RESEARCH ARTICLE

Path-planning with waiting in spatiotemporally-varying threat fields

Benjamin S. Cooper*, Raghvendra V. Cowlagi**

Aerospace Engineering Program, Worcester Polytechnic Institute, Worcester, MA, United States of America

* These authors contributed equally to this work.
** rvcowlagi@wpi.edu

Abstract

We address the problem of finding an optimal path for a vehicle in a planar environment where traversal costs are based on a time-varying spatial field defined over the environment. The resulting optimal path may contain instances of waiting, where the vehicle hovers, parks, or loiters. First, we consider path-planning on a uniform grid over the workspace. It is known that the computational complexity of the problem is significantly higher when waiting is allowed. We study the trade-off between the increased computational complexity and potential cost reductions in the resultant path with allowance for waiting. The results of numerical studies in this work identify characteristics of the threat fields in which optimal paths can involve waiting. Furthermore, we provide a local condition on the threat field that precludes waiting from providing any cost reductions in the resultant path. We show that this condition can be used in the path-planning algorithm to prune search trees and provide significant reductions in computation time without significant suboptimality. Next, we consider path-planning on a vehicle-centric multiresolution grid. We use a wavelet-based multiresolution decomposition to evaluate the multiresolution path planner and compare against the uniform resolution grid using the same family of threat fields. We show that with a vehicle-centric multiresolution map and an appropriate path-planning algorithm, the added computational effort of allowance for waiting is negligible.

1 Introduction

Consider the problem of planning a driving route from one place to another. Minimum-time solutions to this problem are provided by many commercially available gadgets [1] and freely available software applications such as Google Maps [2]. Furthermore, such applications are now capable of using real-time traffic data to provide estimates of the travel duration [3]. Table 1 shows estimated travel durations between the cities Springfield, MA, Boston, MA, and an intermediate town Millbury, MA, which is approximately equidistant from Springfield and Boston. The city halls of each city are chosen as specific starting and ending street addresses. These data are obtained using the Google Maps application on a desktop computer browser, queried at 06:15 AM and again at 07:07 AM on April 10, 2018. Notice the following peculiarity of this result. Per the 06:15 AM query, the estimated travel duration from Springfield to
Table 1. Estimated travel times obtained from the Google Maps application. Data obtained on April 10, 2018.

|                | Spr. to Bos. | Spr. to Mil. | Mil. to Bos. |
|----------------|--------------|--------------|--------------|
| Query at 06:15 AM | 2 hrs. 3 min. | 52 min       | 1 hr. 11 min. |
| Query at 07:07 AM | 2 hrs. 16 min.* | –            | 1 hr. 24 min. |

*: This duration is calculated by adding the Spr. → Mil. duration to the Mil.→ Bos. duration estimated in the 07:07 AM query.

https://doi.org/10.1371/journal.pone.0202145.t001

Millbury, MA is 52 min, and that from Millbury to Boston is 1 hr. 11 min. A vehicle starting from Springfield at 06:15 AM will ostensibly reach Millbury at 07:07 AM. However, in the 07:07 AM query, the estimated travel duration from Millbury to Boston increases by 18%, and the total travel duration from Springfield to Boston will be larger than the 06:15 AM estimate of 2 hrs. 1 min. An explanation of this discrepancy is that the Google Maps application possibly ignores temporal variations in traffic. The algorithmic details of the Google Maps application are proprietary and not publicly known.

Consider now that the objective of the same path planning problem were not to minimize travel duration, but rather to minimize a weighted sum of travel duration and exposure to traffic. Such an objective may be of importance to reduce the health risks to long-haul truck drivers by reducing their exposure to automobile emissions [4–6]. This problem is also of renewed importance because in recent times, a wealth of traffic data has become available to make accurate predictions of traffic [7, 8]. In this case, an optimal route may involve waiting (e.g. at a rest area) for traffic to subside.

This problem is an example of the general problem of path planning with minimum exposure to a scalar field (e.g. traffic density in this example) that varies with space and time. Other applications of this problem include motion-planning for aerial vehicles in inclement weather with physics-based predictive models, and for mobile robots with continually updated estimates of workspace features from noisy measurements.

To serve these motivating applications, we investigate the abstract problem of path planning including waiting for finite intervals of time, in a 2D workspace in the presence of a spatiotemporal threat field, where minimal exposure to the threat is desired, as illustrated in Fig 1. The term “threat” is a generic term to refer to any scalar quantity of interest. The canonical path- and motion-planning problems in the existing literature are of finding obstacle-free paths and/or trajectories between prespecified initial and destination points [9]. Geometric methods [10, 11], randomized sampling-based methods [12, 13], trajectory optimization methods [14], and combinations thereof [15, 16] are reported in the literature to solve these problems. Problems involving predictive models of changes in the environment, where the planner can actively choose to wait for a more “favorable” environment are less thoroughly studied. The robotics literature addresses path-planning with workspace cost maps [17], but the extension of this approach to time-varying cost maps is challenging.

We consider first a scenario where the threat field is fully known, and then a scenario where the field is partially known in vehicle-centric multiresolution detail. The solutions to these problems are conceptually simple: the dimension of the search space is expanded to include the temporal dimension. However, computational implementations are challenging because the search space along this temporal dimension is in principle unbounded even if the 2D spatial workspace is compact.
1.1 Related work

A standard practice to path-planning is the application of Dijkstra’s algorithm or the A* algorithm in a discretized 2D workspace [9, 18], and the point of departure in different approaches is the manner of discretization. Examples of such discretization include quadtree-like workspace cell decompositions [10, 11], visibility roadmaps [19], and other multiresolution grids [20, 21]. In this context, path-planning in a spatiotemporal threat field is formulated by assigning appropriate time-varying edge costs in the associated topological graph.

Optimal path-finding algorithms for graphs with time-varying edge costs appear in [22–25]. The applications reported in the literature of such algorithms are primarily for transportation [26, 27], and for data transmission in networks [24, 28, 29]. The typical objective in such applications is minimum time or minimum delay [30].

An example in which an optimal path can involve waiting is of a passenger at a railway station who must decide whether to board a slower train available immediately or to wait for an express train scheduled for later [30]. An important result is that waiting is never beneficial for minimum time path-planning if the so-called First-In-First-Out (FIFO) condition is satisfied [31]: namely, that departing a position later will not result in arriving earlier (the FIFO condition may not hold for the express train example). However, this condition is in general not satisfied if the cost is different from travel duration, and therefore optimal paths can in general involve waiting. General algorithms for path-planning with waiting to minimize time-varying edge costs appear in [22, 31, 32].

Fig 1. An illustration of the abstract path-planning problem studied in this article. Snapshots of the threat field at different time instants are shown, where higher threat intensities are indicated by darker colors. The vehicle’s location as it moves through the field is indicated by a red dot. (a) Snapshot at $t = 0$ s. (b) Snapshot at $t = 50$ s. (c) Snapshot at $t = 100$ s. (d) Snapshot at $t = 115$ s. The path shown in (d) is a non-waiting minimum exposure path, which will be discussed in the sequel.

https://doi.org/10.1371/journal.pone.0202145.g001
In this paper, we discuss path-planning to minimize a cost defined by the sum of movement cost, waiting cost, and exposure to a spatiotemporal threat. This threat is assumed to be known over the spatial domain of interest (i.e. the vehicle’s workspace) and at all time instants. First, we consider path-planning on a uniform grid over the workspace. It is known that the computational complexity of the problem is higher when waiting is considered [31]. We study the trade-off between the increased computational complexity and potential cost reductions in the resultant path by considering waiting. The results of numerical studies in this work identify characteristics of the field for which optimal paths can involve waiting. Furthermore, we provide a local condition on the threat field that precludes waiting from providing any cost reductions in the resultant path. Next, we consider path-planning on a vehicle-centric multiresolution grid. Recent efforts to utilize non-uniform grids to find optimal paths in spatiotemporal environments include selectively ignoring the time dimension [33] and adaptively discretizing the search grid with respect to variations in a velocity field [34]. However, such multiresolution planning schemes work best when large regions are static [33] or do not explicitly evaluate waiting [34]. We use a wavelet-based multiresolution decomposition to evaluate the multiresolution path planner and compare against the uniform-resolution case using the same family of threat fields.

1.2 Contributions
The contributions of this paper are towards improving the computational efficiency of path-planning algorithms for mobile vehicle applications with time-varying changes in the vehicle’s environment. First, we evaluate the benefit of waiting in a family of threat fields constructed as linear combinations of Gaussian basis functions. Because all square integrable functions can be approximated with arbitrary precision by such Gaussian linear combinations, the results of this paper can be used to train machine learning algorithms to determine for a given application a priori whether or not an optimal path will involve waiting. Second, we present a condition based on local threat field characteristics, to be incorporated in graph search algorithms, that allows pruning search trees that cannot result in an optimal path. We comment on the application of this condition towards path cost reduction and with respect to computational expense through numerical studies. Third, we study the influence of a vehicle-centric multiresolution threat field map, and demonstrate that the added computational expense of considering waiting is negligible if waiting is allowed only in the immediate spatial vicinity of the vehicle. To the best of the authors’ knowledge, there is no literature on multiresolution path-planning with allowance for waiting.

Preliminary results of this work were reported in [35], which addressed a special case of the general problem of path-planning in a time-varying spatial field addressed in this paper. Specifically, [35] did not address waiting, which is the primary focus of this paper. Furthermore, results for only for one representative case study were reported in [35], whereas this paper evaluates the solution over several thousand different randomly generated environments. Therefore, this paper significant advances over [35].

The rest of this paper is organized as follows. Mathematical formulations of the aforesaid problem of interest are presented in Section 2 and its solution methods, including the local no-wait condition, are presented in Section 3. The main results are presented in 4, with conclusions and comments on future work in Section 5.

2 Problem formulations
In what follows, \( \mathbb{R} \) and \( \mathbb{Z} \) represent the sets of reals and integers, respectively.
2.1 Threat field parametrization

We consider a threat field constructed as the weighted sum of a finite number of 2D Gaussian basis functions \( \tilde{c}(X, t) = \sum_{n=1}^{N_p} w_n(t) \phi_n(X, t) \). Here, the spatial basis function \( \phi_n \) is defined for each \( n = 1, \ldots, N_p \) by

\[
\phi_n(X, t) := \frac{1}{\sqrt{2\pi}} \exp \left( \frac{-((X - \mu_n(t))^T (X - \mu_n(t)))}{2 \| \Sigma_n(t) \|^2} \right),
\]

where \( \mu_n(t) = (\mu_{nx}(t), \mu_{ny}(t)) \) defines the spatial mean of \( \phi_n \) and \( \Sigma_n(t) = (\sigma_{nx}(t), \sigma_{ny}(t)) \) defines the spatial spread of \( \phi_n \). In this work, we consider affine functions \( \mu_n \) and \( \Sigma_n \) of the form \( \mu_n(t) = \mu_n0 + \mu_n1t \) and \( \Sigma_n(t) = \Sigma_n0 + \Sigma_n1t \), where \( \mu_n0, \mu_n1, \Sigma_n0, \Sigma_n1 \in \mathbb{R}^2 \) are prespecified constants.

The finite parameterization of the threat field \( \tilde{c} \) using Gaussian functions is justified by the fact that a large class of functions on \( \mathbb{R} \), namely, square integrable functions, can be approximated with arbitrary precision by linear combinations of Gaussian functions [36]. Furthermore, 2D Gaussian functions of varying width can be used to represent varying levels of resolution by including narrower Gaussian functions into higher resolution images or video [37]. Finally, Gaussian function appear in series solutions to several partial differential equations such as the diffusion equation [38], which enables the application of the proposed work to path- and motion-planning with threat field modeled by physical phenomena such as advection-diffusion of gases or radiation in the atmosphere [39].

2.2 Path-planning with uniformly high resolution field map

Let \( \mathcal{W} \subset \mathbb{R}^2 \) be a closed square region, called the workspace, in which the vehicle moves. We consider a strictly positive spatiotemporal scalar field \( \tilde{c}: \mathcal{W} \times [0, \infty) \to \mathbb{R}^+ \), called the threat field, which represents unfavorable regions with higher intensity. For path-planning, we restrict time to a compact interval \( T = [t_0, t_f] \subset \mathbb{R}^+ \). The threat field may represent, for instance, terrain elevation [40], a risk measure [21], a probabilistic occupancy grid [41], or the atmospheric concentration of a gas [39].

First, we consider path-planning on a grid consisting of \( N_G^2 \) points uniformly placed in \( N_G \) rows and \( N_G \) columns. The coordinates in a prespecified Cartesian coordinate axis system of the \( j^{th} \) grid point are denoted by \( X_j \), for each \( j = 1, \ldots, N_G^2 \). The vehicle is assumed to traverse grid points according to a “4-connectivity” rule, and the time taken to traverse between adjacent grid points is a prespecified constant \( t_{step} \). In this paper, we neglect vehicle kinematic and dynamic constraints that can restrict this motion, while noting that such constraints can in the future be incorporated in the proposed grid-world problem setup [16].

We define a graph \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \), where each vertex in \( \mathcal{V} \) is uniquely associated with a grid point, and labeled with superscripts as \( \bar{v}^i, \bar{v}^2, \ldots, \bar{v}^{N_G^2} \). The edge set \( \mathcal{E} \) is defined as the set of pairs of vertices associated with adjacent grid points. A path between vertices \( \bar{v}^{k}, \bar{v}^{k+1} \in \mathcal{V} \), denoted \( \bar{v}(i, j) \), is a finite sequence of vertices \( \{v_0, v_1, \ldots, v_P \} \) with \( v_0 = \bar{v}^{k}, v_P = \bar{v}^{k+1} \), and either \( v_{j-1} = \bar{v}^{i} \) or \( v_{j-1} = \bar{v}^{j} \), for each \( i \in \{1, \ldots, P \} \). Note that subscripts are used to denote indices of vertices. The cost of the path \( \bar{v}(i, j) \) is defined by \( \tilde{g}(\bar{v}) = \sum_{j=1}^{P} \tilde{g}(\bar{v}_{j-1}, \bar{v}_j, t_{step}) \), where \( \tilde{g}: \mathcal{E} \times [0, \infty) \to \mathbb{R}^+ \) is a strictly positive function that assigns time-varying edge transition costs. Specifically,

\[
\tilde{g}(\bar{v}^i, \bar{v}^j, t) = \begin{cases} 
\tilde{c}(X^i, t) + \alpha_m, & \text{if } \bar{v}^k \neq \bar{v}^i, \\
\tilde{c}(X^j, t) + \alpha_m, & \text{if } \bar{v}^k = \bar{v}^j.
\end{cases}
\]

\[ g((\bar{v}^k, \bar{v}^l), t) = \begin{cases} 
\tilde{c}(X^k, t) + \alpha_m, & \text{if } \bar{v}^k \neq \bar{v}^l, \\
\tilde{c}(X^l, t) + \alpha_m, & \text{if } \bar{v}^k = \bar{v}^l.
\end{cases} \]
where $\alpha_m$ and $\alpha_w$ are prespecified strictly positive constants. Note that the preceding definition of a path and its cost implicitly allows for waiting at any grid point, namely, instances where $\bar{v}_{j-1} = \bar{v}_j$ for any $j \in \{1, \ldots, P\}$. The constants $\alpha_w$ and $\alpha_m$ are costs of waiting and of moving, respectively.

**Problem 1.** Find a path $\bar{v}^*(i_s, i_g)$ such that $\bar{J}(\bar{v}(i_s, i_g)) = \bar{J}(\bar{v}^*(i_s, i_g))$ for every other path $\bar{v}(i_s, i_g)$ in $\bar{G}$.

### 2.3 Path-planning with vehicle-centric multiresolution field map

We consider next a multiresolution discretization of the workspace based on the discrete wavelet transform (DWT). This multiresolution discretization algorithm is adopted from the second author’s previous works [35, 42], and provides a so-called *vehicle-centric multiresolution approximation* to the threat field intensity map. The motivation for considering multiresolution discretization is twofold. First, such a discretization may be necessary for practical onboard computations. Second, such a discretization reflects the nature of typical maps available to vehicles in many applications: namely, well-known in the immediate vicinity of the vehicle, partially known in regions farther away, as illustrated in Fig 2. We investigate the potential cost reductions by considering waiting in path-planning. Waiting is allowed only at vertices in the highest resolution region described in Section 2.3. We provide a skeletal overview of this discretization here, and refer the reader to [42] for details.

The DWT represents a scalar field using so-called *approximation and detail coefficients*, which multiply spatial basis functions called *scaling functions and wavelets*. If appropriate, it is possible to use these spatial basis functions to also represent the threat field, without affecting the results presented in this paper. The reasons for using wavelets in this section are the convenient dyadic structure and orthogonality of the basis functions [43].

Without loss of generality, we assume that $W = [0, 1] \times [0, 1]$. For the following discussion, we choose a parameter $D \in \mathbb{Z}_+$ indicating the highest resolution considered in the multiresolution approximation. Specifically, the smallest grid separation (i.e. highest resolution) is $2^{-D}$.

![Fig 2. Example of an intensity map and its vehicle-centric multiresolution approximation according to Eq (8). The vehicle's location is indicated by the black dot near the center. (a) Original field map. (b) Vehicle-centric multiresolution approximation.](https://doi.org/10.1371/journal.pone.0202145.g002)
For context, the grid separation in the uniformly high resolution map considered in the previous subsection is \(2^{-7}\) units.

**Assumption 1.** The field \(\tilde{c}\) is sufficiently smooth such that

\[
\tilde{c}(x, y, t_0) + \sum_{n=1}^{\infty} \frac{(t - t_0)^n}{n!} \frac{\partial^n \tilde{c}(x, y, t)}{\partial t^n}
\]

converges to \(\tilde{c}(x, t)\) for all \(x \in \mathcal{W}\), and \(t \in [0, \infty)\).

**Assumption 2.** The values taken by \(\tilde{c}\) are known at a finite resolution \(m_\ell > -D\), for \(k, \ell = 0, 1, \ldots, 2^{D + m_\ell} - 1\). Without loss of generality, \(m_\ell = 0\).

**Assumption 3.** The values taken by the temporal derivatives of \(\tilde{c}\) at time \(t = t_0\) are known at the same finite resolution as described in Assumption 2.

Relying on Assumption 1, we consider Taylor series expansions of time-varying approximation and detail coefficients of the DWT of the threat field \(\tilde{c}\). The coefficients of these Taylor series expansions are given by

\[
\mathcal{Z}_{m_\ell, k, l}^n \equiv \left\langle \Phi_{m_\ell, k, l}(x, y), \frac{\partial^n \tilde{c}(x, y, t)}{\partial t^n} \right|_{t=t_0} \rightangle,
\]

\[
\mathcal{F}_{m_\ell, k, l}^p \equiv \left\langle \Psi_{m_\ell, k, l}(x, y), \frac{\partial^n \tilde{c}(x, y, t)}{\partial t^n} \right|_{t=t_0} \rightangle,
\]

for \(p = 1, 2, 3, k, \ell \in \mathbb{Z}, m_\ell \geq m \geq m_0 = -D, \text{ and } n \in \mathbb{Z}_{\geq 0}\). Here \(\Phi\) and \(\Psi\) denote families of scaling functions and wavelets, respectively (see [42] for details). The threat field is reconstructed from these coefficients as follows:

\[
\tilde{c}(x, y, t) = \sum_{k, l=0}^{2^{D_{m_\ell}}} \sum_{n=0}^{2^{D_{m_\ell}}} \frac{(t - t_0)^n}{n!} \mathcal{Z}_{m_\ell, k, l}^n \Phi_{m_\ell, k, l}(x, y)
\]

\[
+ \sum_{p=1}^{3} \sum_{m=-m_0}^{m_0} \sum_{k=0}^{2^{D_{m_\ell}}} \sum_{\ell=0}^{2^{D_{m_\ell}}} \frac{(t - t_0)^n}{n!} \mathcal{F}_{m_\ell, k, l}^p \Psi_{m_\ell, k, l}(x, y),
\]

where \(\tilde{c}\) denotes the reconstructed field. To construct a vehicle-centric multiresolution approximation of the threat field, let \(\mathcal{A} \subset \{(m, k, \ell) \in \mathbb{Z}_{3} : m_\ell \leq m < 0, \ 0 \leq k, \ell \leq 2^{D_{m_\ell}}\}\). We define for all \(p = 1, 2, 3\) and \(n \in \mathbb{Z}_{\geq 0}\),

\[
\hat{\mathcal{F}}_m^p \equiv \left\{ \begin{array}{ll} \mathcal{F}_{m_\ell, k, l}^p & (m, k, \ell) \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{array} \right. \]

The reconstruction of the threat field is then performed using \(\hat{\mathcal{F}}_m^p\) in Eq (6) instead of \(\mathcal{F}_{m_\ell, k, l}^p\).

The set \(\mathcal{A}\) contains the indices of detail coefficients that are considered “significant”. Following [42], we choose \(\mathcal{A}\) such that high resolution information is retained in the immediate vicinity of the vehicle’s current location \((x_0, y_0) \in \mathcal{W}\) and it is gradually discarded in regions farther away. To this end, let \(\varphi : \mathbb{Z} \to \mathbb{N}\) be a “window” function that specifies, for each level of resolution, the distance from the vehicle’s location up to which the detail coefficients at that level are
significant. The set $A = \mathcal{A}^{\text{win}}(x_0, y_0)$ of indices is then defined by

$$
\mathcal{A}^{\text{win}}(x_0, y_0) := \left\{ d^*_{m,k}(m, k, \ell) : m_0 \leq m < 0, 
2^m x_0 \leq k \leq 2^m x_0 + g(m), 
2^m y_0 - \ell \leq \ell \leq 2^m y_0 + g(m) \right\}. 
$$

The preceding description of the vehicle-centric multiresolution approximation is minimal; the interested reader is referred to [35, 42] for further details. In what follows, we focus on the following problem.

**Problem 2.** Solve Problem 1 using a vehicle-centric multiresolution approximation $\hat{c}$ of the threat field.

To formulate Problem 2 precisely, we define a multiresolution cell decomposition $\Omega^{\text{mr}}$, which is a partition of $VV$ into square cells of different sizes, such that $\hat{c}$ is constant in the spatial variables over each of the cells. We denote by $\text{cell}(v^k; \Omega)$ the coordinates of the center of the cell associated with a vertex $v^k \in V$, and by $\text{vert}(C; \bar{G})$ the vertex in $\bar{V}$ associated with a cell $C \in \Omega$. We attach with the cell decomposition $\Omega^{\text{mr}}$ a graph $G = (V, E)$ such that each cell in $\Omega^{\text{mr}}$ corresponds to a unique vertex in $V$. Each vertex $j \in V$ corresponds to a set $W(v^k, V) \subset V$, and the collection $\{W(v^k, V)\}_{v^k \in V}$ is a partition of $V$. Specifically:

$$
W(v^k, V) := \{ v^k \in \bar{V} : \text{cell}(v^k; \Omega) \subset \text{cell}(v^k; \Omega^{\text{mr}}) \}. 
$$

Vertices $v^k, v^\ell \in V$ are adjacent in $G$ if there exist $v^k \in W(v^k, V)$ and $v^\ell \in W(v^\ell, V)$ such that $\{v^k, v^\ell\} \in E$. We define edge costs $g : E \rightarrow \mathbb{R}_+$ in $G$ as

$$
g((v^k, v^\ell), t) := \begin{cases} 
\tilde{c}(X^k, t) + z_m, & \text{if } |W(v^k, V)| = 1, v^k = v^\ell, \\
\tilde{c}(v^k, v^\ell) + z_m |W(v^\ell, V)|, & \text{otherwise},
\end{cases}
$$

where $c_\ell$ are time-averaged cell intensities. Owing to the use of the DWT, the computation of these time-averaged intensities involves simple algebraic computations (see [35] for details). This edge cost function considers approximate periods of time and distance required to traverse these cells, which are in turn related to the cell sizes $|W(v^k, V)|$. We restrict waiting to only the uniform resolution window of $\Omega^{\text{mr}}$, precisely, when $|W(v^k, V)| = 1$ and $v^k = v^\ell$ in Eq (10). The cost $J(v)$ of a path in $G$ is the sum of all edge costs.

**3 Methods**

**3.1 General solution to Problem 1**

Problem 1 is similar to a standard path-planning problem with the exception that the edge transition costs are time-varying. This exception, however, significantly complicates the solution of this problem. Specifically, to solve Problem 1, a search must be performed not on the graph $G$ but instead on a much larger graph $\bar{G}T$ obtained as follows. Let $T$ be a line graph whose vertices are the time instants $t_0, t_1, \ldots$, and whose edges are the pairs $(t_{j-1}, t_j)$, where $t_j - t_{j-1} = \text{step}$ for each $j \in \mathbb{N}$. The graph $\bar{G}T$ is then defined as the product of graphs $\bar{G}$ and $T$, i.e., each vertex in $\bar{G}T$ is a pair $(\tilde{v}^k, t_j)$, and the pairs $(\tilde{v}^k, t_j)$ and $(\tilde{v}^\ell, t_m)$ form an edge of $\bar{G}T$ if and only $t_m - t_j = \text{step}$ and either $\tilde{v}^k = \tilde{v}^\ell$ or $(\tilde{v}^k, \tilde{v}^\ell) \in E$, with $k, \ell \in \{1, \ldots, N_G^2\}$ and $j, m \in \mathbb{N}$. The worst-case complexity of Dijkstra’s algorithm for solving Problem 1 is $O(|\bar{G}T| (1 + \log(|\bar{G}T|)))$.

In typical path-planning algorithms, Dijkstra’s algorithm is sped up using a search heuristic: for example, the A* algorithm uses Euclidean distance to the goal as a search heuristic [18].
For Problem 1, we can define for each vertex $v^k \in V, k = 1, 2, \ldots, |V|$, we define the search heuristic

$$h(v^k) := \alpha_m \| x^k - x^f \|, \quad (11)$$

which is Euclidean distance to the goal scaled by the positive constant $\alpha_m$. The scale factor ensures that the heuristic $h$ underestimates the true cost to the goal, and therefore ensures that the search algorithm returns an optimal path [18]. Note that the worst-case computational complexity of the search algorithm does not change despite using the search heuristic [18], although a significant speed-up is achieved for most $(v^k, v^f)$ pairs of initial and goal vertices.

### 3.2 No-wait path-planning

An approximate solution to Problem 1 can be obtained by ignoring the possibility of waiting. To this end, we define a no-wait path as a path $\bar{v} = (v_0, v_1, \ldots, v_p)$ where $v_{j-1} \neq v_j$ for each $i \in \{1, \ldots, P\}$. We define an edge transition cost function:

$$\bar{\gamma}_m((v^k, \bar{v}), t) := \bar{c}(x^k, t) + \alpha_m. \quad (12)$$

The cost of a no-wait path is then defined as $\bar{J}_{nw}(\bar{v}) := \sum_{j=0}^{p} \bar{\gamma}_m((v_{j-1}, v_j)); J_{step}$.

Compared to solving Problem 1 exactly as in Section 3.1, it is significantly easier to find a path $\bar{v}^*_m(i_1, i_1)$ with minimum no-wait cost $\bar{J}_{nw}(\bar{v}^*_m)$. The worst-case complexity of Dijkstra’s algorithm to find this path is $O(|\bar{G}|(1 + \log |\bar{G}|))$, and $|\bar{G}|$ is smaller than $|G \times T|$ by the order of magnitude of $|T|$.

It is therefore natural to question: (1) whether the computational effort in exactly solving Problem 1 is worthwhile: i.e., whether the suboptimality of the $\bar{v}^*_m$, measured as $\bar{J}(\bar{v}^*_m) - \bar{J}(\bar{v}^*)$, is large enough to justify spending the significantly higher computational resources to find the true solution $\bar{v}^*$ to Problem 1, and (2) whether the exact solution $\bar{v}^*$ can be computed faster. In the sequel we address both of these questions.

To address the first question (on computational effort), we characterize the difference $\bar{J}(\bar{v}^*_m) - \bar{J}(\bar{v}^*)$ based on a study of a large number of numerical simulations (discussed in Section 4). The results of this study identify the threat fields where the difference $\bar{J}(\bar{v}^*_m) - \bar{J}(\bar{v}^*)$ is significant. This study consisted of 2000 simulations of different instances of Problem 1. In each simulation, the threat field $\bar{c}$ was constructed with an arbitrary $N_T$ chosen from the set $\{3, 4, \ldots, 40\}$, followed by arbitrary choices of the constants $w_{stop}, w_{stop}, \mu_{max}, \mu_{min}, \Sigma_{step}$, and $\Sigma_{final}$ for each $n = 1, \ldots, N_p$. The weights $w_{stop}, w_{stop}$ were fixed at 1 so that all peaks in the field are of uniform height. The parameters $\mu_{max}, \mu_{min}$ were chosen by random sampling on a uniform distribution over the workspace $W$. The parameters $\Sigma_{step}, \Sigma_{final}$ were chosen by random sampling from a uniform distribution a quarter the size of the workspace $W$. The simulations were run using $N_G^0 = 1600, t = [0, 100]$, and $t_{step} = \frac{\Sigma_{final}}{N_G^0}$, where $V_{final}$ is the width of the (square) workspace. The product graph $G \times T$ in each simulation has approximately $1.2 \times 10^6$ vertices.

The simulations were designed using MATLAB$^\text{TM}$ and executed on a Windows 7 Enterprise$^\text{TM}$ computer using an Intel$^\text{TM}$ Core$^\text{TM}$ i7-4770 3.4 GHz CPU and 16 GB RAM. The results of this study are discussed in Section 4, and MATLAB$^\text{TM}$ source code is available at https://github.com/rvcowlagi/waiting.

The issue of interest here is whether it is beneficial to search in the larger product graph $G \times T$ or the smaller topological graph $G$. This issue depends on the properties of the threat field itself, and not on the specific software implementation of the search algorithm. Therefore, simulation results in MATLAB$^\text{TM}$ suffice to study this issue. Real-time implementations of the
The number of computations in solving Problem 1 can be reduced by pruning edges of $\tilde{G}T$ during the execution of Dijkstra’s algorithm. Specifically, an edge of $\tilde{G}T$ between the pairs $(\tilde{v}^k, t_j)$ and $(\tilde{v}^k, t_{j+1})$, $k \in \{1, \ldots, N_G^2\}, j \in \mathbb{N}$, can be pruned if a condition, as identified next, precludes waiting at $x^k$ at time $t_{j+1}$ from being optimal.

Consider the cost of moving from $x^k$ to an adjacent point $x^\ell$, with $k, \ell \in \{1, \ldots, N_G^2\}$. Accordingly, consider the vertex $(\tilde{v}^k, t_j)$ of $\tilde{G}T$, and two paths for traveling to $x^\ell$. The first path is the single edge in $\tilde{G}T$ from vertex $(\tilde{v}^k, t_j)$ to vertex $(\tilde{v}^\ell, t_{j+1})$ achieves this travel. The second path consists of two successive edges from vertex $(\tilde{v}^k, t_j)$ to vertex $(\tilde{v}^\ell, t_{j+1})$ to vertex $(\tilde{v}^\ell, t_{j+2})$ also achieves this travel, but includes waiting at the point $x^k$ for one time step (see Fig 3). Waiting at $x^k$ is beneficial if the cost of the first path is greater than the cost of the second path, i.e., if

$$\tilde{g}((\tilde{v}^k, \tilde{v}^\ell), t_{j+1}) + \tilde{g}((\tilde{v}^\ell, \tilde{v}^\ell), t_{j+2}) < \tilde{g}((\tilde{v}^k, \tilde{v}^\ell), t_{j+1}),$$  \hspace{1cm} (13)

$$\Rightarrow \tilde{c}(x^k, t_{j+1}) + \alpha_m + \tilde{c}(x^\ell, t_{j+2}) < \tilde{c}(x^\ell, t_{j+1}),$$  \hspace{1cm} (14)

For further insight into this condition, consider a two vertex system with current position $x^k$, and goal position $x^\ell$. Fig 3 illustrates the possible paths and costs. For brevity, define $F_{k,j+1} := \tilde{c}(x^k, t_{j+1}), F_{\ell,j+1} := \tilde{c}(x^\ell, t_{j+1})$, and $F_{\ell,j+2} := \tilde{c}(x^\ell, t_{j+2})$. Waiting is beneficial if

$$\alpha_m + F_{k,j+1} + \alpha_m + F_{\ell,j+2} < \alpha_m + F_{\ell,j+1} \Rightarrow \alpha_m + F_{k,j+1} < F_{\ell,j+1} - F_{\ell,j+2},$$ \hspace{1cm} (15)

Fig 3. Illustration of the local no-wait condition. The decision to wait at vertex $v^k$ or immediately move to vertex $v^\ell$ is based on the future threat field values $\tilde{c}(x^k, t_{j+1}), \tilde{c}(x^\ell, t_{j+1}), \tilde{c}(x^\ell, t_{j+2})$, and waiting and movement costs $\alpha_m$ and $\alpha_m$. 

https://doi.org/10.1371/journal.pone.0202145.g003
and it is easy to see that,

$$a_w + \bar{c}(x^k, t_{j+1}) < -\Delta c(x^k + \Delta x, t_{j+1}) \Delta t$$  \hspace{1cm} (16)

The terms in (16) suggest that an optimal path can involve waiting when the sum of $a_w$ and the threat in the next time step $\bar{c}(x^k, t_{j+1})$ is less than the difference between the field costs at the next vertex $-\Delta c(x^k + \Delta x, t_{j+1}) \Delta t$, where $x' = x^k + \Delta x$. This occurs when the waiting costs and field values are low relative to large decreasing changes in the field threat. In other words, it is beneficial to wait if the threat along the optimal path $\bar{v}^*$ is rapidly diminishing and the cost to remain at current position $x^k$ is low. This observation is supported via numerical experiments. We identify a local no-wait condition as follows.

$$a_w + \bar{c}(x^k, t_{j+1}) + \bar{c}(x^l, t_{j+2}) - \bar{c}(x^l, t_{j+1}) \geq 0$$  \hspace{1cm} (17)

When executing Dijkstra’s algorithm, edges can be pruned using (17). If (17) holds true, then the two-vertex local condition is violated. To decrease the number of edges explored in the graph $G_T$, the neighbors of $v^k$ are evaluated under (17) at each iteration. If any neighbor violates the waiting condition, then $\bar{v}^k$ is marked as a non-waiting node.

### 3.4 Solution to Problem 2

The multiresolution path-planning algorithm utilizes Dijkstra’s algorithm to search the product $G_T$ of graph $G$ and $T$.

Fig 4 shows in pseudo-code form the proposed path-planning algorithm based on the vehicle-centric multiresolution approximation of the spatial field $\bar{c}$. The algorithm iterates Lines 3–10 until the goal is reached. At each iteration, the algorithm computes a vehicle-centric multiresolution approximation and the corresponding cell decomposition graph (Lines 1–4). In Line 5, the optimal path in $G_n$ is computed by a label-correcting algorithm. The vehicle is assumed to traverse the first cell in the path $\pi_n^*$, and the process is repeated for the new vehicle location.

**Multiresolution Path-planning with Time-varying Costs**

```plaintext
procedure MR-APPROX(j)
  1: $A := A^{\text{win}}(\text{cell}((\bar{v}^k, G)))$ using Eqn. (8)

procedure MAIN
  1: $v := \bar{v}^n$, $v_0 := \bar{v}^k$, $n := 0$, $AtGoal := 0$, $\bar{J}(v) := 0$
  2: while $\neg AtGoal$ do
  3:   $A_n := MR-APPROX(v_n^k)$
  4:   $G_n := MR-GRAPH(A_n)$
  5:   $v_{n+1} := \arg\min\{J(v) : v \text{ is a path in } G_n\}$
  6:   $v_{n+1}^k := \text{vert}(\text{cell}(v_{n+1}^k, G_n))$, where $v_{n+1}^l$ is the second vertex in the path $v_{n+1}^*$
  7:   $AtGoal := (v_{n+1}^k = \bar{v}^k)$
  8:   $\bar{v} := (v, v_n^k)$
  9:   $\bar{J}(v) := \bar{J}(\bar{v}) + 3(\bar{v}_{n+1}^k, n\Delta t)$
  10:  $n := n + 1$
```

Fig 4. Pseudo-code for the proposed path-planning algorithm.

https://doi.org/10.1371/journal.pone.0202145.g004
A procedure to determine the locations and the sizes of cells in $\Omega^{\text{mr}}$ in the vehicle-centric multiresolution approximation, including fast updates of these cell locations and sizes with the changing vehicle location, is provided in [42]. Furthermore, a procedure denoted MR-GRAPH to determine the edges in the graph $G$ associated with $\Omega^{\text{mr}}$, including fast updates to the sets of vertices and edges of this graph with the changing vehicle location, is also provided in [42].

One detail not explicitly stated in the pseudo-code in Fig 4 is the ability to bypass the optimization problem in Line 5 when the multiresolution decomposition in Lines 1–4 is unchanged. Because of the vehicle-centric decomposition, if the vehicle has found a waiting beneficial path it will remain stationary for several iterations while it steps through the optimal space-time path, $v^\ast$. In this case, Line 5 can be bypassed until the vehicle moves and the decomposition updates. For the multiresolution planning, this significantly speeds up the planning and allows the planner which considers waiting to compute in the nearly the same time as the no-wait planner as will be seen in the results section.

The topological properties (i.e. cell locations, sizes, and adjacency relations) of the vehicle-centric multiresolution approximation are the same as in [42] and therefore, the proof of completeness of this path-planning algorithm is the same as that provided in detail in [42].

4 Results and discussion

The results in this section are summarized as follows. The allowance for waiting in the path-planning algorithm can reduce paths costs by 25% compared to no-wait path-planning, but this cost reduction incurs an added computational expense of a 30- to 300-fold increase in execution time. Using the local no-wait condition of Section 3.3 reduces this added computation expense to only a 10-fold increase in execution time, while discovering a majority of optimal paths. In comparison to the uniform resolution case, multiresolution discretization has strong computational benefits: specifically, the added computational expense of the allowance for waiting in path-planning is negligible.

4.1 Empirical study with uniformly high resolution map

In what follows, we present the results of the numerical study described in Section 3.2, specifically, the characteristics of the difference in costs between the path solving Problem 1 and the no-wait path: $\mathcal{J}(v_{\text{nw}}^\ast) - \mathcal{J}(v^\ast)$ normalized by $\mathcal{J}(v^\ast)$.

Computational effort for allowance of waiting. Over the 2000 simulations conducted, the greatest suboptimality of the no-wait path was 25%. In 156 of these simulations, i.e., nearly 8% of the simulated cases, $v_{\text{nw}}^\ast$ was suboptimal compared to $v^\ast$, but this suboptimality was significant (greater than 5%) in only in less than 1% of cases of the 2000 simulations. The computation times for finding the no-wait path in each simulation were all under 0.4 s, while those for finding the solution to Problem 1 were in the range of 11–120 s as shown in the histogram in Fig 5. In other words, the computation time to find the solution $v^\ast$ to Problem 1 was between 30–300 times slower than the time to find the no-wait approximate solution $v_{\text{nw}}^\ast$, whereas the suboptimality $\mathcal{J}(v_{\text{nw}}^\ast) - \mathcal{J}(v^\ast)$ was greater than zero in slightly less than 8% of the cases simulated. In certain applications, however, this extra computational effort may be justified, say in the transportation of hazardous waste through adverse weather (threat) [25].

Local no-wait condition—Cost reduction and computational efficiency. The discussion in Section 3.3 indicates that waiting is locally beneficial when the waiting cost $\alpha_w$ and the threat field value are low compared to the local temporal gradient of the threat field. It was noted that the movement cost $\alpha_m$ did not affect this local test, and therefore the ratio $\alpha_m/\alpha_w$ is not relevant to the waiting decision. In Fig 6(a), we note that the cost reduction of waiting occurs
more frequently with a lower minimum values of the field $\tilde{c}$. Fig 6(b)–6(d) summarize the average value across time for the field and its gradients.

When the field temporal gradient is small, i.e., $\nabla \tilde{c}(x, t) \rightarrow 0$, the field is approximately static and an optimal path does not involve waiting. For further insight into the lack of waiting paths for fields with large temporal gradients, consider a vehicle at location $x^k$ at time $t_j$ and the threat value $\tilde{c}(x', t_j)$ at location $x'$ along the optimal path $v^*$. Due to the large gradient, the threat will move or dissipate at time $t_{j+m\Delta t}$. If the threat at position $x'$ dissipates such that $\nabla \tilde{c}(x') \geq \frac{dx}{dt}$, where $\frac{dx}{dt}$ is the speed of the vehicle, then the vehicle arrives after the threat is dissipated and no waiting is required. However, if the vehicle arrives at position $x'$ before time $t_{j+m\Delta t}$, it will incur a threat exposure cost and should wait. These arguments indicate that optimal paths may involve waiting when the field changes moderately over time, i.e., $\nabla \tilde{c}(x, t)$ is neither too high nor too low.

As discussed in Section 3.3, we can modify Dijkstra’s algorithm with the local no-wait condition (16), which leads to a significant computational advantage. Using this local no-wait condition check, the average calculation time is reduced from 78.7s to 5.47s, which is a 140% reduction (see Fig 7). With the local no-wait condition check, 83% of paths are determined in under 6 seconds while still discovering optimal paths that involve waiting. Because this no-wait condition is approximate, potentially optimal waiting paths can be discarded. Among the 2000 simulations conducted, 156 cases involved waiting in the optimal path, but with the local no-wait condition check included, only 13 of these 156 waiting paths were found by the
algorithm. However, these included paths with the greatest reduction in cost: specifically, the reduction of cost of the waiting paths (compared to an optimal no-wait path) with a cost difference greater than 5%, 57% were found by the algorithm including the local no-wait check condition. In summary, the local no-wait condition was observed to significantly reduce computational efforts while finding optimal paths in a majority of the simulated cases.

Fig 6. Difference in costs of waiting-allowed and no-wait optimal paths, for different field characteristics. When the minimum field value is low, optimal paths more frequently include waiting. The greatest differences between paths with and without waiting cluster near a median value suggesting that large and small temporal gradients of the field favor movement over waiting.

https://doi.org/10.1371/journal.pone.0202145.g006

Algorithm computation time for the majority of paths is significantly reduced while still finding optimal paths that include waiting.

https://doi.org/10.1371/journal.pone.0202145.g007
Other dependencies. Fig 8 indicates that a low value of the movement cost constant $\alpha_m$ in the edge transition cost (2), the optimal path more frequently involves waiting. Such cases may occur when the optimal no-wait path $v_{nw}$ is shorter (i.e., has fewer vertices) but has higher path cost $J(v_{nw})$ compared to when $v_{nw}$ is longer than $v_w^*$, but has lower cost path due to threat exposure. For these simulations the waiting cost constant $\alpha_w$ was restricted to the interval (0, 0.1].

The number of parameters $N_P$ used to define the threat field also demonstrated a pattern with respect to cost-reduced waiting paths. In Fig 9, an increase in $N_P$ is associated with a greater number of optimal paths involving waiting. An increase in the density of peaks in the field may lead to lower spatial gradients of the field (min and max), which influences optimal waiting-allowed paths in a manner similar to the aforesaid influence of the temporal gradient.
4.2 Empirical study with vehicle-centric multiresolution map

An investigation with the multiresolution representation was initially performed using a threat field with three parameters (i.e., $N_p = 3$) as shown in Fig 10, corresponding to three “peaks” of the threat field. The center peak decreases in magnitude over time, therefore the optimal path should wait as long as possible then traverse through the center of the field. This scenario was run at four different resolutions, $N_G = \{16, 32, 64, 128\}$, corresponding to $D = \{4, 5, 6, 7\}$ and the results of these simulations are summarized in Table 2. Recall that the parameter $D$ indicates the highest resolution considered in the multiresolution approximation. With $D = 6$ the total computation time for finding the optimal path with the multiresolution map (i.e. solving Problem 2) is of the same order of magnitude as that for finding the optimal path in the uniform resolution map ((i.e. solving Problem 1)). With $D = 7$, computation with the multiresolution map is an order of magnitude faster. This computational speed is achieved with

![Fig 10. Waiting-allowed path with a multiresolution map of a three peak Gaussian field. Here, the center peak fades over time for $D = 4$ and 7 (a and b). The multiresolution path is in blue boxes and the uniform resolution path in solid red. Note the tenfold reduction in calculation time for the $D = 7$ multiresolution case.](https://doi.org/10.1371/journal.pone.0202145.g010)

### Table 2. Summary of cost of and calculation times.

| $D$ | Allow wait | No wait |
|-----|------------|---------|
|     | Uniform | Multires. | Uniform | Multires. |
| 4   | Path cost | 3.41 | 3.43 | 3.57 | 3.88 |
|     | Calc. time | 4.66 s | 256.7 s | 0.79 s | 92.1 s |
| 5   | Path cost | 6.57 | 6.64 | 7.39 | 7.42 |
|     | Calc. time | 59.9 s | 743.9 s | 13.0 s | 358.0 s |
| 6   | Path cost | 12.91 | 13.04 | 14.72 | 14.72 |
|     | Calc. time | 2305 s | 2489 s | 375.8 s | 1284.2 s |
| 7   | Path cost | 26.04 | 26.37 | 29.62 | 29.76 |
|     | Calc. time | 104,940 s | 10,600 s | 21,835 s | 5,978 s |

https://doi.org/10.1371/journal.pone.0202145.t002
suboptimality, i.e., the cost of the solution of Problem 2 is always greater than or equal to the cost of solution of Problem 1. However, with $D = 7$, the maximum such suboptimality observed across all cases was approximately 1%. Next, we discuss this suboptimality with lower values of $D$.

In addition to the three peak field, the numerical study reported in the Section 4.1 was repeated for the multiresolution cases $D = \{4, 5, 6\}$. All parameters of the threat fields remain the same. A set of 100 simulations (in addition to the previous set of 2000 simulations) was run for each $D = \{4, 5, 6\}$, and each simulation included four calculations: waiting with uniform and multiresolution and non-waiting with uniform and multiresolution. The major results are summarized in Figs 11 and 12.

**Fig 11.** Percentage cost reduction when waiting is considered using the multiresolution representation at different resolution levels. Negative cost reductions (or sub-optimality of no-wait paths) indicate that the algorithm anticipated a benefit from waiting but was mistaken.

https://doi.org/10.1371/journal.pone.0202145.g011

**Fig 12.** Suboptimality of using the multiresolution map, with different values of the parameter $D$.

https://doi.org/10.1371/journal.pone.0202145.g012
Fig 12 shows the difference between uniform- and multiresolution cases when both path planners allow waiting, i.e., the difference in the cost of the solution of Problem 2 to that of Problem 1. Therefore, this result is an indication of the suboptimality incurred due to the vehicle-centric multiresolution approximation of the field. This suboptimality increases with increasing values of $D$. This observation is explained as follows. Allowance for waiting is made inside of the high-resolution window of the multiresolution approximation. As $D$ increases, the area of this window is a smaller portion of the overall workspace, and therefore the multiresolution path planner is “myopic” and suboptimal.

Finally, we observe the average computation times for the four cases of waiting-allowed vs. no-wait paths and uniform vs. multiresolution maps, as summarized in Fig 13. It is immediately clear that the multiresolution map bears computational advantages as the resolution parameter $D$ increases. Specifically, \textit{the average computation time between waiting and non-waiting for the multiresolution cases is nearly equal.} This result was previously alluded to when describing the path planning algorithm in Fig 4. Whenever the search algorithm explores waiting, the spatial multiresolution decomposition does not change. Therefore, in the multiresolution path-planning problem, the allowance of waiting does not incur significant additional computational expense.

5 Conclusions

In this paper, we studied the inclusion of waiting at vertices in path-planning for minimizing exposure to a time-varying threat field. We proposed a local no-wait condition based on the threat field characteristics, which decides if waiting should be considered at the current vertex.
in the search algorithm. Through numerical simulations, the threat field characteristics when optimal paths can involve waiting were explored and the local no-wait condition was demonstrated. The local no-wait condition was successful in reducing the majority of test case's computation time by 140% while finding an optimal path in a majority of the simulated cases. We extended the application of multiresolution path planning to allow waiting and found that the vehicle-centric multiresolution approximation of the threat still allowed the discovery of beneficial waiting paths. Importantly, we observed that including waiting in to the multiresolution planner has a negligible computational expense.

Author Contributions
Conceptualization: Benjamin S. Cooper, Raghvendra V. Cowlagi.
Data curation: Benjamin S. Cooper.
Funding acquisition: Raghvendra V. Cowlagi.
Investigation: Benjamin S. Cooper, Raghvendra V. Cowlagi.
Methodology: Benjamin S. Cooper, Raghvendra V. Cowlagi.
Project administration: Raghvendra V. Cowlagi.
Software: Benjamin S. Cooper, Raghvendra V. Cowlagi.
Validation: Benjamin S. Cooper.
Visualization: Benjamin S. Cooper.
Writing – original draft: Benjamin S. Cooper, Raghvendra V. Cowlagi.
Writing – review & editing: Benjamin S. Cooper, Raghvendra V. Cowlagi.

References
1. CNET Editorial Board. Best GPS Systems for 2018; 2017. Online: https://www.cnet.com/topics/gps/best-gps/.
2. Google Inc. Google Maps; 2018. Online: https://maps.google.com.
3. Lowe M. Get real-time commute info and more in one tap; 2017. Online: https://blog.google/products/maps/get-real-time-commute-info-and-more-one-tap/.
4. Peters A, von Klot S, Heier M, Trentinaglia I, Hörmann A, Wichmann HE, et al. Exposure to Traffic and the Onset of Myocardial Infarction. New England Journal of Medicine. 2004; 351(17):1721–1730. https://doi.org/10.1056/NEJMoa040203 PMID: 15496621
5. Raaschou-Nielsen O, Andersen ZJ, Hvidberg M, Jensen SS, Ketzel M, Sørensen M, et al. Lung Cancer Incidence and Long-Term Exposure to Air Pollution from Traffic. Environmental Health Perspectives. 2011; 119(6):860–865. https://doi.org/10.1289/ehp.1002353 PMID: 21227886
6. Rice MB, Ljungman PL, Wilker EH, Dorans KS, Gold DR, Schwartz J, et al. Long-term exposure to traffic emissions and fine particulate matter and lung function decline in the Framingham heart study. American journal of respiratory and critical care medicine. 2015; 191(6):656–664. https://doi.org/10.1164/rccm.201410-1875OC PMID: 25590631
7. Lv Y, Duan Y, Kang W, Li Z, Wang FY. Traffic Flow Prediction With Big Data: A Deep Learning Approach. IEEE Transactions on Intelligent Transportation Systems. 2015; 16(2):865–873.
8. Tan H, Wu Y, Shen B, Jin P-J, Ran B. Short-Term Traffic Prediction Based on Dynamic Tensor Completion. IEEE Transactions on Intelligent Transportation Systems. 2016; 17(8):2123–2133. https://doi.org/10.1109/TITS.2015.2513411
9. LaValle SM. Planning Algorithms. Cambridge University Press; 2006.
10. Kambhampati S, Davis LS. Multiresolution path planning for mobile robots. IEEE Journal of Robotics and Automation. 1986; RA-2(3):135–45. https://doi.org/10.1109/JRA.1986.1087051
11. Latombe JC. Robot Motion Planning. Kluwer Academic Publishers; 1991.
12. LaValle SM, Kuffner JJ jr. Randomized Kinodynamic Planning. International Journal of Robotics Research. 2001; 20(5):378–400. https://doi.org/10.1177/02783640122067453
13. Karaman S, Frazzoli E. Sampling-based Algorithms for Optimal Motion Planning. International Journal of Robotics Research. 2011; 30:846–894. https://doi.org/10.1177/0278364911406761
14. Betts JT. Survey of Numerical Methods for Trajectory Optimization. Journal of Guidance, Control, and Dynamics. 1998; 21(2):193–204. https://doi.org/10.2514/2.4231
15. Plaku E, Kavraki LE, Vardi MY. Motion Planning with Dynamics by a Synergistic Combination of Layers of Planning. IEEE Transactions on Robotics. 2010; 26(3):469–482. https://doi.org/10.1109/TRO.2010.2047820
16. Cowlagi RV, Tsiotras P. Hierarchical Motion Planning with Dynamical Feasibility Guarantees for Mobile Robotic Vehicles. IEEE Transactions on Robotics. 2012; 28(2):379–395. https://doi.org/10.1109/TRO.2011.2171613
17. Jaillet L, Cortés J, Siméon T. Sampling-Based Path Planning on Configuration-Space Costmaps. IEEE Transactions on Robotics. 2010; 26(3):647–659.
18. Nilsson NJ. Artificial Intelligence: A New Synthesis. San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc.; 1998.
19. Lozano-Pérez T. An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles. Communications of the ACM. 1979; 22(10):560–570. https://doi.org/10.1145/359156.359164
20. Behnke S. Local multiresolution path planning. Lecture Notes in Artificial Intelligence. 2004; 3020:332–43.
21. Tsiotras P, Bakolas E. A Hierarchical On-Line Path Planning Scheme using Wavelets. In: Proceedings of the European Control Conference. Kos, Greece; 2007. p. 2806–2812.
22. Orda A, Rom R. Minimum-weight Paths in Time-Dependent Networks. Networks. 1991; 21:295–319. https://doi.org/10.1002/net.3230210304
23. Philpott AB, Mees AI. A Finite-time Algorithm for Shortest Path Problems with Time-varying Costs. Applied Mathematical Letters. 1993; 6(2):91–94. https://doi.org/10.1016/0893-9659(93)90020-N
24. Cai X, Kloks T, Wong CK. Time-varying shortest path problems with constraints. Networks. 1997; 29(3):141–150. https://doi.org/10.1002/(SICI)1097-0037(199705)29:3<141::AID-NET2>3.0.CO;2-H
25. Akgün V, Parekh A, Batta R, Rump CM. Routing of a hazmat truck in the presence of weather systems. Computers & Operations Research. 2007; 34:1351–1373. https://doi.org/10.1016/j.cor.2005.06.005
26. Pallottino S. Shortest path algorithms in transportation models: classical and innovative aspects. Equilibrium and advanced transportation. 1998; 13(1):245–281. https://doi.org/10.1007/978-1-4615-5757-9_11
27. Berube JF, Potvin JY, Vaucher J. Time-dependent shortest paths through a fixed sequence of nodes: Application to a travel planning problem. Computers and Operations Research. 2006; 33(6):1838–1856. https://doi.org/10.1016/j.cor.2004.11.021
28. Dell'Amico M, Iori M, Pretolani D. Shortest paths in piecewise continuous time-dependent networks. Operations Research Letters. 2008; 36(6):688–691. https://doi.org/10.1016/j.orl.2008.07.002
29. Ganguly S, Sen A, Xue G, Hao B, Shen BH. Optimal routing for fast transfer of bulk data files in time-varying networks. 2004 IEEE International Conference on Communications (IEEE Cat No04CH37577). 2004;2(c):1182–1186.
30. Orda A, Rom R. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. Journal of the ACM. 1990; 37(3):607–625. https://doi.org/10.1145/79147.214078
31. Dean BC. Algorithms for minimum-cost paths in time-dependent networks with waiting policies. Networks. 2004; 44(1):41–46. https://doi.org/10.1002/net.20013
32. Chabini I. Discrete Dynamic Shortest Path Problems in Transportation Applications: Complexity and Algorithms with Optimal Run Time. 2013; 53(9):1689–1699.
33. Vemula A, Muelling K, Oh J. Path planning in dynamic environments with adaptive dimensionality. In: Ninth Annual Symposium on Combinatorial Search; 2018.
34. Kulatane D, Bhattacharya S, Hsieh MA. Optimal Path Planning in Time-Varying Flows Using Adaptive Discretization. IEEE Robotics and Automation Letters. 2018; 3(1):458–465. https://doi.org/10.1109/LRA.2017.2761939
35. Cowlagi RV. Multiresolution Path-planning with Traversal Costs based on Time-varying Spatial Fields. In: Proceedings of the 53rd IEEE Conference on Decision & Control. Los Angeles, CA, USA; 2014. p. 3745–3750.
36. Calcaterra C. Linear Combinations of Gaussians with a Single Variance are dense in L2. In: Proceedings of the World Congress on Engineering. vol. 2; 2008.
37. Goshtasby A, Oneill WD. Curve fitting by a sum of Gaussians. CVGIP: Graphical Models and Image Processing. 1994; 56(4):281–288.

38. Crank J. The mathematics of diffusion. Oxford university press; 1979.

39. Demetriou MA, Gatsonis NA, Court JR. Coupled Controls-Computational Fluids Approach for the Estimation of the Concentration From a Moving Gaseous Source in a 2-D Domain With a Lyapunov-Guided Sensing Aerial Vehicle. IEEE Transactions on Control Systems Technology. 2013; 22(3):853–867. https://doi.org/10.1109/TCST.2013.2267623

40. Pai DK, Reissell LM. Multiresolution Rough Terrain Motion Planning. IEEE Transactions on Robotics and Automation. 1998; 14(1):19–33. https://doi.org/10.1109/70.660835

41. Elfes A. Using Occupancy Grids for Mobile Robot Perception and Navigation. Computer. 1989; 22(6):46–57. https://doi.org/10.1109/2.30720

42. Cowlagi RV, Tsiotras P. Multi-resolution Motion Planning for Autonomous Agents via Wavelet-based Cell Decompositions. IEEE Transactions on Systems, Man and Cybernetics: Part B—Cybernetics. 2012; 42(5):1455–1469. https://doi.org/10.1109/TSMCB.2012.2192268

43. Rao RM, Bopardikar AS. Wavelet Transforms—Introduction to Theory and Applications. Addison-Wesley; 1998.