The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Efficient Computation of Level Sets for Path Planning

Bin Xu, Daniel J. Stilwell and Andrew Kurdila

Abstract— **We propose an efficient method for updating a path that was computed using level-set methods. Our approach is suitable for autonomous vehicles navigating in a static environment for which an a priori map of the environment is inaccurate. When the autonomous vehicle detects a new obstacle, our algorithm replans an optimal route without recomputing the entire path. Computational costs when planning paths with level set methods are due to creation of the level set. Once the level set has been computed, the optimal path is simply gradient descent down the level set. Our approach is based on formal analysis of how the level set changes when a new obstacle is detected. We show that in many practical cases, only a small portion of the level set needs to be re-computed when a new obstacle is detected. Simulation examples are presented to validate the effectiveness of the proposed method.**

I. 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 inconsistency between the *a priori* map and actual environment, path replanning is required in order to avoid collisions with obstacles that do not appear on the *a prior* map.

In this paper, our goal is to design a minimal risk path replanning method that decreases the computational costs of using level-set path planning when new obstacles are detected. The proposed method does not account for vehicle dynamics or otherwise addresses path-following limitations. Thus our approach is suited to vehicles that can follow arbitrary paths at potentially slow speeds. This includes certain classes of autonomous surface vehicles, which motivates our work, but also includes classes of ground vehicles and ground hovercrafts. We presume that the environment is represented by a two dimensional uniform-sized occupancy grid map [5] which is initially generated from the *a priori* knowledge of the environment. Obstacles in the environment are detected by on-board sensors that have a limited range. Because the higher occupied probability induces the higher risk for the vehicle to traverse, we can associate a cost function to each grid proportional to the occupied probability indicating the risk for traversal.

Path planning for autonomous vehicles has been studied for decades. Excellent references can be found in [13] and [14]. These methods can be grouped into two categories: local and global replanning. The former considers local environment changes and plans locally to find a collision free path, for example, [2], [8], [11], [12], and [21]. These local planning methods are effective when environment changes are small but they often fail in trap scenarios for which the planner can not make progress toward the desired endpoint due to the lack of global knowledge about the environment. Global replanning can avoid these problems since it takes account of the entire map to find a new path, although global replanning can be computationally expensive. 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. [13], pp.604). The map is modeled by nodes representing sites associated with cost to traverse. Thus, finding an optimal path is treated as an minimal cost path searching problem in graph [24]. Upon the change of environments, the costs to traverse the corresponding nodes will change, and the overall minimal cost to travel from a given node to the goal is consistently updated.

Our approach to path planning is based on level-set methods, which compute minimum risk paths. The minimal risk path problem is modelled as a partial differential equation (PDE) ([3], [9] and [10]). The value of the level sets at each point indicates the overall minimal risk to travel from that point to the goal, and the optimal trajectory is along the gradient of the level sets. The solution of the partial differential equation can be approximated by the fast marching method (FMM) [23]. The method has been successfully applied to path planning when *a priori* maps are accurate, for example, in [7], [9], [16] and [19]. There is limited literature that discusses the level set for replanning paths in a partially known environment. In [20], an E* Lite algorithm is proposed which locally updates level sets at the nodes upon the environment change. The qualitative comparison between A* search and level set methods can be found in [1].

In this paper, we propose a dynamic fast marching method which modifies the original fast marching method [23] such that the new paths are replanned more efficiently upon changes to the environment. Our analysis addresses only the case that the vehicle detects unexpected obstacles. These are obstacles that are detected during the mission, but do not appear on the *a prior* map. For the case that the unexpected empty areas are detected, one would not be required to compute a new path. The proposed path replanner reduces the computation 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 obstacle-free areas, the trajectory remains optimal. This observation allows us to delay computation of the level-set 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

Bin Xu and Daniel J. Stilwell are with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, 24061, USA. bxu, stilwell@vt.edu

Andrew J. Kurdila is with the Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, 24061, USA. kurdila@vt.edu

level sets needs to be recomputed. The proposed algorithm draws heavily from the ideas in [20], and is very similar in implementation. Our contribution is to present rigorous and formal analysis of how changes in the environment produce corresponding changes in the level set, and how these changes yield can be used to reduce computational burden of using level-set methods for path planning. As illustrated by numerical simulations in Section VIII, the proposed algorithm can find new paths with low computation cost, especially for environments similar to riverine systems, which motivates our work, and which we address in Section VIII.

The rest of the paper is organized as follows. In Section II, we introduce the preliminary results of using the level set method for path planning problems. In Section III, we define the obstacle detection and formulate the path replanning problem. In Section IV, we propose path replanning strategy. In both Section V and VI, we show the development of the proposed replanning strategy. Simulation results are illustrated in Section VIII.

II. Preliminaries

In this part, we introduce the level set method for path planning that is originally developed in [9], [17] and [23].

A. Eikonal Equation and Level Sets

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 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 $\mathbf{z}$

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

where $\mathbf{c} \in Lip([0,1]; \overline{\Omega})$ is a parameterized Lipschitz continuous path with, $\mathbf{c}(0) = \boldsymbol{\xi}$ being a current starting point, and $\mathbf{c}(1) = \mathbf{z}$ being the goal (see e.g. [15], pp. 116).

From (1) and from the definition of g, a simple 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],

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

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, 2 and 3 show an example of *a priori* map and its corresponding contours of level sets with respect to targets.

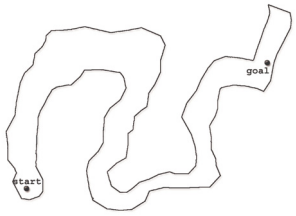

Fig. 1. An *a priori* map $\overline{\Omega}$ for a riverine environment.

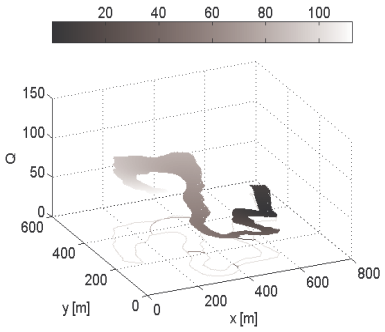

Fig. 2. The level sets for the *a priori* map.

1) Finite Difference Scheme and Its Properties: The Eikonal equation (2) often cannot be solved analytically. In [23], a first order update scheme is proposed which approximates the viscosity solution of (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

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

Correspondingly, we approximate the g with $\mathbf{g}: \;\; \Psi \to \mathbb{R}^1$ 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 $(i,j) \in \Psi$ to be the set of grids $(i+1,j)$, $(i-1,j)$, $(i,j+1)$ and $(i,j-1)$. If grid (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 (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

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

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

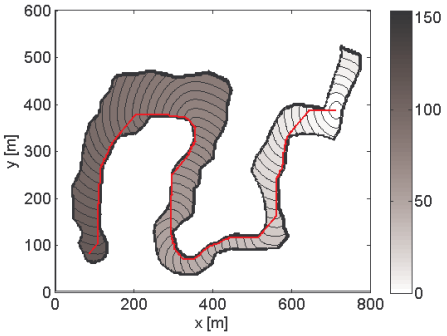

Fig. 3. The level sets contour and the optimal path for the *a priori* map.

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

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

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

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

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

$$\mathcal{Q}(i,j) = \min(a,b) + \mathbf{g}(i,j)\Delta x \tag{7}$$

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

$$(\mathcal{Q}(i,j)-a)^2 + (\mathcal{Q}(i,j)-b)^2 - \mathbf{g}^2(i,j)\Delta x^2 = 0 \tag{8}$$

that is

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

Both (7) and (9) show that for each grid (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 ones. 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 (6) would be negative.

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

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

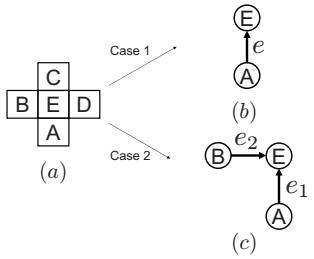

Fig. 4. (a) Grid E and its neighbors; (b) Graph Σ for *Case 1*; and (c) Graph Σ for *Case 2*

III. Problem Formulation

A. Obstacle Detection

Consider the possible presence of obstacles due to inconsistency between the *a priori* map and the actual environment. We denote the set of locations of newly detected obstacle by an open set $O \subset \mathbb{R}^2$ which satisfies $O \subseteq \overline{\Omega}$. Let the detection range of an onboard sensor be r. It takes measurements periodically with period h. Letting

$$t_k = t_0 + kh, \forall k = 0, 1, \ldots, \tag{10}$$

we denote the obstacle detected at time t_k, by the closure of an open subset $O(k)$ satisfying

$$O(k+1) = \cup_{t\in[t_k,t_{k+1}]} B_r(\mathbf{x}(t)) \cap O \tag{11}$$

where $B_r(\mathbf{x}(t))$ is an open ball of radius r centered at the vehicle's current position $\mathbf{x}(t)$. $O(k+1) = \emptyset$ indicates that the vehicle does not detect any obstacle at time t_{k+1}.

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 there is no new obstacle detected at time t_{k+1}, we let $g_{k+1} = g_k$ for the entire domain $\overline{\Omega}$. Otherwise, if $O(k+1) \neq \emptyset$, we let g_{k+1} be such that $g_{k+1}(\boldsymbol{\xi}) > g_k(\boldsymbol{\xi})$ for $\boldsymbol{\xi} \in O(k+1)$ and $g_{k+1}(\boldsymbol{\xi}) = g_k(\boldsymbol{\xi})$ for $\boldsymbol{\xi} \in \overline{\Omega} \setminus O(k+1)$. Thus, the new Eikonal equation that is induced by new cost functions g_k at time t_k becomes

$$\|\nabla Q_k(\boldsymbol{\xi})\| = g_k(\boldsymbol{\xi}), \; Q_k(\mathbf{z}) = 0. \tag{12}$$

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

B. Problem Statement

One way to update the solution of Eikonal equation (12) is to recalculate level sets over the entire domain $\overline{\Omega}$ with respect to the new cost function g_k. For the purpose of real-time application, we propose a new path replanning method such that the solutions of Eikonal equations can be more efficiently updated upon the detection of obstacles.

IV. Path Replanning Strategy

The main features of the path replanning strategy are that we do not need to recompute the optimal path unless an obstacle intersects the path, and when a path is recomputed, we recompute only those nodes that might change value and ignore all other nodes. In the case that newly detected obstacles are near the autonomous vehicle, it is often the case that most grid elements do not need to be recomputed.

Note that we address the path replanning in the presence of unexpected obstacles. Thus, we assume that g_k monotonically increases with the time sequence t_k.

V. Optimality of Trajectories in the Presence of New Obstacles

We show that an optimal path remain 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 level set only when a new obstacle intersects the path. For notational simplicity, without further specification, $g_{k+1} \geq g_k$ means that, for any $\xi \in \overline{\Omega}$, $g_{k+1}(\boldsymbol{\xi}) \geq g_k(\boldsymbol{\xi})$. Correspondingly, similar meaning can be deduced for $\mathbf{g}_{k+1} \geq \mathbf{g}_k$ in discrete case.

Proposition 1: Suppose Q_k and Q_{k+1} are the solutions of (12) for the cost functions g_k and g_{k+1}, respectively. 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: [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 $\mathbf{c}(p) \in Lip(I; \overline{\Omega})$ by

$$J_k(\mathbf{c}) = \int_I g_k(\mathbf{c}(p)) \left\|\mathbf{c}'(p)\right\| dp \tag{13}$$

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

$$Q_k = \min_{\mathbf{c} \in Lip(I;\overline{\Omega})} J_k(\mathbf{c}), \tag{14}$$

and

$$Q_{k+1} = \min_{\mathbf{c} \in Lip(I;\overline{\Omega})} J_{k+1}(\mathbf{c}). \tag{15}$$

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

$$L = \{p \in I : \ \mathbf{c}(p) \subseteq \overline{O(k+1)}\}. \tag{16}$$

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

$$\begin{aligned} \Delta J(\mathbf{c}) =& J_{k+1}(\mathbf{c}) - J_k(\mathbf{c}) \\ =& \int_{I \setminus L} \left(g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))\right) \left\|\mathbf{c}'(p)\right\| dp \\ &+ \int_L \left(g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))\right) \left\|\mathbf{c}'(p)\right\| dp \\ =& \int_L \left(g_{k+1}(\mathbf{c}(p)) - g_k(\mathbf{c}(p))\right) \left\|\mathbf{c}'(p)\right\| dp \end{aligned} \tag{17}$$

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

$$J_{k+1}(\mathbf{c}) \geq J_k(\mathbf{c}) \tag{18}$$

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 (17), since $L = \varnothing$, for the path along $\mathbf{c}^*$

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

Since $\mathbf{c}^*$ is the optimal path given g_k, $J_k(\mathbf{c}^*) \leq J_k(\mathbf{c})$. Together with (18), we conclude that for an arbitrary curve $\mathbf{c}$,

$$J_{k+1}(\mathbf{c}^*) = J_k(\mathbf{c}^*) \leq J_k(\mathbf{c}) \leq J_{k+1}(\mathbf{c}) \tag{20}$$

The inequality (20) indicates the path $\mathbf{c}^*$ is the optimal for the cost function $\mathbf{g}_{k+1}$. ■

VI. Dynamic Fast Marching Method

In this section, we show the development of the proposed dynamic fast marching method. 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 II-A.3, we show that level sets do not necessarily need to be updated everywhere, and we identify the group of grids whose level sets should be updated.

A. The Approximation of Level Sets in Discrete Space

Proposition 1 indicates that the values of level sets monotonically increases if the cost function monotonically increases. We now investigate the corresponding property for the discrete approximation of the level set. The update scheme (6) uses the first order finite difference approximation which introduces approximation errors. It remains a question whether there exist some nodes (i, j) whose approximations are such that $\mathcal{Q}_k(i, j) > \mathcal{Q}_{k+1}(i, j)$ due to approximation errors. With Proposition 2 and 3, we identify the area for which level sets update is necessary. Specifically, we 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 satisfies $\mathcal{Q}_k \leq \mathcal{Q}_{k+1}$.

Proposition 2: Given $\mathbf{g}_k$ and $\mathbf{g}_{k+1}$, let $\mathcal{Q}_k$ and $\mathcal{Q}_{k+1}$ be the discrete solutions to (12) on the same grid with mesh size

Δx. If $\mathbf{g}_{k+1} \geq \mathbf{g}_k$, the approximation satisfies $\mathcal{Q}_{k+1}(i,j) \geq \mathcal{Q}_k(i,j)$ for all grids (i,j).

Proposition 2 is a direct consequence of the numerical algorithm proposed in [22]. We replace $\mathcal{Q}$ and $\mathbf{g}$ in the left-hand side of (6) with $\mathcal{Q}_k$ and $\mathbf{g}_{k+1}$ respectively. Following the algorithm in [22], we can construct a convergent sequence $\mathcal{V}_n$, where $n = 1, 2, \ldots$, $\mathcal{V}_1 = \mathcal{Q}_k$ and $\lim_{n\to\infty} \mathcal{V}_n = \mathcal{Q}_{k+1}$. Since we can show by the algorithm in [22] that $\mathcal{V}_n$ is monotonically increasing, we conclude that $\mathcal{Q}_{k+1}(i,j) \geq \mathcal{Q}_k(i,j)$ for all grids (i,j).

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 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 sets do not necessarily need to be updated at all grids elements, and we can identify the nodes for which the level sets update is necessary.

Proposition 3: Denote the computational dependence between nodes for fast marching algorithm by the graphs Σ_k and Σ_{k+1} for times t_k and t_{k+1} respectively. Assume 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 $\mathcal{Q}_k(l,m) = \mathcal{Q}_{k+1}(l,m)$ where node (l,m) is any direct parent of node (i,j), 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: [Proof of Proposition 3] We only detail the proof for *Case 1*. 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 Figure 4 (a). We assume that A is the only direct parent of E. Thus,

$$\mathcal{Q}_k(A) \leq \mathcal{Q}_k(C), \tag{21}$$

and

$$\min(\mathcal{Q}_k(B), \mathcal{Q}_k(D)) - \mathcal{Q}_k(A) \geq \mathbf{g}_k(E)\Delta x. \tag{22}$$

By Proposition 2, $\mathcal{Q}_{k+1} \geq \mathcal{Q}_k$ for all the neighbor nodes B, C and D. Thus, since $\mathcal{Q}_{k+1}(A) = \mathcal{Q}_k(A)$ and $\mathbf{g}_{k+1}(E) = \mathbf{g}_k(E)$, from (21) and (22) we have

$$\mathcal{Q}_{k+1}(A) \leq \mathcal{Q}_{k+1}(C), \tag{23}$$

and

$$\min(\mathcal{Q}_{k+1}(B), \mathcal{Q}_{k+1}(D)) - \mathcal{Q}_{k+1}(A) \geq \mathbf{g}_{k+1}(E)\Delta x. \tag{24}$$

The above two inequalities imply

$$\begin{aligned} \mathcal{Q}_{k+1}(E) =& \mathcal{Q}_{k+1}(A) + \mathbf{g}_{k+1}(E)\Delta x \\ =& \mathcal{Q}_k(A) + \mathbf{g}_k(E)\Delta x \\ =& \mathcal{Q}_k(E) \end{aligned} \tag{25}$$

which completes the proof for *Case 1*. ■

Consider an arbitrary Σ_k of a level set. Applying *Proposition 3* from the node corresponding to the target $\mathbf{z}$, the following corollary follows.

Corollary 1: Define the set

$$\Upsilon := \{(i,j) \in \Psi : \ \mathbf{g}_{k+1}(i,j) > \mathbf{g}_k(i,j)\}. \tag{26}$$

For a node (i,j) and given the graph Σ_k, $\mathcal{Q}_{k+1}(i,j) > \mathcal{Q}_k(i,j)$ only if $(i,j) \in \Upsilon$, or, in graph Σ_k, (i,j) is a child of a grid in Υ.

B. Dynamic Fast Marching Method

We now describe the proposed method to update level sets. 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. pp. 477, [4]). The second step is to recompute $\mathcal{Q}_{k+1}$ value 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 is not needed. Since obstacles are detected near the autonomous vehicle, the number nodes that must be recalculated is often very small.

VII. 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.

VIII. Illustrations

To illustrate the principal conclusions in this paper, we show a simulation result for an autonomous surface vehicle (ASV) navigating in a riverine environment. Figure 1 and 5 respectively represent the *a priori* map and actual environment both of which span an area of $800m \times 600m$. In Figure 5, the grey area is the obstacles that do not appear in the *a priori* map. The grid size is $3m \times 3m$. During the entire mission, the ASV detected obstacles within about 35 meters range and the proposed method updated level sets a total of eight times. The corresponding locations of where updates occurred are marked in sequence in Figure 5. As shown in Figure 5, the ASV traveled along an obstacle free trajectory. Note that between the 5th to 7th updates, the ASV entered a U-shaped trap taking the route as a shortcut to the goal. But, as soon as the vehicle detected the dead end of the trap, the vehicle managed to escape after the 8th update.

The majority of computation is spent on identifying children nodes of obstacles and recalculating level set values for a subset of them. Let H be the number of child nodes and let K be the number of the nodes that are recalculated.

The computation cost to identify the children of obstacles is $O(H)$ (see e.g., pp. 477 [4]) and the cost to sort and recalculate K nodes is $O(K \lg K)$ (see e.g., pp.140 [4]). 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 each update, Table I 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 nodes recalculated and the total number of the nodes. This column shows that the proposed method significantly reduces the computation cost compared to computing the level sets values 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 updates in Table I.

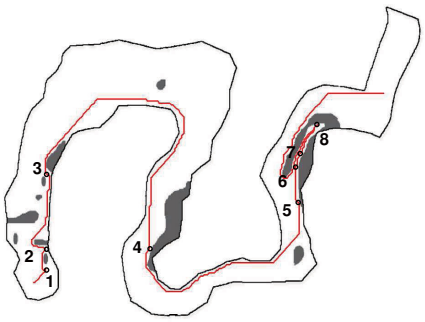

Fig. 5. The grey colored areas are some unexpect obstacles due to the inaccuracy and incompleteness of the *a priori* map in Figure 1. The actual trajectory is marked by red line.

TABLE I

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	1132	173	1.47%
4	811	6105	144	1.22%
5	413	2029	973	8.26%
6	306	1211	117	0.99%
7	365	420	100	0.85%
8	558	420	1940	16.48%
Total Number of Nodes (N): 11775				

IX. CONCLUDING REMARKS

This paper proposes an efficient algorithm that uses the level set method to replan paths. To find an obstacle free path upon the detection of new obstacles, one needs to update level sets which can be computationally expensive if we use the conventional fast marching method. The proposed method reduces computation expenses in two aspects. First, we do not update level sets unless there are obstacles on the current optimal trajectory of the vehicle. Second, when we update level sets, we recompute only a portion of them. In order to justify the proposed method, we provide formal analysis of how level sets change when new obstacles are detected. In the end, simulations are presented to validate the effectiveness of the proposed method.

REFERENCES

[1] K. Alton and I.M. Mitchell, "Optimal path planning under different norms continuous state spaces," *Proc. of IEEE International Conf. on Robotics and Automation*, pp.866-872, 2006.

[2] W. Choi, D. Zhu and J.C. Latombe, "Contingency-tolerant robot motion planning and control," *Proc. of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pp.78-86, 1989.

[3] L.D. Cohen and R. Kimmel, "Global minimum for active contours models: a minimal path approach," *International Journal of Computer Vision*, vol.24, no.1, pp.57-78, 1997.

[4] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to algorithms*, The MIT Press, Cambridge, MA, 1989.

[5] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol.22, no.6, pp.46-57, 1989.

[6] D. Ferguson, M. Likhachev and A. Stentz, "A guide to heuristic path planning," *Proc. of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling*, 2005.

[7] M.S. Hassouna, A.E. Abdel-Hakim and A.A. Farag, "Robust robotic path planning using level sets," *Proc. of IEEE International Conf. on Image Processing*, vol.3, pp.473-476, 2005.

[8] M. Khatib, H. Jaouni, R. Chatila, J.P. Laumond, "Dynamic path modification for car-like nonholonomic mobile robots," *Prof. of IEEE International Conf. on Robotics and Automation*, pp.2920-2925, 1997.

[9] R. Kimmel and J.A. Sethian, "Optimal algorithm for shape from shading and path planning," *Journal of Mathematical Imaging and Vision*, vol. 14, pp.237-244, 2001.

[10] R. Kimmel, A. Amir and A.M. Bruckstein, "Finding shortest paths on surfaces using level set methods," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.17, no.6, pp.635-640, 1995.

[11] B.H. Krogh and C.E. Thrope, "Integrated path planning and dynamic steering control for autonomous vehicles," *Proc. of IEEE International Conf. on Robotics and Automation*, pp.1664-1669, 1986.

[12] F. Lamiraux, D. Bonnafous and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," *IEEE Trans. on Robotics*, vol.2, no.6, pp.967-977, 2004.

[13] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Norwell, MA, 1991.

[14] S. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

[15] P.L. Lions, *Generalized Solutions of Hamilton-Jacobi Solutions*, Pitman Publishing INC, 1982.

[16] I.M. Mitchell and S. Sastry, "Continuous path planning with multiple constraints," *Proc. of the 42nd IEEE Conf. on Decision and Control*, pp.5502-5507, 2003.

[17] S.J. Osher and J.A. Sethian, "Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, 79, pp.12-49, 1988.

[18] A. Papoulis and S.U. Pillai, *Probability, Random Variables and Stochastic Process, 4th Edition*, McGraw Hill, 2002.

[19] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Trans. on Robotics*, vol.23, no.2, pp.331-341, 2007.

[20] R. Phillippsen, "A light formulation of the E* interpolated path replanner," *Technical Report*, Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne, Switzerland, 2006.

[21] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," *Proc. of IEEE International Conf. on Robotics and Automation*, pp.802-807, 1993.

[22] E. Rouy and A. Tourin, "A viscosity solutions approach to shape-from-shading," *SIAM Journal on Numerical Analysis*, vol.29, no.3, pp.867-884, 1992.

[23] J.A. Sethian, "Fast marching method," *SIAM Review*, vol.41, No.2, pp.199-235, 1999.

[24] J.N. Tsitsiklis, "Efficient algorithm for globally optimal trajectories," *IEEE Trans. on Automatic Control*, vo. 40, no. 9, pp. 1528–1538, 1995.