sensors

MDPI

Article

Robot Navigation Based on Potential Field and Gradient Obtained by Bilinear Interpolation and a Grid-Based Search

Gregor Klančar [1] and Andrej Zdešar [1] and Mohan Krishnan [2,*]

[1] Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia; gregor.klancar@fe.uni-lj.si (G.K.); andrej.zdesar@fe.uni-lj.si (A.Z.)
[2] Electrical & Computer Engineering and Computer Science Department, University of Detroit Mercy, Detroit, MI 48208, USA
* Correspondence: mohank@udmercy.edu

Abstract: The original concept of the artificial potential field in robot path planning has spawned a variety of extensions to address its main weakness, namely the formation of local minima in which the robot may be trapped. In this paper, a smooth navigation function combining the Dijkstra-based discrete static potential field evaluation with bilinear interpolation is proposed. The necessary modifications of the bilinear interpolation method are developed to make it applicable to the path-planning application. The effect is that the strategy makes it possible to solve the problem of the local minima, to generate smooth paths with moderate computational complexity, and at the same time, to largely preserve the product of the computationally intensive static plan. To cope with detected changes in the environment, a simple planning strategy is applied, bypassing the static plan with the solution of the A* algorithm to cope with dynamic discoveries. Results from several test environments are presented to illustrate the advantages of the developed navigation model.

Keywords: robot navigation; path planning; potential field; bilinear interpolation; dynamic local re-planning

Citation: Klančar, G.; Zdešar, A.; Krishnan, M. Robot Navigation Based on Potential Field and Gradient Obtained by Bilinear Interpolation and a Grid-Based Search. *Sensors* **2022**, *22*, 3295. https://doi.org/10.3390/s22093295

Academic Editor: Andrey V. Savkin

Received: 4 April 2022
Accepted: 23 April 2022
Published: 25 April 2022

1. Introduction

The main goal of a navigation function is to create feasible, safe paths that avoid obstacles and allow a robot to move from its start configuration to its goal configuration [1]. Online robot navigation and path planning consists of two complementary aspects. In the global path-planning phase, the task is to find an optimal path to the intended goal, starting from the robot's starting position and using all the previous information about the environment. This plan takes into account the need to avoid obstacles but only those that are assumed to be present before the robot starts to move towards the goal. This is coupled with the local obstacle avoidance phase in which the robot avoids new obstacles detected by its sensors while navigating the planned path. The former can be thought of as proactive, while the latter is reactive. With this understanding comes the acceptance that the former is usually more optimal than the latter in some sense. The biggest challenge in real-world applications is the ability to handle unanticipated changes in both structured and unstructured environments. Because the discovery of new obstacles is an evolutionary process, it cannot be assumed that the overall path that is eventually completed will be as optimal as a path that was planned with the knowledge of all the aspects of an unchanged environment at the beginning. However, this is not a fair comparison because new map situations are usually discovered after the robot has begun to execute the original plan. Moreover, due to these new discoveries, the robot may find itself in situations where it seems to be trapped if it continues to follow the global plan while only imposing a requirement to avoid collisions with the new obstacles. The goal is then to develop strategies to overcome the new obstacles in an effective and situationally appropriate way while the robot continues to head towards its original goal.

It is important to understand the trade-off involved in the above situation, which can be thought of as a see-and-react strategy when dealing with new discoveries. In contrast to this approach, there is the alternative of re-implementing the global path-planning strategy at the point where new environmental discoveries are made. However, implementing a full-featured online global path-planning strategy is usually not feasible due to computational costs. Speed is essential because potential delays in reacting could affect the ability to deal with new discoveries safely and efficiently unless one is willing to slow down or stop until a new plan is available. This is particularly true when considering that changes in the environment relative to prior assumptions are quite likely in actual navigation applications. It is to address this need that several incremental methods have been developed [2–6], which reduce computational and storage costs by reusing existing planning information.

An extensive survey of path planning algorithms has been carried out in [7]. Algorithms are divided into categories and sub-categories within them, based on the modality of their development. This work falls in the sub-category of graph search , which supports a variety of path-planning approaches, but here specifically, a graph search based on the use of a uniform grid. It combines this with the adoption of a variant of the original artificial potential fields (APF) method [8]. According to [7], the APF is categorised as reactive manoeuvring. Thus, essentially, a uniform grid-based graph search is combined with a reactive manoeuvring technique to carry out global and local path planning.

The environment can be represented as a graph using cell decomposition or roadmaps. Examples of the latter approach are the Voronoi graph [9], which can produce optimum clearance from the obstacles, and the tangent graph [10], which contains the optimal solution and requires less memory than the visibility graph, its superset. In [11], a tangent graph is constructed for obstacles described with analytic curves in which a finite search algorithm can be used to find the optimal path. The optimal path found in the tangent graph may not be smooth. The authors in [11] combine the tangent graph with online reactive navigation to generate smooth paths for the unicycle drive. Although the algorithm always finds a path, the path may not be optimal. The computational complexity of roadmap-based path-planning algorithms depends on the number of obstacles and the complexity of the obstacle shapes (i.e., the number of the primitives describing all obstacles). The problem of reducing the computational complexity of constructing a tangent graph was addressed in [12]. Among the cell decomposition approaches, grid-based tessellation is the most common and is also used in this work. In grid-based approaches, the complexity mainly depends on the number of cells—a smaller cell size leads to a higher path resolution. The cell size must be small enough to describe the environment with sufficient detail, but must not be too small, as this significantly increases the computational complexity of the search algorithms. The proposed approach introduces an interpolation that can produce smooth paths even at a coarse map resolution.

The earliest graph-theoretic path-planning algorithm is arguably the one developed by Dijkstra [13], which inspired many subsequent variations. It has two aspects associated with its basic construction that could make it less suitable for use in some real-world robot navigation problems. Firstly, it finds the optimal paths between a source node and all destination nodes (or equivalently, between multiple source nodes and a single destination node). In many applications involving only a single robot and a single destination, it increases the computational burden in doing much more than is needed. Secondly, it does not accommodate new discoveries made as the robot's sensor horizon advances on the way to the goal. Yet, in other applications involving several missions originating at different locations, with the need to converge to a single destination such as in warehouses [14,15], or games [16,17], Dijkstra's algorithm matches the need. The computational burden associated with it then becomes justifiable, especially if it is required to be exercised occasionally and the results reused with different starting locations. The focus then shifts to the second issue mentioned above, as to how to make the Dijkstra paths viable even when dealing with environmental changes.

The A* algorithm [18] and its derivatives, such as D*Lite [4], are computationally efficient algorithms compared to Dijkstra in the specific task that they address of finding a path between a single source and goal nodes. This reduced task allows an informed search strategy in the form of a heuristic to be deployed, which leads to the computational savings. The one-source–one-goal paradigm can be identified with a single robot trying to get to a single destination. Both A* and D*Lite accommodate new obstacle discoveries, but the latter is an incremental algorithm which makes it suitable when there are continuing map changes while navigating to the same goal.

Some of the prior efforts of other researchers that use methods related to our path planning and navigation strategy are now discussed. The concept of artificial or virtual potential functions (APF) was first proposed in [8]. They are called artificial because these are not actual electric potentials but are only conceptualised as such. In the original formulation, as explained in Choset [19], the two attractive and repulsive potential functions were algebraically summed to obtain the overall potential function. In practice, the ad hoc parametric choices of the model could set up local minima at which the net force on the robot is zero, resulting in the robot being trapped on its way to the goal.

Let us now turn our attention from the core APF concept to how it was actually realised in prior robot navigation research. Ratering and Gini [20] proposed a hybrid potential field consisting of the combination of a global potential field calculated with a variant of Dijkstra's algorithm and a local potential field synthesised with the help of sonar measurements. Wang et al. [21] also constructed the global and local planners separately. Distance transformation, another variant of Dijkstra's algorithm, is used for the global planner, while an APF-based method is used for the local planner. The overall navigation strategy is characterised by a mediation between the strict need to achieve the subgoals of the global plan and the freedom of the APF-based local planner, so that local minima are avoided. Azmi and Ito [22] propose a technique to handle the local minima problem, in this case, a repetitive oscillatory excursion between two local minima. A map transformation operation was proposed that resulted in the stalemate being resolved through the rotation of the environment space. Lazarowska [23] devised the planning of trajectories for autonomous ships navigating amongst both static and dynamic obstacles. The static APF model accounted for the compliance of special maritime rules that prescribed deliberate actions to avoid collisions between ships. Similarly, Klančar and Seder [15] combined the static APF with local reactive model–predictive planning to avoid collisions among multiple robotic vehicles in warehouse navigation. Amiryan and Jamzad [24] used the APF to complement a pre-determined path generated by a sampling-based path planner such as Rapidly-exploring Random Tree (RRT) [6] to avoid local minima problems. In [25], a hybrid planning method is proposed that combines a particle swarm optimisation algorithm with the APF for static obstacles and the potential field prediction for dynamic obstacles. Several solutions have been proposed to overcome problems with local minima, such as representing concave obstacles by convex representations [26], adding virtual obstacles to move away from local minima [27] or by small perturbations of the APF based on the input-to-state stability property [28]. Alternatively, a robot navigation function can be determined using deep neural networks with reinforced learning as in [29], ant colony optimisation [30], simulated annealing, particle swarm optimisation [25], genetic algorithms and fuzzy logic [31] or the like.

The sampling of the APF-based literature discussed above indicates that the APF concept continues to play a role in robot navigation. Specific use cases range from the original concept of an integrated formulation premised on attractive/repulsive forces to separate formulations addressing the static and dynamic planning phases, to dealing with dynamic moving obstacles and other variations.

The main contributions of this paper are the following.

- A new navigation function is proposed that generates smooth and collision-proof paths by using the bilinear interpolation (BiLI) method to obtain an artificial potential field gradient-descent navigation function from a discrete cost-to-goal (CtG) map obtained

by an optimal discrete grid-based search method. The approach is computationally efficient as it relies on a coarse discrete graph search that can be precomputed for static environments and known goals and can be easily reused for multiple missions from different parts of the environment that need to navigate to a common goal. The bilinear interpolation method implements a continuous potential field and driving direction from a discrete grid-based search.

- Although BiLI is commonly used in computer vision applications, its use for robot planning requires some enhancements, such as handling occupied cells whose values are not defined, interpolating at the environmental boundaries and ensuring continuous gradient descent, which are the main novelties of this work. The resulting path is collision-proof, continuous and close to the optimum, even at the course grid resolution used. It also avoids the problems with local minima that are common in general APF-based methods.
- BiLI can be applied in dynamic environments where incremental graph search methods such as D* or similar [2,4,5,32] can be used to efficiently account for the changes in the environment. In this work, a simple strategy is proposed using Dijkstra for the global CtG map planning and A* for dealing with locally sensed environment changes. When different types of map changes are detected that affect the CtG values, the proposed model uses the A* algorithm to find an emergency bypass path to areas of the environment where the old CtG values are still valid. The bypass path is followed by the determination of a final gradient-descent path segment to the overall goal. Then, a situational decision is made whether to take the additional path segments at the point of the map change or to keep the original path. Using A* to determine the diversion path is relatively fast compared to regenerating the CtG values with Dijkstra, as usually only a few cells need to be examined.

The remaining parts of the article are organised as follows. The first part of the path-planning model developed in this effort is explained in Sections 2 and 3, with the help of a test environment scenario. The reactive strategy to negotiate a blockage of the global path is formulated in Section 4 with the same environment. Section 5 contains a description of additional test scenarios formulated to evaluate this model and the attendant results. Finally, Section 6 contains a brief summarising discussion of the work and suggests further steps, while Section 7 draws some broad conclusions.

2. The Environment and APF Generation

Following a description of the environment, the formulation of the APF values for each cell of the grid-based discretisation is discussed in this section.

2.1. The Environment

A sample environment is shown in Figure 1 which represents the static map in one experiment. Appropriate modifications to it will be subsequently made that reflect the dynamic discovery of new obstacles. Moreover, other obstacle configurations will be created later that represent additional challenges addressed in this work.

The environment consists of a walled-off 10 by 10 m field of play. It can be scaled up to any size, as desired. The entire area is divided into 400 cells (20 by 20), with each cell being 0.5 m square. The environment features four prominent symmetrically positioned "obstacle islands" that, given the starting and goal locations (also shown), block line of sight to the goal for significant portions of a possible path.

The broad characteristics of the environment are:

- Even when an obstacle only overlaps part of a cell, the entire cell is considered occupied. Thus, the map of the environment is in the form of a binary occupancy grid, which also factors in obstacle inflation.
- Dynamic changes in the environment over the initial static knowledge are assumed to be small and localised.

- They can be in the form of additions or subtractions. That is, cells that were unoccupied might be occupied as well as the opposite.

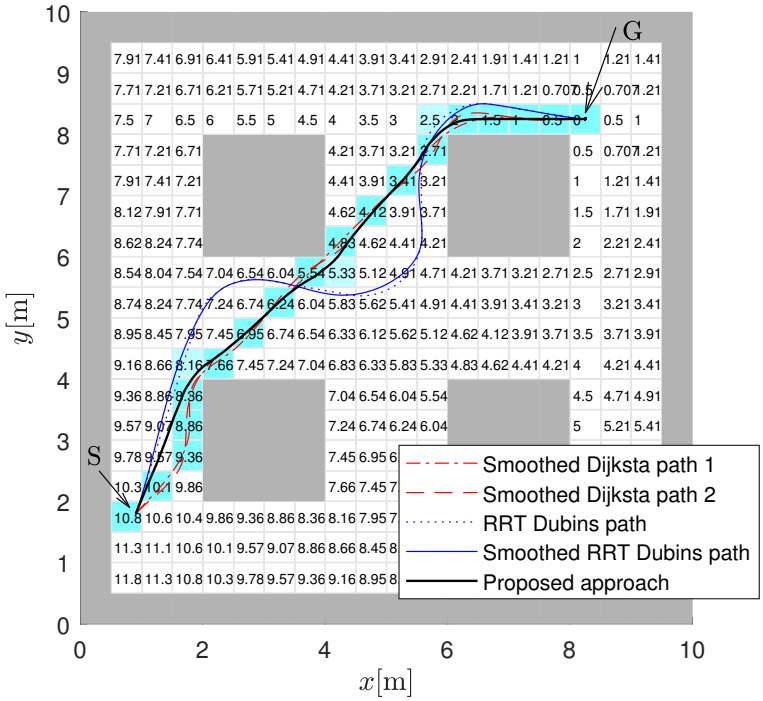

Figure 1. Static map of environment with start (S) and goal (G) locations, discrete CtG values assigned to the free cells (white cells, gray cells belong to obstacles) and some different smooth paths that connect the start and the goal locations.

2.2. Formulation of APF

A variant of the original APF (artificial or virtual potential functions) concept [8] is used to fulfill the global path-planning function. As pointed out earlier, the original APF was conceived as a function that is continuous with respect to space, that addressed both goal seeking and obstacle avoidance in an integrated manner. It followed from attractive and repulsive forces between artificial electrical charges. However, that approach is known to potentially spawn local minima trapping the robot.

As opposed to the classical approach, the floor space is tessellated into a suitable x–y grid to accommodate the use of a standardised discrete occupancy map to represent obstacles. The Dijkstra algorithm is used to generate the cost-to-goal (CtG) value for all cells in the environment, which constitutes the global cost map. In doing this, diagonal cell-to-cell transitions are given appropriate differential weights relative to horizontal and vertical transitions. The CtG values are shown overlaid in Figure 1 within each cell for the assumed environment. For example, the CtG is zero for the goal cell and 10.8 m for the starting cell.

Within the grid map, the Dijkstra algorithm can be used to obtain the shortest path. In Figure 1, the cells that lead from the start to the goal cell are shaded with a light blue colour. A smooth path can then be obtained if a spline (using, e.g., Bézier curves or clothoid curves) is fitted over the centres of the cells that comprise the path. In Figure 1, the Automated Driving Toolbox in Matlab is used to smooth the paths using cubic splines as shown in two examples (Smoothed Dijkstra path 1 and Smoothed Dijkstra path 2). This approach does not ensure that the smooth path does not go too close or even over the obstacles unless collision checking is also made during spline fitting. In Figure 1 is also an example of the path obtained by the approach proposed in this paper. This is a smooth path that is obtained based on the interpolated gradient of the APF that takes obstacles into account implicitly. Figure 1 also presents a smooth path that is obtained with Rapidly-exploring Random Tree (RRT*) [33]. In this case, RRT* uses Dubins' curves [1] to obtain the path from the start to

the goal, and the obtained path is further smoothed by fitting a cubic Bézier spline. The shape of the curve depends on many constraints (e.g., path curvature, segment length, safety distance, vehicle constraints, etc.) that can be given to the algorithm to optimise; therefore, paths with different shapes and smoothness can be obtained. Moreover, the RRT approach is stochastic; therefore, a completely different solution can be obtained in every run of the algorithm even if the input conditions do not vary. Some smoothed paths can be very oscillatory or make large turns around the obstacles, or path smoothing can also produce infeasible paths that collide with obstacles. The proposed approach in this paper is deterministic and produces smooth and near optimal paths that ensure a minimum safe distance from the obstacles, and it is also computationally efficient because it produces satisfactory results even when the cell size is relatively large.

The CtG numbers serve as the classical APF values that are the basis of the global planning within the environment, after some refinement discussed shortly. Just as in the classic APF method, a gradient descent determines the direction of motion. If the static map does not change, a gradient-descent approach based on the CtG values can be used to move the robot all the way to the goal. However, details need to be addressed, such as how the gradients are calculated and smoothed for a function that is discrete over the floor space, as well as how dynamic discoveries in the environment are handled, etc.

3. Interpolation and Smoothing of Potentials and Gradients for Path Planning

The discretisation of the floor—a decision made to contain the computational cost as well as to simplify the consideration of obstacle occupancy—correspondingly creates a discrete CtG surface. This then restricts the resolution of the gradient determination which affects the smooth navigation. To address this, a refinement is introduced through the use of a well-known technique of image resampling known as bilinear interpolation (BiLI) [34,35], which operates on pixels that are like the cells in our environment, as discussed below.

3.1. Bilinear Interpolation

The conceptual basis of BiLI is explained using Figure 2 [35]. BiLI uses a 2 by 2 cell window to interpolate CtG values within the centred unit square region within this window (dashed square in Figure 2), thus creating new data points in an educated manner. It does so by using linear functions to perform the interpolation in what is essentially a planar extension of 1D linear interpolation. The spline-based function representing the interpolating surface is associated with 4 parameters whose values need to be estimated. This is achieved using the CtG cell values at the corners of the unit square.

According to location of a point $[x, y]^T$ in a cell $\mathbf{M}$, a 2 by 2 region of cells around it is chosen for the interpolation, whose centres are connected by a dashed square in Figure 2. Normalised coordinates are found by centres of these cells as:

$$x_n = \frac{x - x_0}{d_c} \quad , \quad y_n = \frac{y - y_0}{d_c} \ , \tag{1}$$

where $[x_0, y_0]$ is the origin of the normalised coordinates defined by the lower left corner of the dashed square and d_c is the cell size. The interpolated and discrete CtG value (potential in the sequel) in normalised coordinates are expressed as $P_n(x_n, y_n) = P(x, y)$ and $U_n(x_n, y_n) = U(x, y)$, respectively. The potential for the four adjacent cell centres (corners of dashed square in Figure 2) are

$$p_{cr} = U_n(x_n, y_n)\Big|_{x_n=c,\, y_n=r}, \tag{2}$$

where $c, r \in \{0, 1\}$ and $U_n(x_n, y_n) = U(x, y)$.

The interpolated potential $U_n(x_n, y_n)$ at any given normalised position $[(x_n, y_n)]^T$ inside the quadrant of cell $\mathbf{M}$ delineated by the unit square is defined as [35]:

$$P_n(x_n, y_n) = w_{00}p_{00} + w_{01}p_{01} + w_{10}p_{10} + w_{11}p_{11}, \tag{3}$$

where the BiLI weights are given by: $w_{00} = (1-x_n)(1-y_n)$, $w_{01} = (1-x_n)y_n$, $w_{10} = x_n(1-y_n)$ and $w_{11} = x_n y_n$.

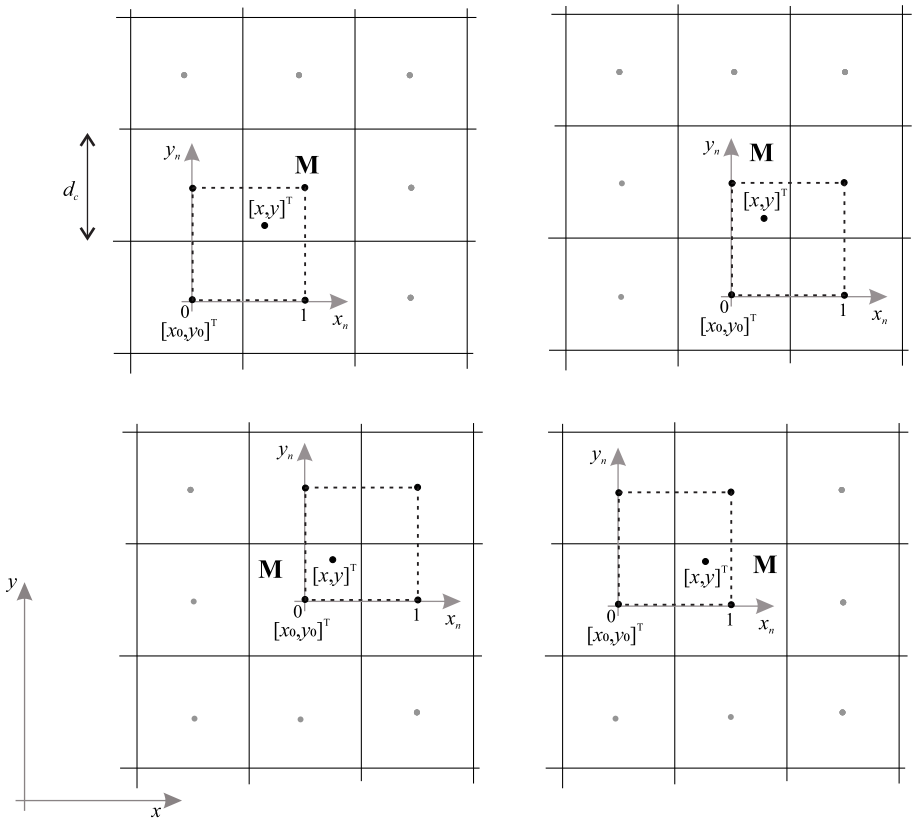

Figure 2. Basis of bilinear interpolation. Interpolated potential at a given point $[x,y]^T$ is defined by the discrete potential at centres (black dots) of four cells connected by the dashed square. Gray dots denote centres of cells.

By following the negative gradient of interpolated potential $P(x,y) = P_n(x_n,y_n)$, the safe path from everywhere in the environment towards the goal location (with potential 0) can be obtained. The negative gradient of $P(x,y)$ in $[x,y]^T$ can be obtained as :

$$\begin{aligned} \mathbf{g}(x,y) &= -\nabla P(x,y) = -\left[\tfrac{\partial P(x,y)}{\partial x}, \tfrac{\partial P(x,y)}{\partial y}\right]^T - \tfrac{1}{d_c}\left[\tfrac{\partial P_n(x_n,y_n)}{\partial x_n}, \tfrac{\partial P_n(x_n,y_n)}{\partial y_n}\right]^T \\ &= -\tfrac{1}{d_c}\begin{bmatrix} p_{11}y_n - p_{01}y_n + p_{00}(y_n-1) - p_{10}(y_n-1) \\ p_{11}x_n - p_{10}x_n + p_{00}(x_n-1) - p_{01}(x_n-1) \end{bmatrix}. \end{aligned} \tag{4}$$

3.2. Adjustments of Bilinear Interpolation for Path Planning

Before applying the interpolation of Equation (3), a check needs to be performed if any of the three neighbour cells of the cell **M** (see Figure 2) involved in the interpolation are occupied. Note that cell **M** is never occupied as we are interpolating potential at a point $[x,y]^T$ inside it. For occupied cells, the potential is typically infinite or undefined, as motion over the obstacles towards the goal should not be possible/permitted. The potential for occupied cell $U(x_m, y_m)$ (with centre at x_m, y_m) is reconstructed from the eight-cell neighbourhood by finding the unoccupied cell with the largest potential, as:

$$\begin{aligned} \{c,r\} &= \underset{c,r}{\operatorname{argmax}}\{U(x_m + d_c c, y_m + d_c c) \neq \infty\} \\ U(x_m,y_m) &= U(x_m + d_c c, y_m + d_c r) + d_c\sqrt{c^2+r^2} \end{aligned}, \tag{5}$$

where $c, r \in \{-1, 0, 1\}$.

Additionally, a check needs to be performed if any of the four cells needed for the interpolation (also interpolation cells, see Equation (3) and Figure 2) is outside the environment. A simple solution to this could be that the grid cell area is always at least one cell larger than the area we are interpolating. More general solution applies calculation of the potential and gradient for a nearby location $[x_t, y_t]^T$, where all four cells used in the interpolation are inside the environment. For a position $[x, y]^T$, where one or more interpolation cells are outside the environment, the nearby location is determined by translating the position from the border for $\frac{d_c}{2}$ in x and/or y direction towards the inside of the environment as follows:

$$x_t = \begin{cases} x & ; \quad x_{min} \leq x \leq x_{max} \\ x_{min} + \frac{dc}{2} & ; \quad x < x_{min} \\ x_{max} - \frac{dc}{2} & ; \quad x > x_{max} \end{cases} \qquad y_t = \begin{cases} y & ; \quad y_{min} \leq y \leq y_{max} \\ y_{min} + \frac{dc}{2} & ; \quad y < y_{min} \\ y_{max} - \frac{dc}{2} & ; \quad y > y_{max} \end{cases}, \tag{6}$$

where environment borders are defined by $x_{min}, x_{max}, y_{min}, y_{max}$. For translated nearby location, interpolated potential is computed from (3) noted as $P(x_t, y_t)$ and the gradient from (4) noted as $\mathbf{g}_t(x_t, y_t)$. Finally, the appropriate potential for each interpolating cell outside the environment (noted as P^*) are reconstructed using Lie derivative (also direction derivative):

$$P^* = P(x_t, y_t) + \mathbf{g}_t(x_t, y_t) \times [x - x_t, y - y_t]^T. \tag{7}$$

Figure 3 (top-left image) shows the resulting potential (CtG values) surface obtained through application of the BiLI technique, corresponding to the environment of Figure 1. Notice how the CtG surface slopes continuously downward from the start point to the goal point, with the four "islands" represented by infinite potentials. This follows from the fact that the CtG values will monotonically decrease from any cell in the environment towards the goal. Moreover, a few sample paths obtained by following the negative gradient (computed from Equation (4)) from different locations towards the goal are shown in the top-right image with blue line. Notice discontinuous change of the negative gradient direction near occupied cells, which can also be observed by checking gradients near obstacle (the cells near the central obstacle in the bottom-left image from Figure 3 with enlarged cell at obstacle corner). To improve this and obtain smoother paths (as illustrated in the top-right figure by purple line), we additionally propose interpolation of the negative gradients in Section 3.3.

3.3. Interpolation of Gradients

The resulting potential in Figure 3 is continuous, but its negative gradient may be discontinuous especially in the vicinity of obstacles, as seen from the bottom-left image in Figure 3 where, at the boundaries between the quadrants of a cell, the gradient shows discontinuities. The negative gradient indicates the required driving direction to reach the goal in optimal manner. Every discontinuity of the gradient is problematic, as a wheeled robot would need to stop and rotate on the spot to reliably follow the course of the negative gradient towards the goal. To improve this, we propose interpolation of the negative gradient, similar to what was performed for the potential field.

For chosen location $[x, y]^T$ in cell $\mathbf{M}$, we estimated the interpolated potential $P(x, y)$ and its negative gradient $\mathbf{g}$ using Equations (3) and (4). Interpolated potential is obtained from the four interpolation cells centre potentials as indicated by Figure 2. In the following, we will use the same interpolation principle to also obtain the interpolated negative gradient $\mathbf{h}(x, y)$ with continuous course as shown in the right images of Figure 3. The gradient for the centres of the four interpolating cells can be estimated from (4), which for each cell considers only the left neighbour ($x_n = 0$) to obtain x element of the gradient $\mathbf{g}$ or only the upper neighbour ($y_n = 0$) to obtain y element of $\mathbf{g}$. Therefore, we estimate the cell centre gradient considering the smallest discrete potential (valid for the cell centre) of both neighbour cells in x or y axis. Denote the interpolating cell centre gradients (similarly

as their potential in (2)) by $\mathbf{h}_{cr} = [hx_{cr}, hy_{cr}]^T$ where $c, r \in \{0, 1\}$. For a cell with centre coordinates $x_m = x_0 + d_c c$, $y_m = y_0 + d_c r$ the cell gradient reads

$$\mathbf{h}_{cr} = -\begin{cases} \frac{1}{d_c}\begin{bmatrix} e(U(x_m + d_c e, y_m) - p_{cr}) \\ f(U(x_m, y_m + d_c e) - p_{cr}) \end{bmatrix} & ; \quad U(x_m, y_m) < \infty, \\ \frac{1}{d_c}\begin{bmatrix} S_x(U(x_m + d_c S_x, y_m) - p^*_{cr}) \\ S_y(U(x_m, y_m + d_c S_y) - p^*_{cr}) \end{bmatrix} & ; \quad U(x_m, y_m) = \infty, \end{cases} \tag{8}$$

where

$$\begin{aligned} e &= \underset{s}{\operatorname{argmin}}\{U(x_m + d_c s, y_m)\} \;\; ; s \in \{-1, 0, 1\} \\ f &= \underset{s}{\operatorname{argmin}}\{U(x_m, y_m + d_c s)\} \;\; ; s \in \{-1, 0, 1\} \end{aligned} \tag{9}$$

and $S_x = \operatorname{sgn}(x - x_m)$, $S_y = \operatorname{sgn}(y - y_m)$ where sgn(.) denotes the sign function. The first part of (8) relates to the case where the cell is free and the second for the case when cell is occupied. If the cell is occupied ($U(x_m, y_m) = \infty$), then its potential is updated (noted as p^*_{cr} in (8)) from already known potential $P(x, y)$ and the gradient $\mathbf{g}(x, y)$ in current position $[x, y]^T$ as follows:

$$p^*_{cr} = P(x, y) + \mathbf{g}(x, y) \times [x_m - x, y_m - y]^T.$$

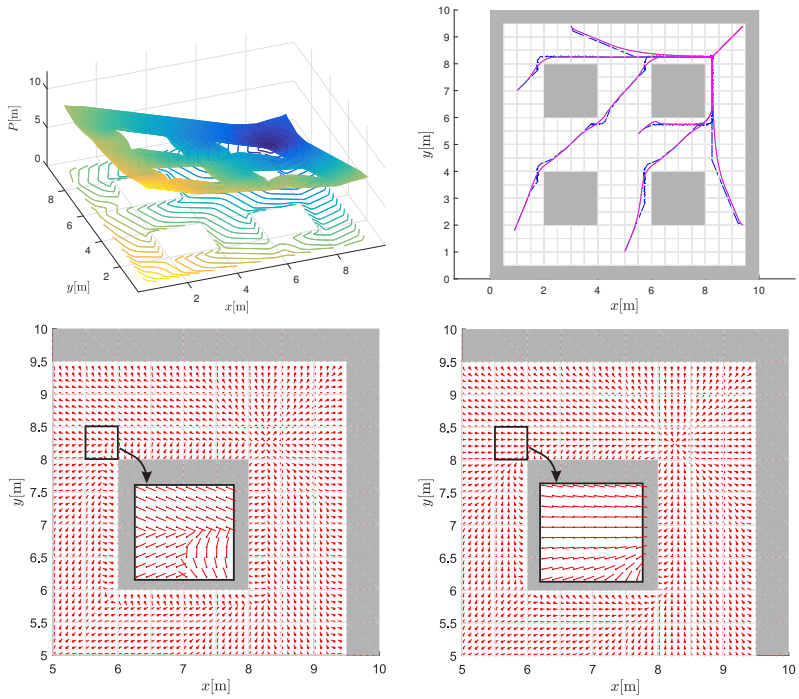

Figure 3. Interpolated potential surface $P_n(x_n, y_n)$ (CtG values, darker colours denote lower CtG values) obtained through bilinear interpolation with contours of equal potential, corresponding to the discrete CtG values of environment of Figure 1 (**top-left**). Obtained paths following the negative gradient (blue line) and interpolated negative gradient (purple line) are shown (**top-right**). Part of the environment near the goal with negative gradients (red lines going from black dots outwards) computed from (4) (**bottom-left**) and interpolated negative gradients (**bottom-right**).

Before computing the cell gradients in (8) and (9), a check needs to be made if any of the neighbour cells is outside the environment. If this is the case, the potential of this cell is reconstructed similarly as in (7) considering the known interpolated potential $P(x, y)$ and gradient $\mathbf{g}(x, y)$ for the point $[x, y]^T$.

From estimated cell gradients $\mathbf{h}_{00}$, $\mathbf{h}_{01}$, $\mathbf{h}_{10}$ and $\mathbf{h}_{11}$ (Equation (8)), the final interpolated gradient in current position is obtained by:

$$\mathbf{h}(x, y) = w_{00}\mathbf{h}_{00} + w_{01}\mathbf{h}_{01} + w_{10}\mathbf{h}_{10} + w_{11}\mathbf{h}_{11}, \tag{10}$$

where the same weights w_{00}, w_{01}, w_{10}, w_{11} as in (3) are used. The comparison of the gradient field **g** and the improved interpolated gradient **h** is shown in Figure 3 in the lower graphs. The obtained planned paths by following the interpolated gradient of the potential field result in collision safe and smooth paths as shown in the top-right graph of Figure 3.

Bilinear interpolation can therefore be elegantly used to obtain continuous potential field as well as appropriate desired driving directions (the interpolated negative gradients **h**) based on a discrete grid-based cost map (discrete CtG potential field). The obtained paths in path planning are collision safe, without local minima and with continuous course of driving direction, which is important for robotic vehicles. Note that Bicubic interpolation [15,36] could also produce smoother interpolations, but it requires 4 by 4 neighbourhood, which is problematic in the vicinity of obstacles or in narrow corridors, as the occupied cells have infinite CtG value. Occupied cells require special treatment before they are used in the interpolation. This becomes even more challenging for bigger neighbourhoods (e.g., 4 by 4 as opposed to 2 by 2). Therefore, for the path planning, we propose the use of bilinear interpolation with appropriate preprocessing of occupied cells and with additional gradient interpolation to obtain smooth paths. The path is smooth, even as it is intuitive and optimal. As mentioned earlier, no local minima will exist in the CtG contour, unlike in the classical APF formulation, because of the inherent manner of its construction. However, if the static map is augmented by new obstacles put in play after its creation, this could change. This eventuality is dealt with in the next section.

Let us analyse the computational complexity of the proposed smooth path planning. The path is obtained with a gradient-descent method. The number of steps that are required to reach the goal is dependent on the size of the sampling step. The sampling step can be lower bounded to prevent oversampling and upper bounded to obtain desired smoothness of the path. Consider that a particular path passes over M cells and that the sampling step is selected in a way that on average (or at maximum) L iterations of the gradient descent are made in each of the cells. This means that computational complexity of gradient descent is $\mathcal{O}(LM)$, and it is therefore dependent on the sampling step and path length. In each step of the gradient-descent calculation, an interpolated value of the gradient is obtained from the four nearest surrounding cells (see Equation (10)) around the current path point. The gradient in each of these four cells is calculated from the APF of the four neighbouring cells (Equation (8)). Therefore, to compute the interpolated gradient for a given cell, a neighbourhood of twelve cells is needed in total. Hence, the gradient does not need to be determined for every cell, but only for the cells that are along the path. We assume that the gradient calculations can be cached; therefore, the computational time and space complexity of obtaining the gradients in the cells around the path are $\mathcal{O}(M)$. Note that cell gradient calculations could also be made in parallel if calculation speed is crucial. We calculate the values of the APF for the entire map with N cells using Dijkstra algorithm, which in case of a grid map has a computational time complexity of $\mathcal{O}(N \log(N))$ and final space complexity of $\mathcal{O}(N)$. The values of the potential field could also be determined only for the cells in the vicinity of the path that are sufficient for gradient calculation. This is a viable option when we would like to quickly determine only a single path, especially if the map is very large. In this case, the Dijkstra algorithm can be stopped once the goal is reached (in this step, it is also beneficial to use A*) and then the Dijkstra algorithm is resumed only if the value of the potential for an unknown cell needs to be known and only until the final value of the cell potential is known. However, in our case, we calculate the APF for the entire map, as this enables fast recalculation of various paths that lead to a single goal (or begin at a common start), which is beneficial for the cases when part of the map changes, as presented in the next section.

4. Discovery of New Obstacles and Reactive Avoidance Manoeuvre

A change in the environment is shown in Figure 4 with the addition of an L-shaped obstacle (in Figure 4 (top-right)) cluster roughly halfway through the initially planned path in Figure 4 (top-left). The detection of these additional obstacles is assumed to take

place when the robot's sensor horizon includes the region, through an iterative comparison between what it sees with its sensor and what it expects to see from the initial static map. The range of the sensor used will have some effect on when any reactive manoeuvre is initiated, but this detail is not critically relevant to our model development.

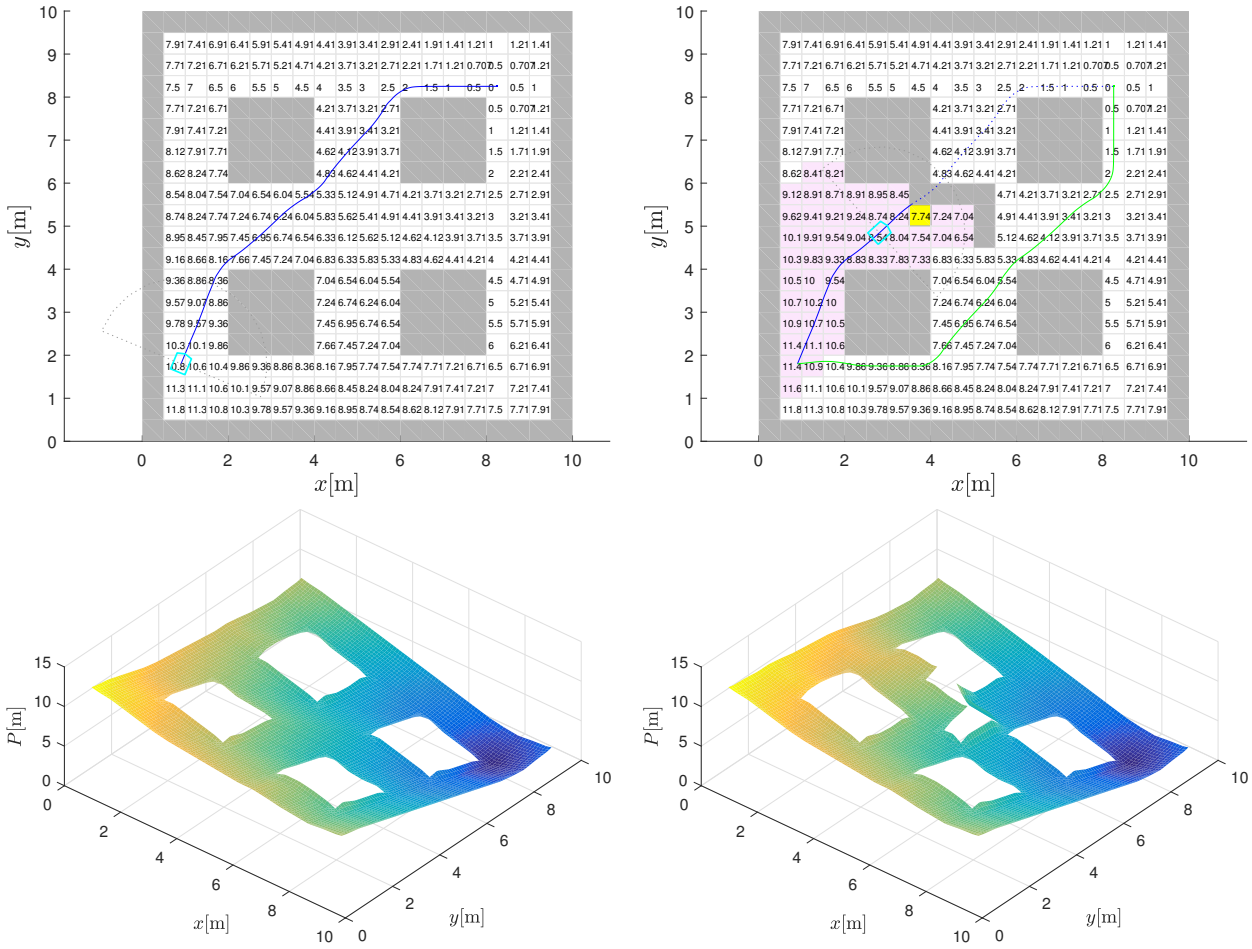

Figure 4. Augmented environment with new discoveries. Static map of environment with planned path in blue line towards goal (**top-left**). Change of environment with new L-shaped obstacle where robot (shown by the sky blue line and its sensor range by the gray dots) is blocked at stuck cell (highlighted by yellow) if continuing based on static CtG map. Replanned CtG values for new environment with highlighted light purple cells where CtG values have changed and new gradient-descent path in green (**top-right**). Interpolated potential (CtG, darker color denote lower values) surface before (**bottom-left**) and after the environment change (**bottom-right**).

It should be noted that if the new obstacle does not impede motion, then nothing needs to change. However, following the initially planned global path here will stop the robot at the "stuck cell" (see the global path in Figure 4 (top-left) in conjunction with the "stuck cell" shown in yellow in Figure 4 (top-right)), which is effectively a local minimum forced by the new obstacle. Essentially, the robot ends up in a trap in pursuing gradient descent based on the initial plan. In fact, the discovery—here, an addition to the static obstacle map—can be taken as an indication that the CtG values in the vicinity are no longer reliable.

The results of repeating the Dijkstra algorithm to refresh the CtG values of the entire environment with the new obstacles included are also shown in Figure 4 (top-right). The cells with modified CtG values are highlighted (in comparison to plot (top-left)), which clearly reveals that the new obstacles only influence the CtG values (increases them) of a small subset of the cells that are upstream of the new obstacles. The downstream CtG

values are unaltered, as would be expected. The new interpolated CtG surface is shown in Figure 4 (bottom-right) and is in accordance with the updated map.

The new gradient-descent path from the original starting point is also shown in Figure 4 (top-right) with the green line, which confirms that if we had been aware of these new obstacles at the very beginning, the global path planning would have accounted for it and the CtG numbers would have been monotonic again. This recalculation could have also been performed from the trapped position of the robot to the goal. However, this is the calculation that is to be avoided because of the associated computational burden. The new path is just presented here to make a point.

Thus, the challenge is to come up with a reactive strategy that enables the robot to get around the obstruction through developing a bypass path that involves minimal computation effort. This path should lead the robot to an area where the old CtG values can be used again to continue travel. The flowchart shown in Figure 5 is a broad representation of the core method adopted. There are minor case-based variations stemming from the type of map change encountered, which are not included in this basic flowchart for simplicity. The explanation that follows is with reference to this flowchart as well as the two in Figures 6 and 7 that follow, dealing with lower-level steps in the algorithm.

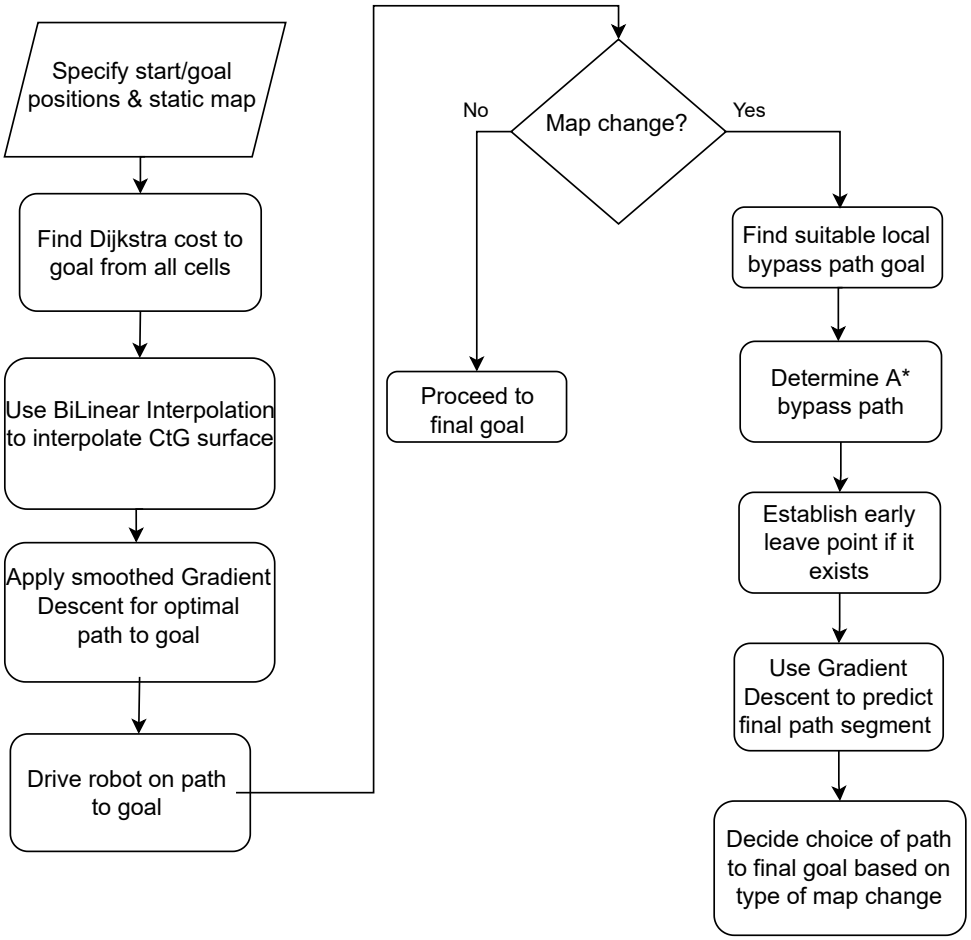

Figure 5. Core flowchart of algorithm (some variations based on case). Leave point determination applies only when current path is blocked.

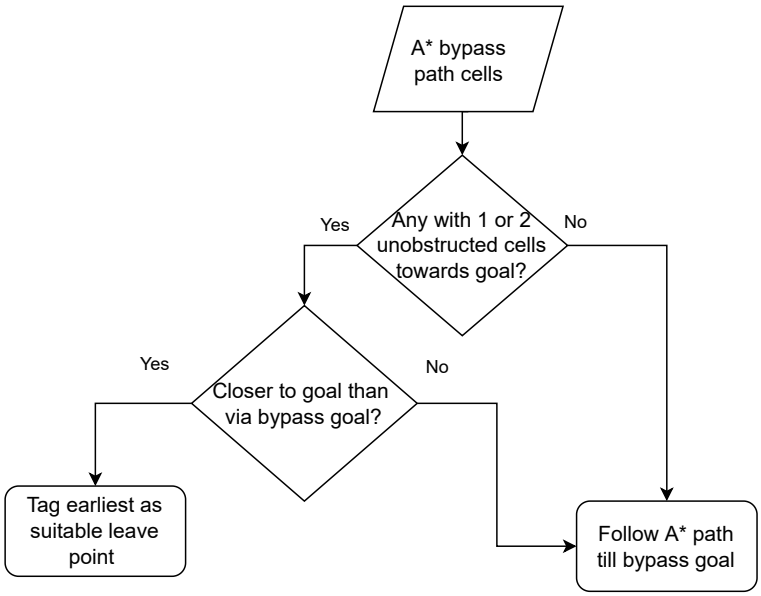

Figure 6. Finding suitable leave point. Leave point is established if cell on A* path has immediate unobstructed cells in original goal direction and is closer to it than path via bypass goal.

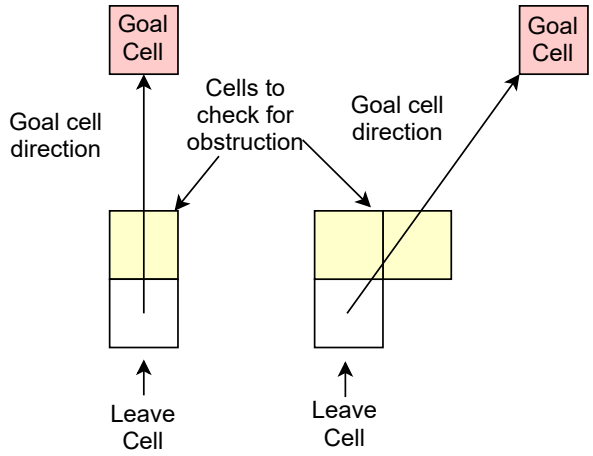

Figure 7. Checking blockage condition for leave cell candidates. Based on alignment of potential leave cell with original goal cell, occupancy status of either one or two cells is checked to determine whether obstruction is present.

A predefined, 5 by 5 search window of cells (see the yellow dashed square in Figure 8 (top-right)), which can be thought of as a "fishing net", is centred at the stuck robot cell (the yellow cell in Figure 8 (top-right)). The cells within the window are examined to find the lowest CtG value that is smaller than the CtG value at the stuck cell, as per the original static map, to use as a temporary intermediate goal (the green cell in Figure 8 (top-right) to help negotiate the blockage caused by the newly discovered obstacles with a sensor (e.g., LiDAR). An inherent assumption is that the size of the search window is sufficient to uncover a cell with a CtG value that is on the other side of the blockage, and hence unaffected by it. This is facilitated by centring the net at the stuck robot cell right next to the new obstacles blocking the path, even though the blockage may have been detected even further away by the robot's LiDAR. If this step does not uncover a satisfactory temporary local goal, larger nets are cast iteratively until a suitable cell is found. Moreover, it might be possible to get more creative with the footprint adopted for this search window, which is beyond the scope of this work.

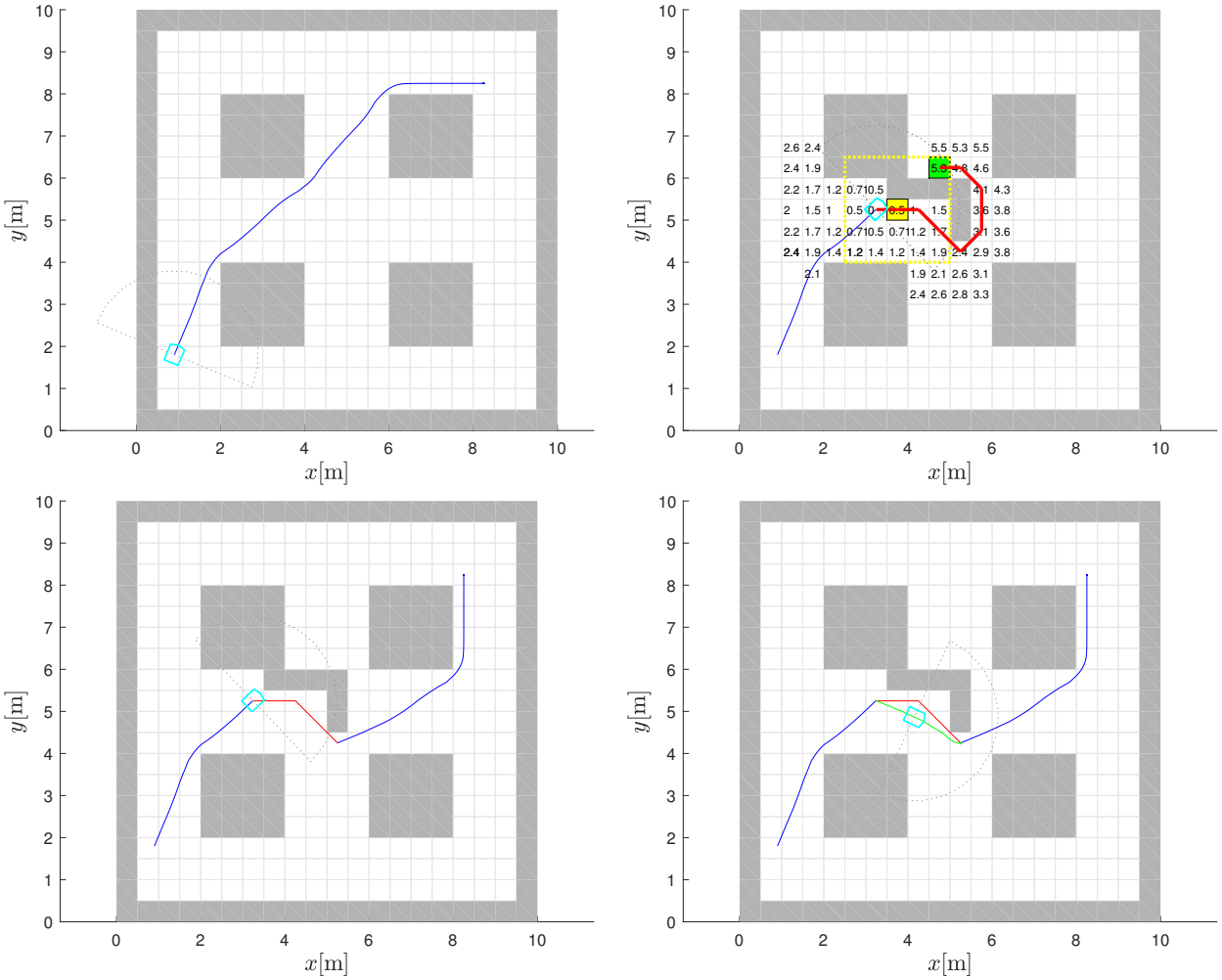

Figure 8. Strategy to negotiate dynamic obstacles. Original planned path based on the static environment map shown is shown with blue line (**top-left**). New obstacle blocking path initiates A* bypass (red line) calculation from stuck cell (highlighted in yellow) to the temporary intermediate goal cell (highlighted in green) (**top-right**). The leave point and final overall path is shown with A* bypass (**bottom-left**) and in by smoothed bypass (green line) using BiLI interpolation (**bottom-right**).

With the stuck cell and the temporary goal identified, the A* algorithm is then run to find a diversionary path from the robot's stuck location to the temporary local goal identified, as described above. This step should not result in an "unreachable goal" being returned, as the algorithm is executed with the known map at the time. The resulting path obtained is also shown in Figure 8 (bottom-left). It should be noted that using D*Lite instead of A* will not result in any computational savings, because the algorithm needs to be run just once, in which case there is no advantage of one over the other. Whether it is necessary to follow the A* bypass path all the way to completion would depend on the situation. The reason is that if in negotiating the obstacle via the A* path the robot finds itself in a cell that is closer to the final goal, completing the A* path and then proceeding to it is not as efficient compared to treating the cell as a leave point. This additional condition built into the algorithm is laid out in the flowchart extension presented in Figure 6.

A good leave cell must also satisfy another condition. It should also be one for which there is at least one unobstructed cell within its 8-cell neighbourhood in the direction of the goal. This is illustrated in Figure 7 for the two situations that represent all the possibilities that can occur. Essentially there are two cases, because of the quantisation imparted by the discretised cell structure. Either one or two cells need to be checked, depending on the relative positioning of the goal cell and the candidate leave cell. That is, are they vertically

or horizontally aligned or at a different angle, in which case the line joining them will be straddled by two cells. In the latter case, only one needs to be free to meet the leave condition. If none of the leave cells pass the dual tests, the bypass path is followed all the way to the temporary bypass goal.

When a leave cell is established, it is taken to mean that the local obstruction has been satisfactorily bypassed and the old CtG values are valid again. The algorithm reverts to the gradient descent using the old CtG values. It should be noted that it is possible to determine a suitable leave point and the final gradient-descent segment even before the robot moves from the stuck cell. This enables the path to be evaluated before travel. The leave point and the final overall path of the robot between the original starting point and goal location are also shown in Figure 8 with A* bypass (bottom-left) and by the smoothed bypass path using BiLI interpolation (bottom-right).

The use of the CtG values and gradient descent as an overarching method to drive to the goal and the separate handling of unexpected obstructions through a bypass path ensures that, unlike in the classical formulation of the APF, local minima cannot be formed at the global path planning with static map phase. That is, the two-step approach results in local minima being caused only by newly discovered obstacles and transfers the burden of resolving them to the local path-planning phase. This is a key element of the path-planning strategy used here.

The planning model discussed here was evaluated in additional experiments. The environments used and the attendant results obtained are discussed in the next section.

5. Additional Experiments and Results

5.1. Obstacle Missing from Static Map within Sensor View

A new environment is shown in Figure 9 (top-left) with the start (lower left) and goal (upper right) locations, as well as the path resulting from the global path-planning phase. As the robot travels towards the goal and its sensor horizon advances, it discovers a change in the static map on which the planning was premised. A cell in the "wall" that was thought to be blocked is found to be clear from the robot's sensor view. Here, the change is a subtraction as compared to the earlier example. The cleared cell as well as the cell where the discovery was made are shown in Figure 9 (bottom-left).

This can be used to trigger an opportunistic strategy—when a blockage is removed, there is a possibility that a shorter path to the goal exists and needs to be investigated. The A* algorithm is invoked to determine a path to the region of the cell, which is now open, so that the old CtG surface can be used again. This is accomplished by setting the target cell for the A* bypass path to an unblocked cell beyond the cleared cell in the direction of the goal. The potential A* bypass path is shown in Figure 9 (bottom-left) in red and the corresponding smoothed version by the green line in Figure 9 (bottom-right). The remaining path from that cell to the original goal cell is again obtained through using the CtG values and gradient descent and is shown in Figure 9 (bottom-right) in which the overall path from the original starting position to the goal is also evident.

If there are multiple occupied cells that are now clear, all of them need to be assessed in the same manner as discussed above. Before taking a bypass path, a check needs to be conducted to see whether the modified path (made up of an A* segment followed by a gradient-descent segment) is shorter than the remaining current path to the goal. Only if the newer path is better should the robot use it to get to the goal instead of the original path. In the environment considered, this happens to be the case.

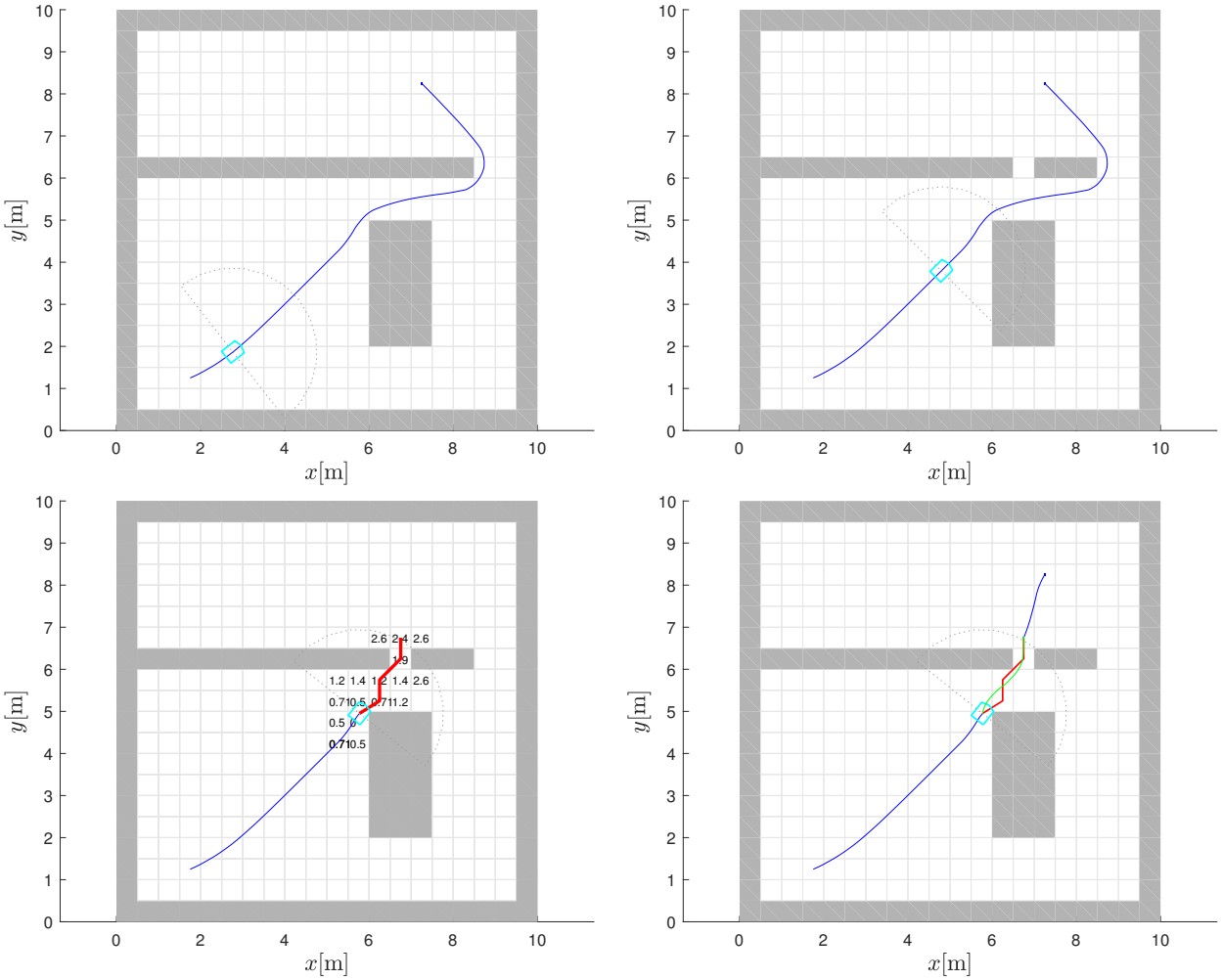

Figure 9. Path planning with obstacle clearance within sensor view. Initial path planned (blue line) based on knowledge of global static map is followed (**top-left** and **top-right**). Robot detects change in static map using sensor view (cleared cell) and computes A* bypass path (red line) to an unblocked cell beyond cleared cell (**bottom-left**). The smoothed bypass path (green line) connects to the gradient-descent path (blue line above the obstacle) based on CtG (**bottom-right**).

5.2. Obstacle Missing from Static Map Outside of Sensor View

It is also possible to conceptualise a situation where the robot comes to know about the removal of an obstacle from the static map in a region of the environment outside the sensor range of the robot while it is in motion on the current path to the goal. For example, this could happen when another robot passing by that region notices and relays the change(s) to a central station and/or all agents. While this could be a corrected error in the map creation, it could also be the result of a temporary obstacle (for example, a fallen tree) being cleared.

This situation can be handled in the same way as the previous one. A potential A* bypass path can be estimated from the robot's current position to a target cell just beyond the cell whose occupancy status has changed. This serves as a bridge to an area where the old CtG values are still valid. As before, the next segment of the path is established using the gradient descent from the temporary bypass goal on to the original destination. Then, based on whether this new alternate route is shorter than the remaining part of the current route, the robot can decide whether to switch to the alternate path or continue on the current one.

An example of this case is shown in the environment in Figure 10. The various parts of the figure are in line with Figure 9 and can be understood with the help of the caption.

In the environment of Figure 10, the information on the map change helps chart a shorter route to the goal.

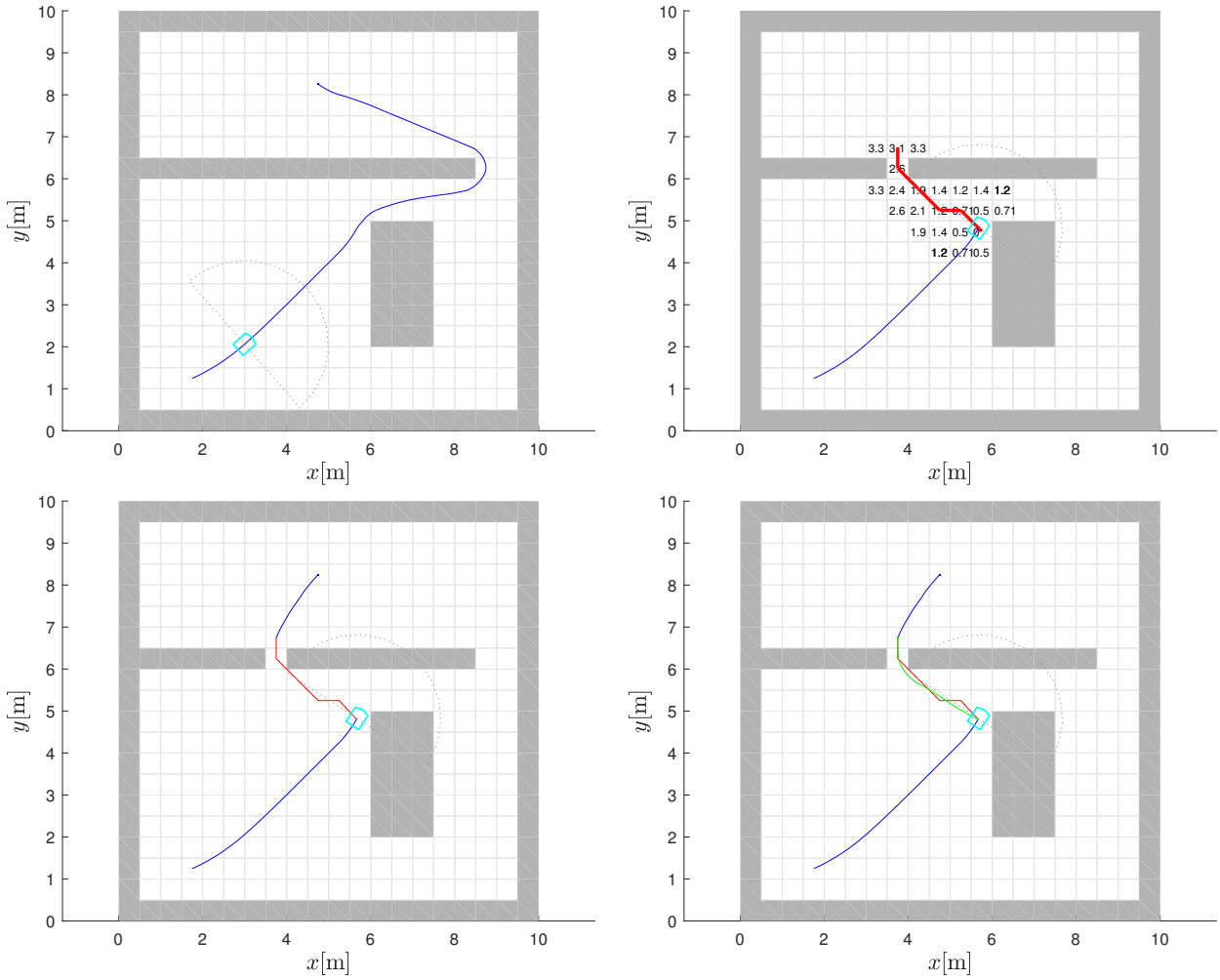

Figure 10. Path planning with obstacle cleared outside the robot sensor view. Initial path (blue line) planned based on knowledge of global static map is followed (**top-left**). Robot is informed of cleared obstacle cells (outside robot's sensor view) and A* bypass path (red line) to an unblocked cell beyond cleared cell is computed (**top-right**). The composite alternative path made up of final gradient-descent path (blue line above the obstacle) using the original CtG (**bottom-left**) and smoothed bypass version (green line) (**bottom-right**) is shorter than the one in the (**top-left**) figure and robot switches to it.

5.3. U-Shaped Trap

The last environment considered is one which incorporates the classic U-shaped concave trap. The static map is initially empty and the planned path between the start position and the goal is shown in Figure 11 (top-left), before the blockage is discovered, and is as expected. Even when the lower part of the U-shaped particle is discovered (Figure 11 (top-right)), the robot continues to proceed on the initial straight path recommended by global path planning. It does this until it encounters the core structure of the trap and the attendant local minima created (Figure 11 (bottom-left)). The ability to resolve an unexpected concave obstacle configuration on the planned path is a good test of the ability of an algorithm, because those that are purely combinatorial will fail this test.

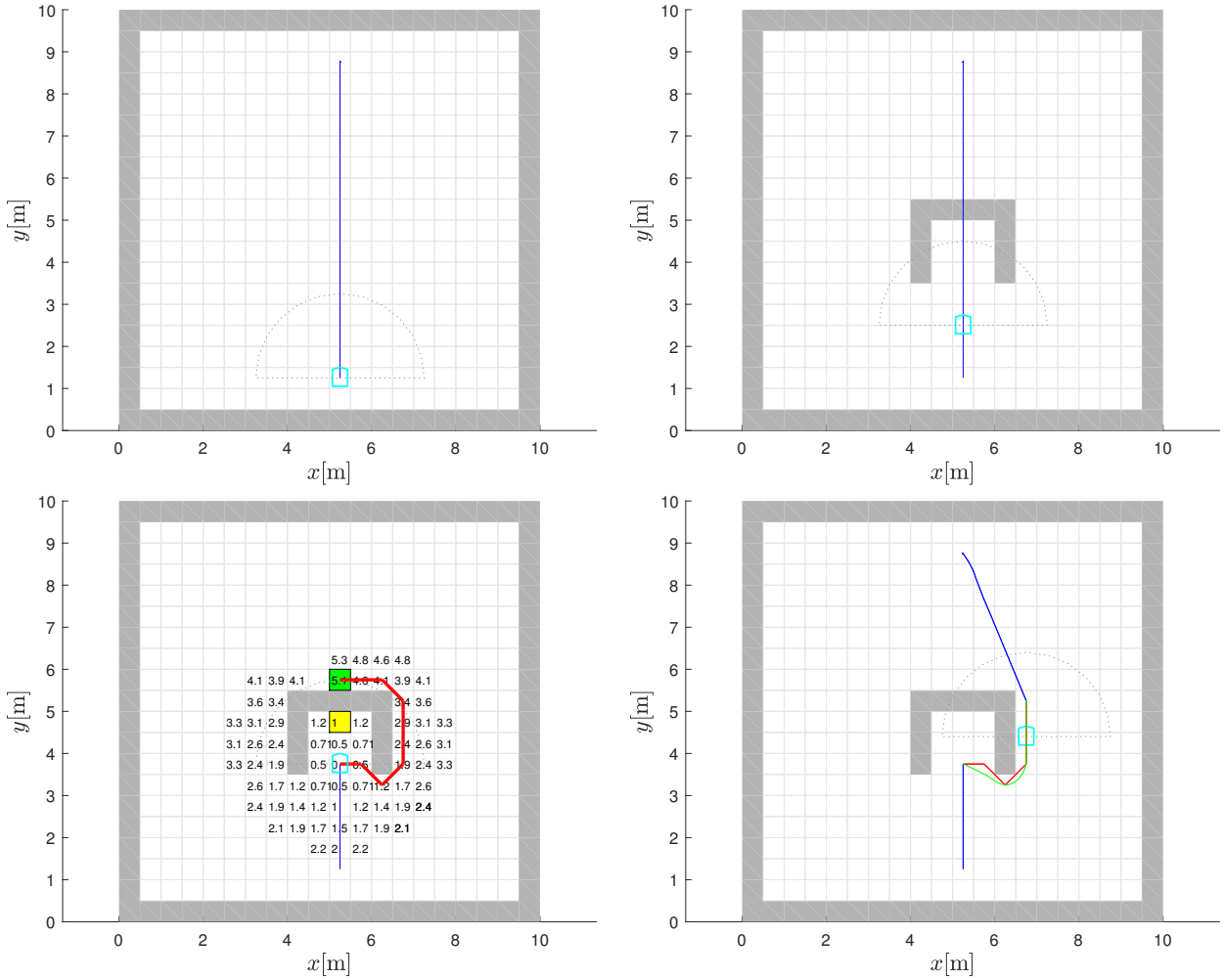

Figure 11. Path planning with appearance of unexpected U-shaped trap. Robot initially plans path (blue line) in empty static map (**top-left**) and starts to travel on it (**top-right**). When blockage is within sensor range, a bypass path is computed by a red line (**bottom-left**). The path is smoothed (green line) and connects with the final gradient-descent path (blue line) using CtG (**bottom-right**).

In accordance with the algorithm, a 5 by 5 window of cells is examined around the stuck cell (the yellow cell in Figure 11 (bottom-left) where the robot's initial gradient-descent path is blocked by the obstacle) to find the lowest CtG value below the current one (temporary intermediate goal cell marked by green in Figure 11 (bottom-left)). As explained earlier, such a cell is considered as being in an area where the CtG values are unaffected by the new blockage. If in some other case, the chosen 5 by 5 search window size does not produce a suitable target, because the cell with a minimum CtG value smaller than the current value lies within the trap zone, the search can be repeated using a larger 7 by 7 window and so on. This will eventually yield a temporary bypass goal outside the trap. The A* bypass path to this cell is also shown in Figure 11 (bottom-left) in red. The smoothed version obtained by the bilinear gradient interpolation in green, the post-bypass path segment based on the gradient descent, and the overall composite path between the start and goal points is shown in Figure 11 (bottom-right). Note that the path follows the computed bypass only until the leave point where the old CtG values and its interpolated gradient-descent path can be followed again.

6. Discussion

Although bilinear interpolation and the calculation of gradients from a discrete grid are well-established in image processing, their direct application to a discrete APF can lead to

several problems. At the points where the cell is connected to its neighbour, discontinuities can occur in the gradients, making the use of a gradient descent problematic. This can lead to undesirable zigzag paths when following the direction of the gradient descent. This problem is much more pronounced near obstacles. The potential of occupied cells is not defined by the CtG assignment and can be considered infinite, as the cell should not be part of a path to the goal. This can lead to problems with local minima near obstacles where the direction of the gradient descent could change by more than 90°.

To interpolate the potential appropriately, one could also use some other higher-order interpolation technique, such as the bicubic interpolation [15,36]. The advantage of this technique would be a smooth gradient transition at the cell border because it uses third-order polynomials for the interpolation, which are continuous up to the second derivative (C^2). However, bicubic interpolation requires the use of a 4 by 4 neighbourhood, which becomes problematic near obstacles or in narrow corridors because the occupied cells have infinite (undefined) potential values. These occupied cells need special treatment before they can be used in the interpolation. Therefore, bicubic interpolation brings its own problems, as it requires a neighbourhood of 16 cells for the interpolation, in contrast to bilinear interpolation, which only requires 4 cells.

An additional problem that plagues bicubic interpolation is the occurrence of anomalies such as surface oscillations and the possibility of local minima near obstacles. Third-order polynomials have a continuous gradient which, while fitting the equidistant cell centres near obstacles (e.g., obstacle corners), causes oscillations in between (a common problem in the interpolation where the fit is perfect at the data point but could be oscillatory in between, known as the Runge phenomenon).

In image processing, the gradient is normally computed with the convolution of the image with the gradient operator. There are various gradient operators, such as one-dimensional operators (e.g., $[1, \ -1]$, $[1, \ 0 \ -1]$) and Robert's cross or the Prewitt, Sobel and Scharr operators [37], which are more robust to noise. Some of these filters introduce gradient shifting and/or smoothing. In our case, averaging is not required, as the APF inherently does not contain noise. The proposed method for calculating the APF gradient is therefore different from the gradient methods used in image processing because it is designed in a way that the gradient in the cell centres always points towards the neighbouring cell with the lowest potential, or the gradient is zero if the current cell has the lowest potential. This ensures that the interpolated gradient points towards the cells with the lowest potential, regardless of the potential magnitude in the cell neighbourhood (without an undesired gradient shift and averaging). Therefore, an optimum smooth path from the start to the goal can be obtained, because the obtained path accurately follows the bottom of the valley that is defined by the APF.

The calculation of the potential field and the gradient in the free space away from the obstacles is straightforward, but in the vicinity of the obstacles, special care is needed to determine the appropriate potential and gradient, as presented in the paper. In the cells surrounding the obstacles, the gradient is calculated from the potential field of the neighbouring cells. If some of these adjacent cells are occupied by obstacles, the potential in these cells may be different depending on which side of the obstacle the gradient is calculated—this occurs in the case of thin or diagonally touching obstacles. Therefore, the batch calculation of the APF can only be performed for the cells that do not touch the obstacles. Moreover, to determine the optimal path, the proposed approach can calculate the gradients online only for the cells that are surrounding the tip of the path while it is being generated. One could resample the grid to double/quadruple the resolution of the map to alleviate the problems encountered near thin obstacles. However, this would require more computational resources (by a factor of four in the case of a double resolution) to calculate the APF and its gradient. However, because the gradient is interpolated, smooth paths are obtained even if the resolution of the map is low.

The proposed bilinear interpolation is applied to the path planning in a static and dynamic environment. A simple model for combining the global and local path planning

that also derives from the original potential fields concept is used. Its key aspects are as follows. A method is needed for multiple missions that could potentially require navigation to the same destination in the environment. A static APF is therefore interpolated based on the pre-calculated CtG values for the cells navigating the path to the goal. A global plan is created based on these CtG values using a gradient descent on the static APF. Along its path, local map changes in the environment can be detected in various ways. A bypass strategy is formulated that enables the robot to find and evaluate a temporary bypass path through the use of the A* algorithm. A case-based decision is made whether to take the alternate path. Obtaining the diversionary path is relatively fast compared to regenerating CtG values for the entire environment. The benefits will be proportionately even greater for larger and less constricted environments.

In summary, the proposed approach introduces the following contributions/modifications in APF-based path planning: a small required neighbourhood (2×2, compared to other interpolations), an easier treatment of the occupied cells in the interpolation, and no anomalies that could result in local minima or the oscillating direction of the gradient-descent path near obstacles. The basic bilinear interpolation has a discontinuous gradient between cells, which we take into account by our proposed additional interpolation for gradients. Standard image processing techniques usually apply a batch calculation to the entire image, which simplifies the algorithmic flow in an obstacle-free area. However, additional special care is required to determine the appropriate potential and gradient near the obstacles. The proposed approach reduces the computational effort as the interpolation is only performed on cells along the path and not for the entire environment as is usually the case with batch image processing algorithms.

Due to the applied interpolation in the potential function, good quality paths are obtained even at a rough grid resolution of the environment. These contribute to the computational and memory requirement efficiency of the approach.

7. Conclusions

This work proposes a new approach to construct a navigation function in a variant of artificial potential fields (APF) that can be applied to navigation, path planning and the control of a mobile robot. The navigation function guarantees safe guidance to a goal without local minima in concave traps, which is a common problem with APFs. Optimality and convergence are inherited from an optimal grid-based search which results in a discrete APF.

To obtain a smooth navigation function and the associated gradient-descent driving direction, we apply a bilinear interpolation with several novel extensions that allow an efficient application to path planning. First, we propose a reconstruction of the discrete potential for the neighbourhood of cells used in the interpolation that belong to obstacles or are outside the environment. Second, we introduce an additional interpolation of the gradient-descent directions from the estimated discrete gradients of the interpolated cells. This leads to a smooth, optimal and collision-safe path or navigation towards the goal. Third, the proposed interpolation approach can be performed online and is computationally efficient as it only interpolates the discrete cell potentials along the planned path.

Several path planning results are provided to illustrate the performance of the navigation function. To illustrate the application in dynamic environments, we propose a simple strategy to combine global and local path planning to bypass detected dynamic changes. The strategy enables a robot to find and evaluate a temporary bypass path around newly detected obstacles through the use of the A* algorithm.

Author Contributions: Conceptualisation, methodology and software, M.K., A.Z. and G.K.; writing—original draft preparation, M.K.; writing—review and editing, M.K., A.Z. and G.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly founded by a US Fulbright Scholar grant to Mohan Krishnan and partly by the Slovenian Research Agency under Grants P2-0219 and L2-3168.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006; p. 1007.
2. Stentz, A. Optimal and efficient path planning for partially-known environments. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; Volume 4; pp. 3310–3317. [CrossRef]
3. Stentz, A. The Focussed D* Algorithm for Real-Time Replanning. In Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 20–25 August 1995.
4. Koenig, S.; Likhachev, M. Fast replanning for navigation in unknown terrain. *IEEE Trans. Robot.* **2005**, *21*, 354–363. [CrossRef]
5. Philippsen, R.; Siegwart, R. An Interpolated Dynamic Navigation Function. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 3782–3789. [CrossRef]
6. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
7. Sánchez-Ibáñez, J.R.; Pérez-del Pulgar, C.J.; García-Cerezo, A. Path Planning for Autonomous Mobile Robots: A Review. *Sensors* **2021**, *21*, 7898. [CrossRef] [PubMed]
8. Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
9. Bhattacharya, P.; Gavrilova, M.L. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. *IEEE Robot. Autom. Mag.* **2008**, *15*, 58–66. [CrossRef]
10. Liu, Y.H.; Arimoto, S. Path Planning Using a Tangent Graph for Mobile Robots Among Polygonal and Curved Obstacles: Communication. *Int. J. Robot. Res.* **1992**, *11*, 376–382. [CrossRef]
11. Savkin, A.V.; Hoy, M. Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments. *Robotica* **2013**, *31*, 323–330. [CrossRef]
12. Eapen, N.A. Path planning of a mobile robot among curved obstacles through tangent drawing and trapezoidal decomposition. *Eng. Sci. Technol. Int. J.* **2021**, *24*, 1415–1427.[CrossRef]
13. Dijkstra, E.W. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
14. Fragapane, G.; de Koster, R.; Sgarbossa, F.; Strandhagen, J.O. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *Eur. J. Oper. Res.* **2021**, *294*, 405–426. [CrossRef]
15. Klančar, G.; Seder, M. Coordinated Multi-Robotic Vehicles Navigation and Control in Shop Floor Automation. *Sensors* **2022**, *22*, 1455. [CrossRef] [PubMed]
16. Mali, A.D.; Motion Planning in Computer Games. In *Encyclopedia of Computer Graphics and Games*; Lee, N., Ed.; Springer International Publishing: Cham, Switzerland, 2018; pp. 1–6. [CrossRef]
17. Banik, S.; Bopardikar, S.D. Attack-Resilient Path Planning Using Dynamic Games With Stopping States. *IEEE Trans. Robot.* **2022**, *38*, 25–41. [CrossRef]
18. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
19. Choset, H.; Lynch, K.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.; Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementations*; MIT Press: Cambridge, MA, USA, 2005; p. 603.
20. Ratering, S.; Gini, M. Robot navigation in a known environment with unknown moving obstacles. *Auton. Robot.* **1995**, *1*, 149–165. [CrossRef]
21. Wang, L.C.; Yong, L.S.; Ang, M. Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment. In Proceedings of the IEEE Internatinal Symposium on Intelligent Control, Vancouver, BC, Canada, 30 October 2002; pp. 821–826. [CrossRef]
22. Azmi, M.Z.; Ito, T. Artificial Potential Field with Discrete Map Transformation for Feasible Indoor Path Planning. *Appl. Sci.* **2020**, *10*, 8987. [CrossRef]
23. Lazarowska, A. Comparison of Discrete Artificial Potential Field Algorithm and Wave-Front Algorithm for Autonomous Ship Trajectory Planning. *IEEE Access* **2020**, *8*, 221013–221026. [CrossRef]
24. Amiryan, J.; Jamzad, M. Adaptive Motion Planning with Artificial Potential Fields Using a Prior Path. *arXiv* **2020**, arXiv:2005.04191.
25. Mandava, R.; Bondada, S.; Vundavilli, P. An Optimized Path Planning for the Mobile Robot Using Potential Field Method and PSO Algorithm. In *Soft Computing for Problem Solving*; Advances in Intelligent Systems and Computing; Springer: Singapore, 2019; Volume 2, pp. 139–150. [CrossRef]
26. Lyu, H.; Yin, Y. Fast Path Planning for Autonomous Ships in Restricted Waters. *Appl. Sci.* **2018**, *8*, 2592. [CrossRef]
27. Park, M.; Lee, M.C. A new technique to escape local minimum in artificial potential field based path planning. *KSME Int. J.* **2003**, *17*, 1876–1885. [CrossRef]
28. Guerra, M.; Efimov, D.; Zheng, G.; Perruquetti, W. Avoiding Local Minima in the Potential Field Method using Input-to-State Stability. *Control Eng. Pract.* **2016**, *55*, 174–184. [CrossRef]

29. Cheng, C.; Chen, Y. A Neural Network based Mobile Robot Navigation Approach using Reinforcement Learning Parameter Tuning Mechanism. In Proceedings of the 2021 China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 2600–2605. [CrossRef]
30. Porta-Garcia, M.; Montiel Ross, O.; Castillo, O.; Sepúlveda, R.; Melin, P. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Appl. Soft Comput.* **2009**, *9*, 1102–1110. [CrossRef]
31. Tarokh, M. Path planning of rovers using fuzzy logic and genetic algorithm. In Proceedings of the World Automation Conference ISORA-026, Maui, HI, USA, 11–16 June 2000; pp. 1–7.
32. Đakulović, M.; Sprunk, C.; Spinello, L.; Petrovic, I.; Burgard, W. Efficient navigation for anyshape holonomic mobile robots in dynamic environments. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 2644–2649. [CrossRef]
33. Karaman, S.; Frazzoli, E. Optimal kinodynamic motion planning using incremental sampling-based methods. In Proceedings of the 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, USA, 15–17 December 2010; pp. 7681–7687.
34. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed.; ACM: New York, NY, USA, 1992; pp. 123–128.
35. Bilinear Interpolation. Available online: https://en.wikipedia.org/wiki/Bilinear_interpolation (accessed on 15 February 2022).
36. Keys, R. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoust. Speech Signal Process.* **1981**, *29*, 1153–1160. [CrossRef]
37. Forsyth, D.; Ponce, J. *Computer Vision: A Modern Approach*; Prentice Hall: Hoboken, NJ, USA, 2011.