 Original maze and original trajectory

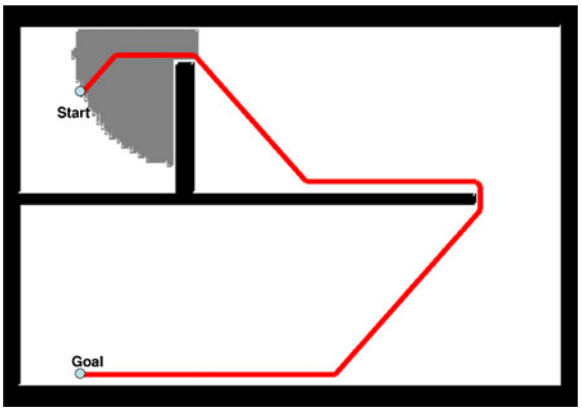

Fig. 10 Case of obstacle addition and corresponding trajectory

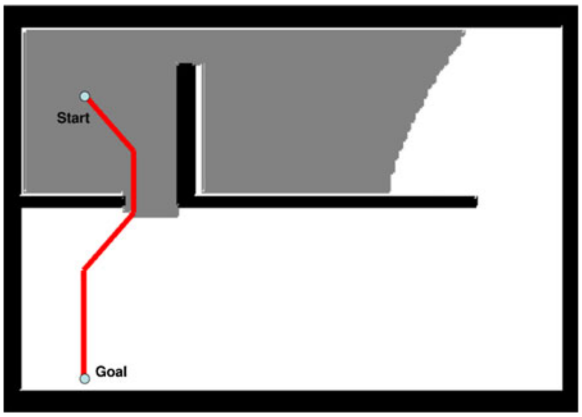

Fig. 12 Case for which both obstacle addition and removal happen simultaneously

and removal happen simultaneously. The level set function is recomputed by the proposed method in Section 6.

8.2 An Example for ASV Navigation

We show an example of an ASV navigating in a riverine environment. Figure 1 represents the a priori map which spans an area of 800 m × 600 m. In Fig. 13, the grey area represents obstacles that do not appear in the a priori map. The grid size is 3 m × 3 m. We assume that the ASV detects obstacles up to 35 m range. There are eight occasions where an obstacle is detected on the path and a portion of the level set function is recomputed. The location of each path update event is indicated in Fig. 13. Between the 5th and 7th update events, the ASV entered a U-shaped trap taking the route as a shortcut to the goal. The ASV computes a correct path to the goal at the 8th update event when it detects the dead end of the trap.

The principal tasks are to identify children of nodes for which the risk value has changed and to recalculate a subset of the level set function. Let H be the number of child nodes and let K be the number of the nodes that are recalculated. The computational cost to identify the children of obstacles is $O(H)$ (see e.g., [4, p. 477]) and the

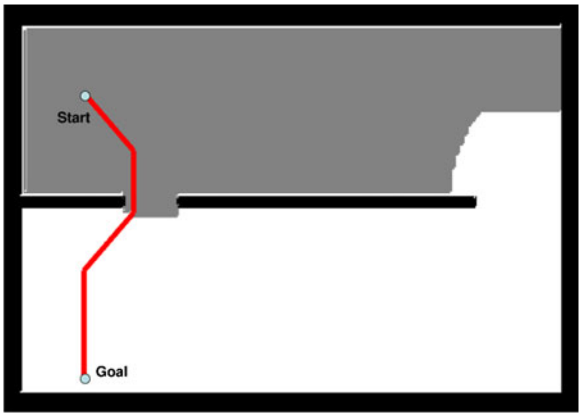

Fig. 11 Case of obstacle removal and corresponding trajectory

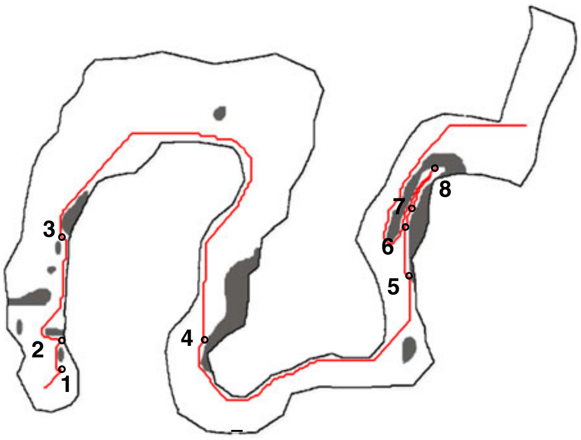

Fig. 13 The *grey colored areas* are some unexpect obstacles due to the inaccuracy and incompleteness of the a priori map in Fig. 1. The actual trajectory is marked by *red line*

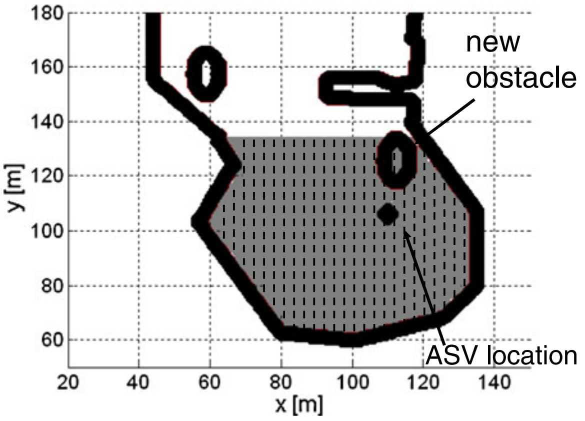

Fig. 14 The *shaded area* corresponds to the nodes for which values of the level set function need to be recomputed

cost to sort and recalculate K nodes is $O(K \lg K)$ (see e.g., [4, p. 140]). Since H and K depends on environment changes and the autonomous vehicle locations during the mission, the execution time varies for different scenarios. However, the newly detected obstacles are often close to the vehicle and thus the number of nodes K that need to be recomputed is small in most cases. For example, in Fig. 14, the shaded area shows the locations of the children of the newly detected obstacle. However, the values of level set function need to be recomputed for only $K < H$ of these nodes, as shown in Fig. 15. Figures 14 and 15 illustrate that only a small portion of the nodes are updated. For each update event shown in Fig. 13, Table 1 lists the number of the nodes which are detected as obstacles, children of the obstacles H and nodes which are recomputed K before the node corresponding to the location of the vehicle has been calculated. The last column of the table is the ratio between the number of the nodes that are recalculated and the total number of the nodes. This column shows that the proposed method significantly reduces the computation cost compared to computing the values of level sets function over the entire domain. Since the trap scenario at the 8th update is more complicated, the number of the nodes recalculated are correspondingly the largest among the eight update events in Table 1.

Table 1 Computation cost of dynamic fast marching method for ASV navigation example

Update	Number of obstacles	Children of obstacles (H)	Nodes recalculated (K)	Percentage (K/N) (%)
1	109	443	119	1.01
2	142	251	141	1.19
3	757	1,132	173	1.47
4	811	6,105	144	1.22
5	413	2,029	973	8.26
6	306	1,211	117	0.99
7	365	420	100	0.85
8	558	420	1,940	16.48

Total number of nodes (N): 11,775

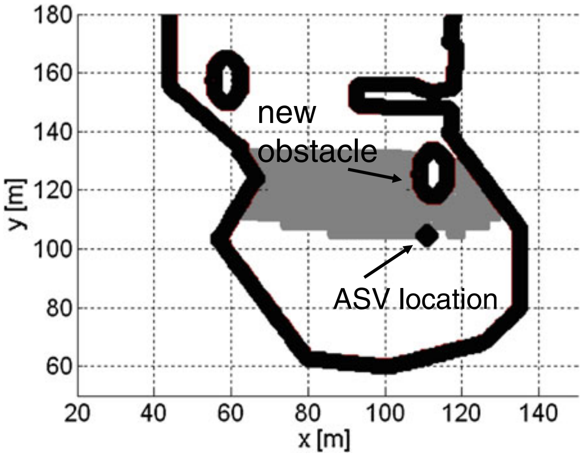

Fig. 15 The *grey area* corresponds to the nodes for which values of the level set function are recomputed

9 Concluding Remarks

This paper proposes an efficient algorithm that uses a level set method to replan paths. Although the fast marching method is a method for computing a solution to the Eikonal equation, even this method can be prohibitively slow for very large areas. The proposed method reduces computation expenses in two aspects. First, we do not update the level set function unless there are obstacles on the current optimal trajectory of the vehicle. Second, when we update level set values, we recompute only a portion of the level set function. In order to justify the proposed method, we provide formal analysis of how level sets values change when new obstacles or new empty areas are detected.

References

1. Alton, K., Mitchell, I.M.: Optimal path planning under different norms continuous state spaces. In: Proc. of IEEE International Conf. on Robotics and Automation, pp. 866–872 (2006)
2. Choi, W., Zhu, D., Latombe, J.C.: Contingency-tolerant robot motion planning and control. In: Proc. of IEEE/RSJ International Workshop on Intelligent Robots and Systems, pp. 78–86 (1989)
3. Cohen, L.D., Kimmel, R.: Global minimum for active contours models: a minimal path approach. Int. J. Comput. Vis. **24**(1), 57–78 (1997)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press, Cambridge (1989)
5. Elfes, A.: Using occupancy grids for mobile robot perception and navigation. Computer **22**(6), 46–57 (1989)
6. Ferguson, D., Likhachev, M., Stentz, A.: A guide to heuristic path planning. In: Proc. of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (2005)
7. Hassouna, M.S., Abdel-Hakim, A.E., Farag, A.A.: Robust robotic path planning using level sets. In: Proc. of IEEE International Conf. on Image Processing, vol. 3, pp. 473–476 (2005)
8. Khatib, M., Jaouni, H., Chatila, R., Laumond, J.P.: Dynamic path modification for car-like nonholonomic mobile robots. In: Proc. of IEEE International Conf. on Robotics and Automation, pp. 2920–2925 (1997)
9. Kimmel, R., Sethian, J.A.: Optimal algorithm for shape from shading and path planning. J. Math. Imaging Vis. **14**, 237–244 (2001)
10. Kimmel, R., Amir, A., Bruckstein, A.M.: Finding shortest paths on surfaces using level set methods. IEEE Trans. Pattern Anal. Mach. Intell. **17**(6), 635–640 (1995)
11. Koenig, S., Likhachev, M.: Improved fast replanning for robot navigation in unknown terrain. Technical Report GIT-COGSCI-2002/3, College of Computing, Georgia Institute of Technology, Atlanta, GA (2002)
12. Krogh, B.H., Thrope, C.E.: Integrated path planning and dynamic steering control for autonomous vehicles. In: Proc. of IEEE International Conf. on Robotics and Automation, pp. 1664–1669 (1986)
13. Lamiraux, F., Bonnafous, D., Lefebvre, O.: Reactive path deformation for nonholonomic mobile robots. IEEE Trans. Robot. **2**(6), 967–977 (2004)
14. Latombe, J.C.: Robot Motion Planning. Kluwer Academic, Norwell (1991)
15. LaValle, S.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
16. Lions, P.L.: Generalized Solutions of Hamilton–Jacobi Solutions. Pitman, New York (1982)
17. Mitchell, I.M., Sastry, S.: Continuous path planning with multiple constraints. In: Proc. of the 42nd IEEE Conf. on Decision and Control, pp. 5502–5507 (2003)
18. Osher, S.J., Sethian, J.A.: Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations. J. Comput. Phys. **79**, 12–49 (1988)
19. Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J., Lane, D.: Path planning for autonomous underwater vehicles. IEEE Trans. Robot. **23**(2), 331–341 (2007)
20. Phillippsen, R.: A light formulation of the E* interpolated path replanner. Technical Report, Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne, Switzerland (2006)
21. Quinlan, S., Khatib, O.: Elastic bands: connecting path planning and control. In: Proc. of IEEE International Conf. on Robotics and Automation, pp. 802–807 (1993)
22. Rouy, E., Tourin, A.: A viscosity solutions approach to shape-from-shading. SIAM J. Numer. Anal. **29**(3), 867–884 (1992)
23. Sethian, J.A.: Fast marching method. SIAM Rev. **41**(2), 199–235 (1999)
24. Sun, X., Yeoh, W., Koenig, S.: Dynamic fringe-saving A*. In: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 891–898 (2009)
25. Tsitsiklis, J.N.: Efficient algorithm for globally optimal trajectories. IEEE Trans. Automat. Contr. **40**(9), 1528–1538 (1995)
26. Xu, B., Stilwell, D.J., Kurdila, A.J.: Efficient computation of level sets for path planning. In: IEEE/RSJ International Conf. on Intelligent Robots and Systems (2009)