AIAA SciTech Forum
January 3-7, 2022, San Diego, CA & Virtual
AIAA SCITECH 2022 Forum

10.2514/6.2022-1592

Backtracking Hybrid A^* for Resource Constrained Path Planning

Bryce Ford *, Rachit Aggarwal † and Mrinal Kumar ‡
Mechanical and Aerospace Engineering, The Ohio State University, Columbus OH 43210

Satyanarayana Gupta Manyam§
Infoscitex Corporation, Dayton OH 45431-1672

David Casbeer ¶ and David Grymin‖
Wright-Patterson Air Force Base, Dayton OH 45433

This paper considers resource constrained path planning for a Dubins agent. Resource constraints are modeled as path integrals that exert a path-dependent load on the agent that must not exceed an upper bound. A backtracking mechanism is proposed for the Hybrid-A* graph search algorithm to determine the minimum time path in the presence of the path loading constraint. The new approach is built on the premise that inadmissibility of a node on the graph must depend on the loading accumulated along the path taken to arrive at its location. Conventional hybrid-A^* does not account for this fact, causing it to become suboptimal or even infeasible in the presence of resource constraints. The new approach helps "reset" the graph search by backing away from a node when the loading constraint is exceeded, and redirecting the search to explore alternate routes to arrive at the same location, while keeping the path load under its stipulated threshold. Backtracking Stopping criterion is based on relaxation of the path load along the search path. Case studies are presented and numerical comparisons are made with the Lagrange relaxation method to solving equivalent resource-constrained shortest path problems.

I. Introduction

This paper considers minimum time path planning for a Dubins agent. In addition to the usual kinematic constraints and static obstacles on the search map, additional *integral constraints*, also known as *resource constraints*, or *path loading constraints* are included. As an example, consider an unmanned aerial vehicle (UAV) flying over an active wildfire, as shown in Fig. (1). The scenario on the left depicts path planning with a point-wise constraint imposed in terms of avoiding a critically high heat flux contour. This is similar to a no-fly-zone (NFZ) constraint, where the NFZ boundary is defined in terms of the shown heat-flux contour (e.g. set at 7 kW/m^2). The image on the right shows the same path-planning goal but with a path-loading constraint, in which there are no no-fly-zones (NFZs). However, the UAV platform must maintain its temperature under a set threshold, determined by safety specifications of equipment onboard. Clearly, the UAV is free to fly across all heat-flux contours so long as the temperature onboard, given by the path integral of the heat flux, is maintained under the set threshold at all times. Of course, it is possible to pose a combination of point-wise and integral constraints in a general path-planning problem as the one shown in Fig. (1) (right).

Path loading constraints have been studied under the framework of resource-constrained shortest path problems (RCSPP) in the existing literature, e.g. see Ref. [1]. The shortest path problem with integral constraints is also known as the Constraint Shortest Path (CSP) problem. When the speed of traversal is a constant, as for a Dubins agent, this translates to a minimum time problem. CSP problems are commonly found in operations research and network routing

*Undergraduate Student, AIAA Student Member
†Ph.D. Student, AIAA Student Member
‡Associate Professor, AIAA Associate Fellow
§Research Scientist, 4027 Colonel Glenn Highway, Suite 210, AIAA Member
¶Technical Area Lead for UAV Cooperative and Intelligent Control, Control Science Center of Excellence, Air Vehicles Directorate, AIAA Associate Fellow
‖Controls Science Engineer, Control Science Center of Excellence, Air Vehicles Directorate, Air Force Research Laboratory, Wright-Patterson Air Force Base. E-mail: david.grymin.1@us.af.mil

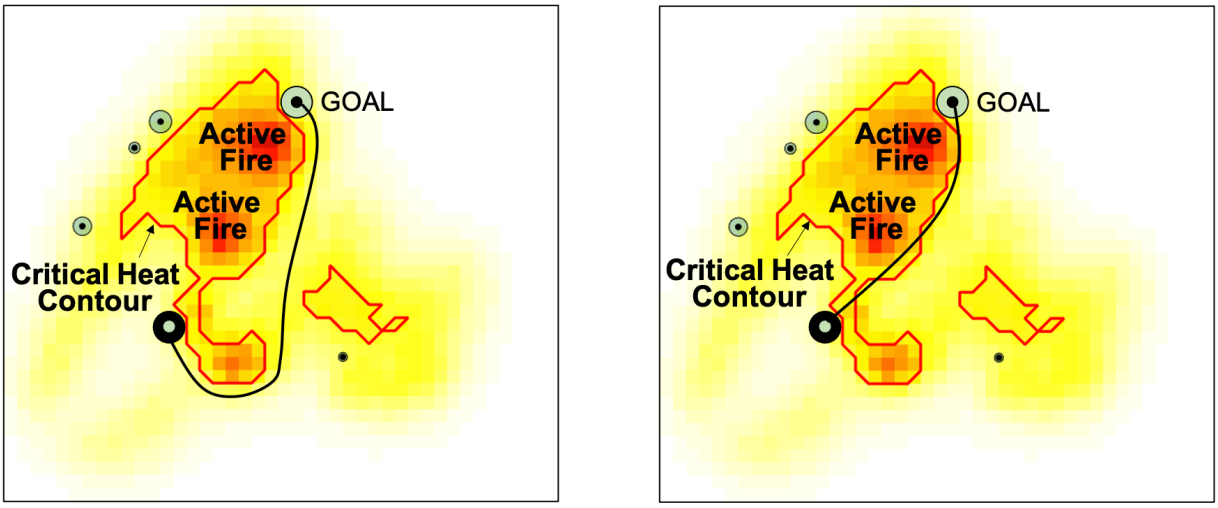


Fig. 1 Left: Point-wise Constraint Enforced in terms of Avoiding High (Critical) Heat Flux Contour. Right: Integral Path Loading Constraint Enforced in terms of Maintaining Platform Temperature Under Set Threshold (Temperature = Path Integral of Heat Flux)

[2, 3]. The presence of the resource constraint makes the problem generally unsuitable for graph search methods like the Dijkstra's algorithm or the A^* algorithm. Although it can be solved using mixed integer linear programming (MILP), the computational load is generally prohibitive. Alternatively, the Lagrange Relaxation approach [2, 3] can be employed to transform the problem into a dual problem that leads to a relaxation of the constraint coupling. This yields a problem that is simpler to solve but results in an approximate solution. Further, it can be iteratively solved using existing unconstrained discrete optimization techniques such as Dijkstra's algorithm. The method described in this work is compared against the Lagrange Relaxation based Aggregate Cost (LARAC) algorithm described in Refs. [4, 5]. It uses Dijkstra's algorithm in an iterative manner by modifying the edge weights until the constraint is met. Due to the iterative nature, henceforth, it is referred to as iterative Dijkstra or iDijkstra algorithm.

Graph search algorithms are a popular choice for path-planning problems without resource constraints because they can effectively deal with complex obstacles that create non-convex search spaces. As mentioned above, popular grid-based algorithms include Djikstra's (uniform) search [6], A^* (directitonal) search [7], θ^* (grid-detached) search [8] and their numerous variants. A common shortcoming of the techniques mentioned so far is that they do not conform to the vehicle's dynamic constraints, due to which the resulting "optimal" trajectory may not be feasible. Recently, the hybrid-A^* ($\mathscr{H}A^*$) approach has gained popularity [9, 10] for kinematic agents with simple motion primitives, such as the Dubins model. In hybrid-A^*, known motion primitives of the vehicle's kinematic model are employed to generate candidate nodes for exploration of the search space. A significant difference is that the $\mathscr{H}A^*$ search space includes position *and* heading of the vehicle. In other words, the search occurs over the *pose-space* of the vehicle, i.e. (x, y, ψ), causing dimensionality to increase by one for the traditional planar planning problem. While the hybrid-A^* search does not evolve on a grid, a "companion" grid is employed in the background to help control the growth in size of the search frontier. The companion grid partitions the search space into a manageable number of discrete cells in the pose space. Presence of a "visited" (aka *closed*) node in a given cell marks the cell visited (*closed*) as a whole, thereby preventing additional candidate nodes being added in the cell. This approach helps control the computational burden associated with $\mathscr{H}A^*$, while generating smoother optimal paths that are suitable for kinematically constrained vehicles, such as a fixed wing UAV.

This paper presents a *backtracking hybrid-A^* graph search* for path-planning with resource (path loading) constraints. As previously mentioned, traditional graph search is not suited to planning with path loading constraints because such constraints accumulate over the trajectory of the agent. When a candidate node is determined to be inadmissible on account of the loading constraint, conventional graph search effectively treats it as the violation of a point-wise constraint (Fig. (1): left) as opposed to an integral constraint (Fig.(1): Right). The key idea behind the new approach is that the inadmissibility of a candidate node is on account of *the path taken to arrive at its location*. Therefore, instead of classifying the candidate node as "inadmissible", a better alternative is to retreat the search, receding away from the candidate node, thereby allowing the agent to shed the integral load along the path leading up to the constraint violation.

Upon the conclusion of the backtracking step, $\mathscr{H}A^*$ search resumes until another load violation is encountered (or the goal is reached). Of course, the stopping criteria for backtracking is important in this procedure. To this end, an auxiliary optimization problem is solved on the entire graph first, in which *the integral constraint is defined as the cost function*. This can be done using a uniform graph search method like the Djikstra's algorithm. The result of this graph search is the minimum load possible at each node. Then, backtracking stops at the node where the actual path-load encountered is less than a relaxation factor times the minimal load possible at the node. The results of the proposed backtracking $\mathscr{H}A^*$ algorithm are shown using numerical results and flight tests. Optimality gains over traditional $\mathscr{H}A^*$ are clearly demonstrated. Numerical comparisons are also shown with the Lagrange Relaxation based Aggregate Cost (LARAC) algorithm.

II. Problem Statement

A. Vehicle Dynamics

This work employs the Dubins kinematic motion model for agent dynamics [11]

$$\dot{x} = v\cos\psi \tag{1a}$$
$$\dot{y} = v\sin\psi \tag{1b}$$
$$\dot{\psi} = u \tag{1c}$$

Eq. (1) captures motion in two dimensions (x, y), with heading angle ψ. The agent's speed, v, is assumed to be constant. The control input is u, which commands the heading rate. An upper bound is imposed on the maximum heading rate on account of the minimum allowable turning radius of the vehicle (aka curvature constraint, aka maximum steering angle constraint), given as follows

$$|u| \leq U \tag{2}$$

The Dubins kinematic model is commonly used in path-planning for ground vehicles with turn-rate constraints [11]. It is also a popular choice for fixed altitude trajectory planning for unmanned aerial vehicles. It offers a rapid means to path planning due to the existence of analytical optimal (shortest path) solutions between a given pair of admissible (unobstructed) waypoints [11]. Two notable extensions of the Dubins model have been proposed in the literature:

1) Reeds-Shepp paths [12], which include reversing motion (i.e. $\dot{\psi} = \pm u$). This model is applicable to ground vehicle motion planning, especially slow moving vehicles in tight spaces such as a parking lot.
2) "G^2" and "G^3" paths (Refs. [13, 14]), which are smoother variants of the Dubins path. The Dubins path's curvature profile is discontinuous (a "G^1" path), which translates to an uncomfortable ride at best and an undeliverable control actuation at worst. Path feasibility is improved by adding steering rate or steering acceleration constraints, which correspond to G^2 or G^3 trajectories, respectively. The G^3 option results a continuously differentiable steering profile.

This work uses the original Dubins motion primitives without reversing motion. Consider an agent with minimum turn radius R traveling at constant speed v, such that $R = v/U$. A constant time discretization of ΔT is used as the time step for motion planning, resulting in motion primitives of constant and equal arc lengths. The Dubins model admits only three unique motion primitives, which are enumerated below:

1) Heading straight (S): the agent continues without a change in heading, resulting in "forward motion" covering a distance of $v\Delta T$,
2) Turn left at minimum turn radius (L): the agent turns left at maximum steering rate $\dot{\psi} = U$, resulting in a circular arc of length $v\Delta T$, and,
3) Turn right at minimum turn radius (R): the agent turns right at maximum steering rate $\dot{\psi} = -U$, resulting in a circular arc of length $v\Delta T$.

The hybrid-A^* graph search approach adopted in this paper employs the above three actions to generate new candidate nodes. Clearly, the nodes generate by these actions do not conform to a grid-like structure. Therefore, a companion grid is employed to discretize the search domain and the resulting cells are used to determine visitation of the search procedure within the discretized space. When a node is created and marked as "visited" inside a given cell, the cell is marked as "visited" in return. All future nodes created in the same cell are discarded. This process is described further in Sec.(III).

B. Cost and Constraint Model

Recall that the agent is assumed to travel at constant speed. Consequently, the planning objective for the agent is to achieve a desired terminal pose (position and heading) in *minimum time*. The agent is subject to point-wise no-fly-zone constraints, as well as an integral (path loading/resource) constraint. The optimization problem can be posed as follows

$$J^*(t, \mathbf{s}, u) \;=\; \min_u t_f \tag{3}$$

where, $\mathbf{s} \;=\; (x, y, \psi)$ is the vector state. The agent must follow the following dynamic constraints given in Eqs. (1) along with the upper bound on the maximum turning rate. The following no-fly-zones constraints are known, each modeled as a polygonal keep-out zone:

$$\bigvee_{j=1}^{M_i} a_{i,j}x + b_{i,j}y > c_{i,j}, \quad i = 1, \dots N \tag{4}$$

where, $(a_{i,j}, b_{i,j}, c_{i,j})$ are parameters of the j^{th} vector of the i^{th} polygonal comprising of M_i vertices. The integral, path-loading constraint is given as

$$\underbrace{\text{“}\delta(t) = 1\text{”}}_{\text{“damage”}} \;\equiv\; \underbrace{\int_0^t \underbrace{\mathcal{F}_{\mathcal{L}}(\tau, \mathbf{s}(\tau), u(\tau))}_{\text{rate of damage}}\, d\tau}_{\text{path load}} \;\leq\; \underbrace{\mathcal{L}^\star}_{\text{loading limit}} \tag{5}$$

In the above equation, the the rate of damage to the platform is given by the function $\mathcal{F}_{\mathcal{L}}(\cdot, \cdot, \cdot)$ and is context dependent. When considering the example of flight-over-fire, it represents a normalized heat flux as follows

$$\mathcal{F}_{\mathcal{L}}(t, \mathbf{s}(t), \mathbf{u}(t)) \;=\; \frac{A}{mc_p}\phi(\mathbf{s}(t)) \tag{6}$$

the variables above have the usual meaning (m = vehicle mass, A = incident area, c_p = heat capacity, $\phi(\cdot)$ = heat flux). Heat flux *can be* allowed to be negative (cooling effect), allowing the planner to perform load-shedding by interspacing flight through hot regions with flight over cooler areas, thus keeping the total rise in temperature under loading limits:

$$\dot{\phi}(x(t), y(t)) \;=\; \begin{cases} \dot{\phi}_{\text{rad}}(x(t), y(t)) = \dot{q}'', & \text{if } \dot{\phi}_{\text{rad}}(x(t), y(t)) > 0 \\ \dot{\phi}_{\text{cool}}(x(t), y(t)) = -h(T - T_{\text{min}}), & \text{otherwise} \end{cases} \tag{7}$$

This paper only considers non-negative loading constraints such that loading relief is not possible. In other words, $\dot{\phi}_{\text{cool}}(x(t), y(t)) \;=\; 0$. While convection and radiation are the primary modes of heat transfer from the wildfire, radiation becomes the dominant form of heat transfer for surfaces exceeding $400°C$ [15]. As the flame temperatures of a wildfire can vary between $800°C$ and $1000°C$, the fire's radiative heat that extends up into the atmosphere is the primary concern as it affects the safety the UAV. Each burning location is treated as a radiative surface that emanates energy per unit area at a rate according to the Stefan-Boltzmann law:

$$E = \sigma_s \epsilon T_f^4 \tag{8}$$

where T_f is the absolute temperature of the flame in K, $\epsilon \in [0, 1]$ is the emissivity and $\sigma_s = 5.67 \times 10^{-8} W/m^2K^4$ is the Stefan-Boltzmann constant. This rate of heat energy loss is referred to as the *heat flux* radiates in all directions from the flame surface A_1. The radiant heat flux that is incident upon a small element of a secondary surface A_2 due to the flame's flux is given as:

$$\dot{q}'' = E\phi \tag{9}$$

where ϕ, known as the configuration factor - a dimensionless quantity that describes the geometric relation (e.g. distance and orientation) between surfaces A_1 and A_2. The heat flux from the multiple radiating surfaces are added to obtain the net heat flux. In the field of fire safety, the incident heat flux $\dot{q}''$ is often used as the metric to indicate the level of danger. Firefighters can generally tolerate a maximum incident heat flux of 7 kWm^{-2} [16] and wood ignites within a few seconds of exposure to heat in excess of 20 kWm^{-2} [15]. Moreover, the high temperatures can also cause updrafts

which can lead to flight instability.
Finally, we have the following boundary conditions (including given initial pose and desired final pose):

$$\begin{aligned} \text{Terminal Conditions:} \quad \{x(0), y(0), \psi(0)\} &= \{x_0, y_0, \psi_0\} \\ \{x(t_f), y(t_f), \psi(t_f)\} &= \{x_f, y_f, \psi_f\} \end{aligned} \tag{10a}$$

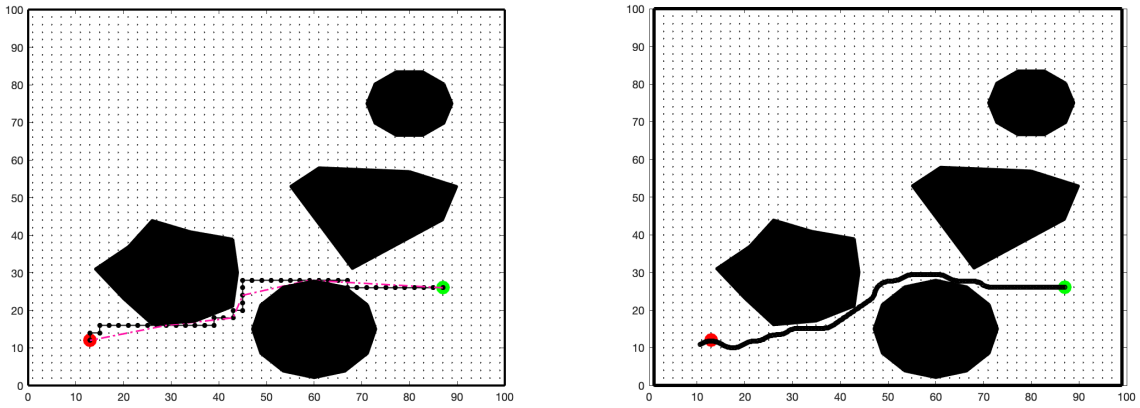
$$\begin{aligned} \text{State Bounds:} \quad x_{\min} &\leq x(t) \leq x_{\max} \\ y_{\min} &\leq y(t) \leq y_{\max} \end{aligned} \tag{10b}$$

III. Solution Methodology

A. Hybrid A^*

The hybrid A^* algorithm was developed as a means to "detach" the graph search from the grid. This is especially relevant to robotic agents with non-holonomic constraints, e.g. fixed wing unmanned air vehicles, that are unable to change their heading in a non-smooth manner. Fig.(2) illustrates the difference between optimal paths generated by the A^* search and the hybrid-A^* ($\mathscr{H}A^*$) search. Fig.(2(a)) shows the case in which four actions are considered in the A^* search (Up, Down, Left, Right). The A^* solution is bound to the graph, resulting in a non-smooth, "node-hopping" trajectory. Note however that the jagged nature of the A^* solution can be reduced through post-processing steps such as connecting the farthest nodes in the trajectory that are intervisible. This is shown using the dash-dot pink line in Fig.(2(a)). Not only just this process *straighten out* the A^* solution, it also further reduces the path length of the trajectory to be traversed.

On the other hand, the $\mathscr{H}A^*$ algorithm builds its graph using the motion primitives described in Sec.(II.A). Also, the search space now contains three variables, namely, two components of position (x, y) and one component of heading (ψ) for path-planning in two-dimensional space, e.g., ground robots or constant altitude flight. It does use a companion grid, $\mathscr{D}$, in the three dimensional search space, that functions in the background to determine visitation of various regions of the three dimensional search space. The view shown in Fig.(2(b)) only illustrates a two dimensional cross section of the companion grid, $\mathscr{D}$. Clearly, $\mathscr{H}A^*$ is still a discrete graph search, in the sense that the trajectory is constructed by patching together finite-sized motion primitives extracted from a motion model. The time-step is fixed, such that



(a) Optimal Trajectory Generated by A^* Graph Search

(b) Optimal Trajectory Generated by $\mathscr{H}A^*$ Graph Search

Fig. 2 Smoothness of A-Star Versus Hybrid A-Star.

each candidate action (S, L, R) has the same path length: see Fig.(3) for an illustration. A zoomed-in view is shown, with the companion grid, $\mathscr{D}$, shown using empty circles. Note that only the $x - y$ cross section of the grid $\mathscr{D}$ is visible. Discretization of the third variable, i.e. heading angle, is illustrated using round dials the represent a discrete set of heading angles. A good rule of thumb for the design of the companion grid is to use the vehicle's kinematic capabilities,

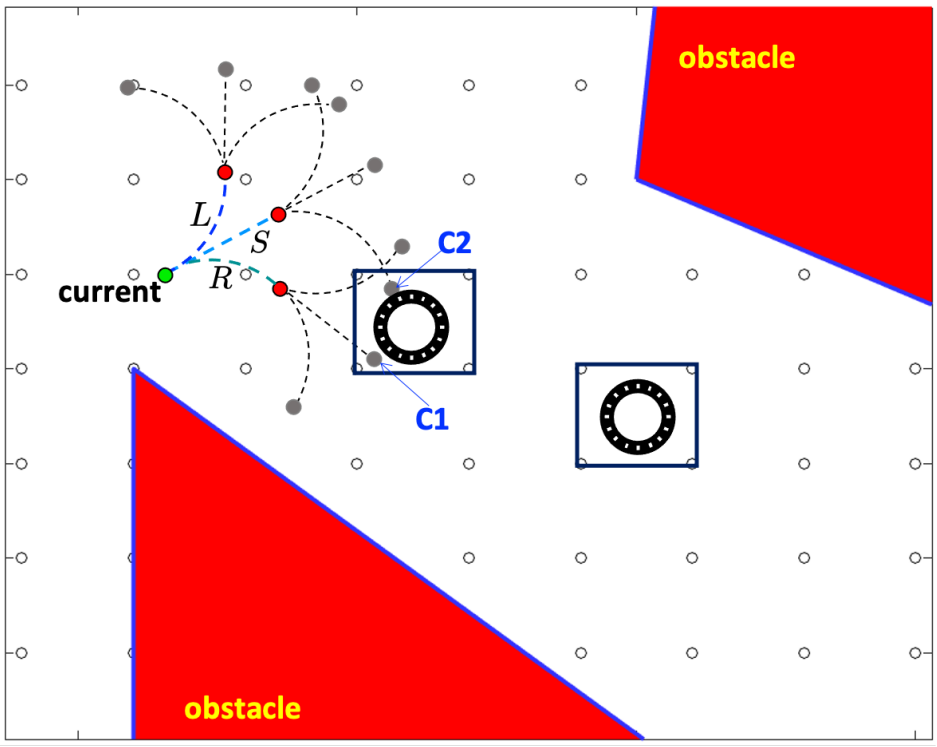


Fig. 3 The $\mathscr{H}A^*$ algorithm: an illustration of the companion grid, $\mathscr{D}$, and the generation of candidate nodes using Dubins motion primitives

e.g. $\delta x \approx v\Delta T$, $\delta y \approx v\Delta T$ and $\delta\psi \approx v\Delta T/R$, such that

$$\mathscr{D} = \{x_i\}_{i=1}^{N_x} \otimes \{y_i\}_{i=1}^{N_y} \otimes \{\psi_i\}_{i=1}^{N_\psi} \tag{11}$$

where, $N_x = \left\lceil [(x_{\max} - x_{\min})/\delta x] \right\rceil$, and N_y and N_ψ are similarly defined. Candidate paths (S, L, R) emerging from the *current* location of the graph search (green circle) are shown, resulting in three new *candidate* nodes (red circles). Clearly, the candidate nodes do not coincide with the nodes on the companion grid, $\mathscr{D}$. The purpose of the companion grid is to restrain the growth of candidate nodes. Unhindered kinematic exploration through the use of motion primitives will invariably cause an explosive growth in the number of candidate nodes, e.g. the filled gray circles in Fig.(3) show the second level of candidate nodes generated starting from the current node (green). The home cell of each candidate node in the companion grid is identified using the position *and* heading information of the node. If this node is marked as "visited" (or "closed") during the graph search, its home cell is simultaneously marked as "visited" or "closed". All future candidate nodes generated in this cell are automatically discarded, thereby restricting the computational burden of $\mathscr{H}A^*$. Some important points to note about this process:

- Cells in $\mathscr{D}$ are three dimensional, characterized by x, y and ψ information. Fig.(3) shows nodes in $\mathscr{D}$ (empty black circles) in the $x-y$ cross section. The ψ-dimension is illustrated using a dial for highlighted two $x-y$ cells.
- Consider the two gray candidate nodes marked C1 and C2. While they both fall in the same $x-y$ cell, they have very different heading coordinates and are in different cells in $\mathscr{D}$. Therefore, the "closeness" of candidate nodes must account for all three dimensions.
- As previously mentioned, when a candidate node is marked visited (moved to the closed set), so is its home cell and no further candidate nodes in this cell are considered.

The rest of the conventional $\mathscr{H}A^*$ method is outlined in in Algorithm (1). The following nomenclature is employed in this algorithm:

- O: Starting point for the graph search (initial conditions), such that $O.x = x_0$, $O.y = y_0$ and $O.\psi = \psi_0$.
- G: Goal (destination) of path-planning, such that $G.x = x_f$, $G.y = y_f$ and $G.\psi = \psi_f$.
- DOM: domain of solution

- OBS: set of obstacles
- DAMAGE: the function $\mathcal{F}_{\mathcal{L}}$ in Eq.(5). This function must be integrated along the traversed path to determine the loading at a given candidate node.
- Front: the set of "open nodes" (i.e. admissible candidate nodes)
- Closed: the set of "closed nodes" (i.e. nodes marked visited)
- Parent: the parent node (e.g. green node is the parent for each of the three red nodes in Fig.(3).

The subroutine GETACTIONS employs Dubins' motion primitives to generate new candidate nodes starting from the *Current* node. PUSH and POP are standard push/pop heap operations. Subroutine CHECK performs collision avoidance checks and subroutine HEURISTICS evaluates the heuristic cost function. The subroutine LOAD determine the path load at arrival for each candidate node. Line 26 of the algorithm indicates that conventional $\mathscr{H}A^*$ treats obstacle avoidance and integral constraints in essentially the same way. If either constraint it violated by a candidate node, it is not considered as part of the admissible search space. While this may appear reasonable, it is the incorrect way

Algorithm 1 $\mathscr{H}A^*$ Search with Integral (Path-Load) Constraints

Require: $O, G, v, R, \Delta T$, DOM, OBS, DAMAGE
1: $\delta x \leftarrow v * \Delta t$, $\delta y \leftarrow v * \Delta t$, $\delta\psi \leftarrow v * \Delta t/R$
2: $Gx \leftarrow \text{DOM}_{x_{\min}}$: $-\delta x$: $O.x$: δx: $\text{DOM}_{x_{\max}}$
3: $Gy \leftarrow \text{DOM}_{y_{\min}}$: $-\delta y$: $O.y$: δy: $\text{DOM}_{y_{\max}}$
4: $G\psi \leftarrow \text{DOM}_{\psi_{\min}}$: $-\delta\psi$: $O.\psi$: $\delta\psi$: $\text{DOM}_{\psi_{\max}}$
5: $\mathscr{D} \leftarrow Gx \otimes Gy \otimes G\psi$
6: Current $\leftarrow O$
7: Front $\leftarrow O$, Closed $\leftarrow$ Empty, Parent $\leftarrow O$
8: Cost $\leftarrow 0$, PathLoad $\leftarrow 0$
9: **while not**(Empty(Front)) **do**
10: **if** Size(Front) > 1 **and** Visited(Front(1)) **then**
11: Front $\leftarrow$ POP(Front)
12: **else**
13: Current $\leftarrow$ Front(1)
14: **end if**
15: **if** Reached(G) **then**
16: End
17: **end if**
18: Front $\leftarrow$ POP(Front)
19: GETACTIONS(Current, v, R, ΔT)
20: **for** k = 1 to NumActions **do**
21: Candidate $\leftarrow$Applyaction(k)
22: Candidate-Adm $\leftarrow$ CHECK(DOM, OBS)
23: Candidate-Cost$\leftarrow$ Cost(Current) + ActionCost(k)
24: Candidate-CostTG$\leftarrow$HHEURISTIC(NewNode, G)
25: Candidate-Load$\leftarrow$LOAD(NewNode, DAMAGE)
26: **if** Candidate-Adm **and** Candidate-Load **then**
27: Front $\leftarrow$ PUSH(Candidate)
28: Closed $\leftarrow$ Closed $\cup$ Front(1)
29: **end if**
30: **end for**
31: **end while**

of dealing with integral constraints. If a candidate node violates obstacle avoidance conditions, the node is certainly inadmissible. However, if it exceeds the path-loading limit, it is not through its own fault: see Fig.(4):Top. The violation of the integral constraint at node N^* is on account of the particular path (sequence of parents) that led to the candidate, i.e. $\sum_{k\in\text{path}} \Delta\mathcal{F}_{\mathcal{L}}(k)\Delta T > \mathcal{L}^*$. It is entirely possible to find an alternate path, albeit at a higher cost that permits the node N^* to be admissible.

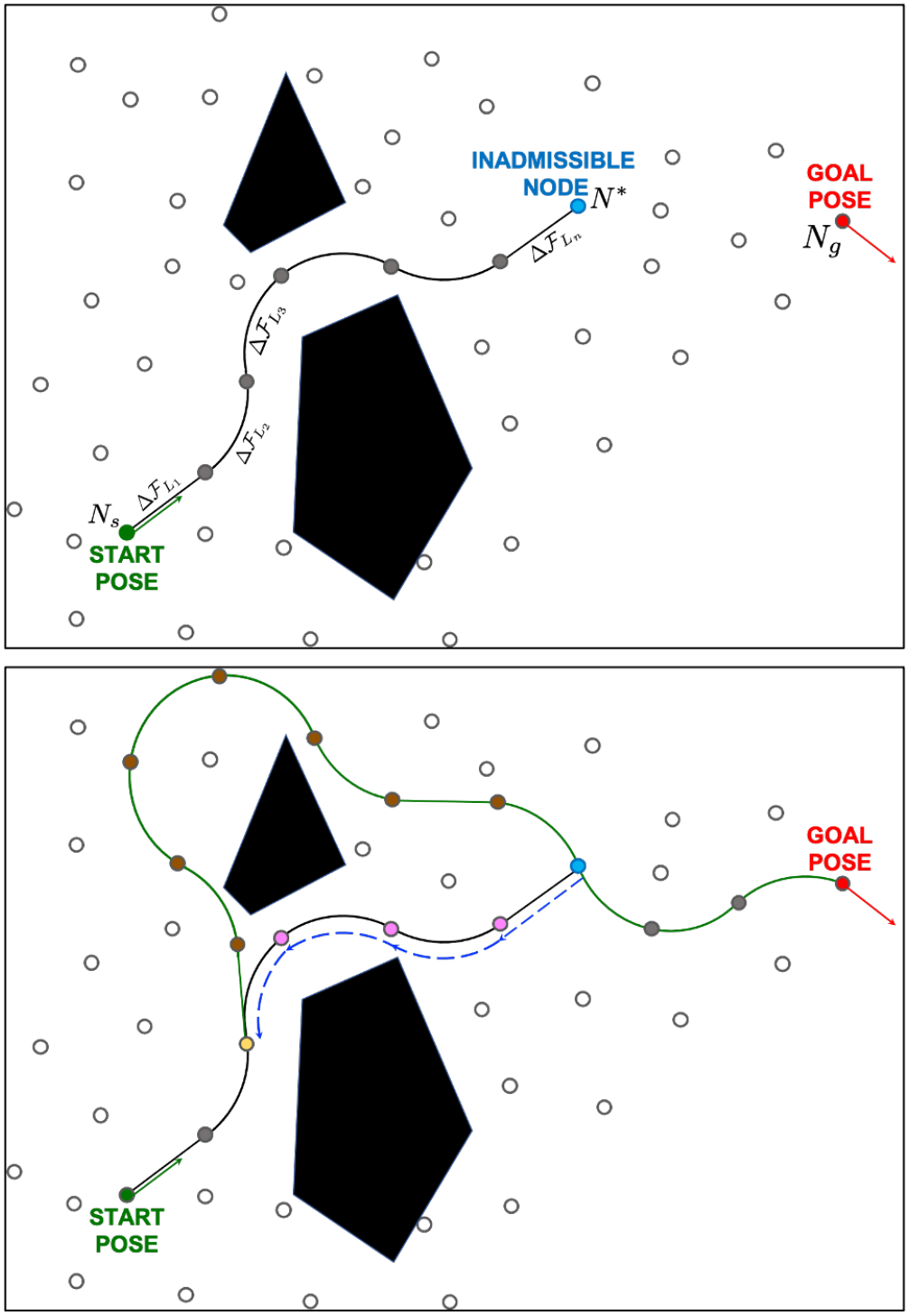


Fig. 4 Top: Accumulation of load along the sequence of parent nodes (filled gray circles) causes inadmissibility of node N^*. Bottom: Backtracking allows the graph search to perform "load-shedding", retreating until a redirection of the search procedure can find an alternate route towards the goal.

B. Backtracking $\mathscr{H}A^*$ ($\mathscr{BH}A^*$)

This paper develops a novel *backtracking* procedure for $\mathscr{H}A^*$ based on the insights outlined in the previous section. It offers a way to adapt a directional graph search method (including other methods like A^* or θ^*) to adapt to problems including path dependent constraints. The approach is based on the following principles:

- Node-inadmissibility due to violation of the integral constraint (Eq.(5)) is path-dependent. In fact, an entire section of the search space can get blocked-off because of such indirect violations. In Fig. (4):Top, the blue node violates the cumulative loading constraint due to the path taken to arrive at it (gray nodes).
- Graph search must backtrack from a node characterized as "inadmissible", with an appropriate stopping condition (Fig.(4): Bottom blue node → yellow node). All nodes shown in the backtracking path (blue, pink and yellow) belong to the visited set (*closed set*) and must be released back into the unexplored search space. In addition, all children (and ensuing posterity) of these nodes, both in the closed and open sets, must also be released.
- Usual directional search resumes (at the yellow node) by picking the second best node, causing a redirection of the graph search.

Backtracking Stopping Criterion

Clearly, the stopping criterion is a crucial element of the backtracking procedure. In the current paper, backtracking is stopped with the aid of an auxiliary optimization problem that seeks to minimize the constraint function, starting from the specified starting pose. This step is completed offline before the $\mathscr{H}A^*$ with backtracking commences. Define the following auxiliary optimization (minimization) problem

$$\mathcal{J}^*(t, u^*, \mathbf{s}) \;=\; \min_{u} \int \mathcal{F}_{\mathcal{L}}(\tau, \mathbf{s}(\tau), u(\tau)) d\tau \tag{12}$$

The constraints for the above minimization problem include the kinematics given in Eq.(1), initial conditions given in Eq.(10a) and domain bounds in Eq.(10b). There are no terminal conditions because the objective is to determine the minimum constraint value within each cell in $\mathscr{D}$. In other words, the above problem (Eq.(12)) is to be solved in a "hybrid-Djikstra" framework, in the sense that:

1) Dubins motion primitives are employed for candidate node generation, and,
2) but no heuristic cost is used to ensure a uniform search in the (x, y, ψ) domain.

The outcome of the hybrid-Djikstra search is a tabularization of the minimum possible path-load value inside each cell of the companion grid, $\mathscr{D}$. This information can now be used to build a stopping criterion for $\mathscr{BH}A^*$. Consider a relaxation parameter $\xi > 1$ (strictly greater than). Then, backtracking stops at the first node M^* for which

$$\int_0^t \mathcal{F}_{\mathcal{L}}(\tau, \mathbf{s}(\tau), u(\tau)) d\tau \;\leq\; \xi \mathcal{J}^*(\cdot, \cdot, \mathscr{C}(M^*)) \tag{13}$$

where, $\mathscr{C}(M^*)$ is the cell in $\mathscr{D}$ containing M^*. In words, the backtracking process stops at a node M^* (depicted as the yellow node in Fig.(4): Bottom) where the actual path-load encountered while minimizing the cost given in Eq.(3) is less than or equal to ξ times the minimum possible path-load possible at that node. This allows for a reasonable means for load-shedding, in the sense that we stop within a factor ξ of the minimum possible integral load, before re-commencing the hybrid-A^* search procedure towards the desired goal. The full $\mathscr{BH}A^*$ algorithm in given in Algorithm (2). In this algorithm, in addition to the variables defined in Algorithm (1), the following additional terms are introduced:

- BT-Closed: sequence of parents (elements of the *closed set*) encountered while retreating from the candidate node that do not satisfy the stopping criterion in Eq.(13).
- BT-Front: The set of all children of each element in the set BT-Closed, including children in the *Closed* and *Open* sets.

IV. Results

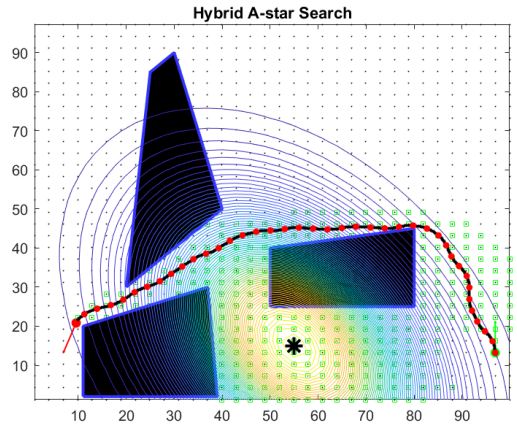
A. Numerical Studies

We present two examples to demonstrate the results of $\mathscr{BH}A^*$ compared to $\mathscr{H}A^*$. All tests were performed on a system with an I7-6700k at 3.99 GHz and 16 GB RAM. For the first case, in Fig. (5(b)) the start and goal poses (O = [96.89 m, 13.23 m, 1.57 rad], G = [9.56 m, 20.84 m, 4.37 rad]) are represented with green and red arrows respectively. The agent has an assumed constant speed of 3 m/s and a turn radius of 8 m. The auxiliary grid is constructed with

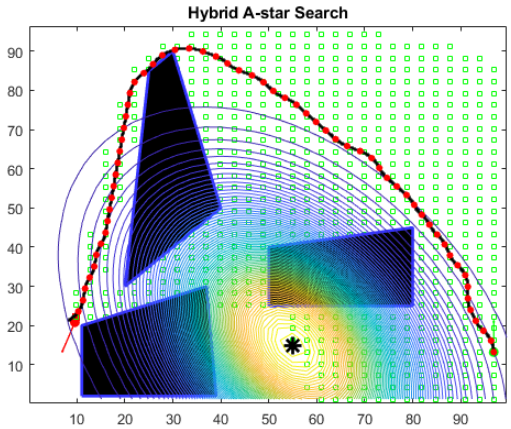
Algorithm 2 Backtracking Hybrid A^* ($\mathscr{BH}A^*$) Search with Integral (Path-Load) Constraints

Require: $O, G, v, R, \Delta T$, DOM, OBS, DAMAGE

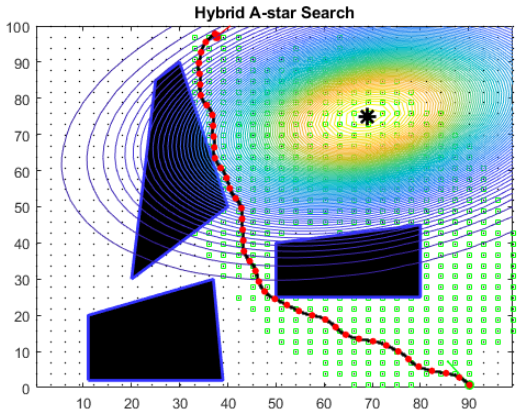
$\delta x \leftarrow v * \Delta t$, $\delta y \leftarrow v * \Delta t$, $\delta\psi \leftarrow v * \Delta t / R$
$Gx \leftarrow \text{DOM}_{x_{\min}}\colon -\delta x\colon O.x : \delta x\colon \text{DOM}_{x_{\max}}$
$Gy \leftarrow \text{DOM}_{y_{\min}}\colon -\delta y\colon O.y : \delta y\colon \text{DOM}_{y_{\max}}$
$G\psi \leftarrow \text{DOM}_{\psi_{\min}}\colon -\delta\psi\colon O.\psi : \delta\psi\colon \text{DOM}_{\psi_{\max}}$
$\mathscr{D} \leftarrow Gx \otimes Gy \otimes G\psi$
Current $\leftarrow O$
Front $\leftarrow O$, Closed $\leftarrow$ Empty, Parent $\leftarrow O$
Cost $\leftarrow 0$, PathLoad $\leftarrow 0$
while not(Empty(Front)) **do**
    if Size(Front) > 1 **and** Visited(Front(1)) **then**
        Front $\leftarrow$ POP(Front)
    else
        Current $\leftarrow$ Front(1)
    end if
    if Current-Load **then**
        BT-Closed $\leftarrow$ {Parent sequence, starting from Candidate | Eq.(13) is not true}
        NumBTClosed $\leftarrow card$(BT-Closed)
        BT-Open = {}
        for q = 1 to NumBTClosed **do**
            BT-Current $\leftarrow$ BT-Closed(q)
            BT-Open = BT-Open $\cup$ Children(BT-Current)
        end for
        Front $\leftarrow$ Front\BT-Open
        Closed $\leftarrow$ Closed\BT-Closed
    end if
    if Reached(G) **then**
        End
    end if
    Front $\leftarrow$ POP(Front)
    GETACTIONS(Current, $v, R, \Delta T$)
    for k = 1 to NumActions **do**
        Candidate $\leftarrow$Applyaction(k)
        Candidate-Adm $\leftarrow$ CHECK(DOM, OBS)
        Candidate-Cost$\leftarrow$ Cost(Current) + ActionCost(k)
        Candidate-CostTG$\leftarrow$HEURISTIC(NewNode, G)
        Candidate-Load$\leftarrow$LOAD(NewNode, DAMAGE)
        if Candidate-Adm **then**
            if Candidate-Load **then**
                Front $\leftarrow$ PUSH(Candidate)
                Closed $\leftarrow$ Closed $\cup$ Front(1)
            end if



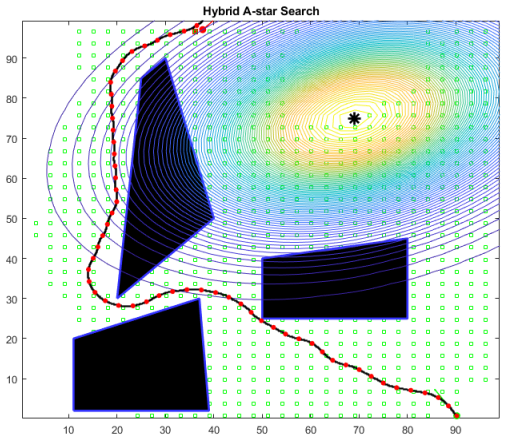
(a) Optimal Trajectory Generated by $\mathscr{H}A^*$ Graph Search With Backtracking



(b) Optimal Trajectory Generated by $\mathscr{H}A^*$ Graph Search Without Backtracking

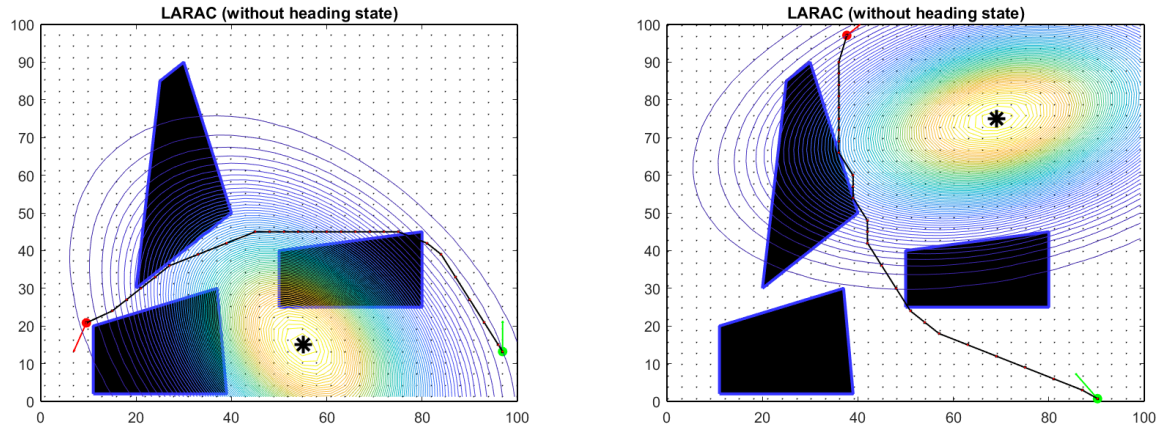


(c) Optimal Trajectory Generated by $\mathscr{H}A^*$ Graph Search With Backtracking, $\mathscr{H}A^*$



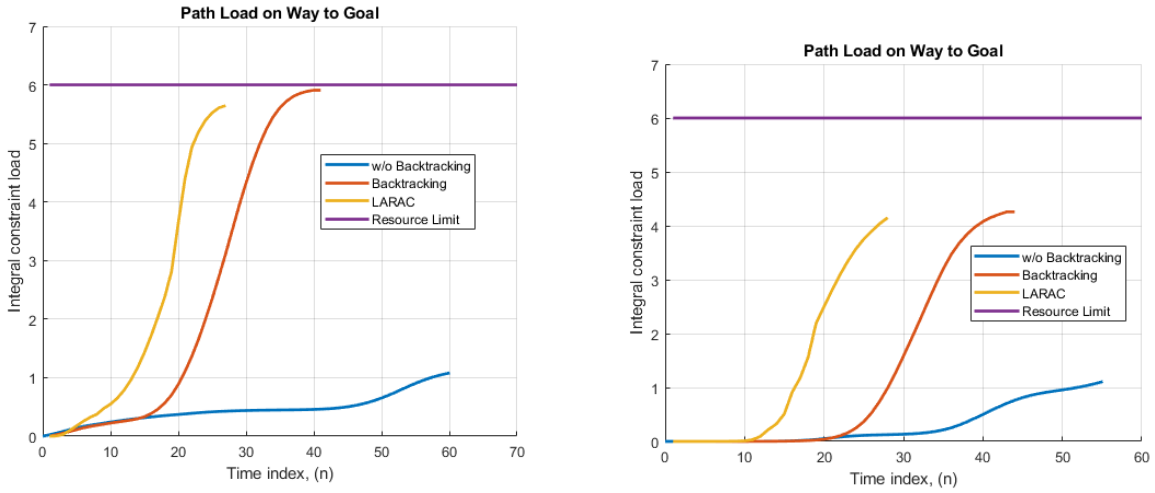
(d) Optimal Trajectory Generated by $\mathscr{H}A^*$ Graph Search Without Backtracking

Fig. 5 Optimal Trajectories Generated With and Without Backtracking. All visited nodes are shown as green circles.



(a) Optimal Trajectory using LARAC algorithm for example Fig. (5(a)) (b) Optimal Trajectory using LARAC algorithm for example Fig. (5(c))

Fig. 6 Optimal Trajectories Generated using LARAC algorithm. The current framework uses 16 nearest-neighbor connectivity grid and agnostic of Dubins motion primitives.



(a) Path-Load Accumulated For Path 5(a), 5(b), and 6(a)

(b) Path-Load Accumulated For Path 5(c), 5(d), and 6(b)

Fig. 7 Path-Load accumulation comparison. Note that the LARAC solutions take fewer steps due to not using Dubins primitives, however the shape of the path is very similar to that of $\mathscr{BH}A^*$

Table 1 A^* Graph Search With Backtracking Performance While Varying Relaxload

ξ	Path Length	Run Time (Seconds)	Number of Times Backtracking is Used	$\sum_{k \in \text{path}} \Delta\mathcal{F}_{\mathcal{L}}(k)\Delta T$
1.1	141	11.40	358	4.182
1.2	147	14.07	580	5.470
1.3	102	2.27	28	5.567
1.4	102	2.54	53	5.567
1.6	177	20.91	1571	3.920
1.8	no solution	29.52	1915	n/a
2.5	165	23.67	1911	5.460
3.0	192	25.52	2091	5.106

$\delta x = \delta y = 3\ m$, and $\delta\psi = v\Delta T/R = 0.375\ rad$. The DAMAGE, $\mathcal{F}_{\mathcal{L}}$ is defined as a multivariate Gaussian distribution (in this case $x - y$) with contours that represent heat flux in a simple fire. The loading limit, $\mathcal{L}^\star$ is set to 6. $\mathcal{L}^\star$ combined with values Σ mean that nodes in the "core" of DAMAGE are inadmissible due to their minimum possible path-load being greater than 6. OBS consists of a set of 3 polygonal obstacles. $\mathscr{H}A^*$ is able to find a path that traverses above the obstacles, avoiding the core of the pdf, thus incurring a cumulative path-load of 1.08 (Fig. (7(a))), and a path length of 180 m. The green circles serve as a 2D representation of the nodes explored in the 3D grid, and as seen in Fig. (5(a)), the majority of the grid is explored. $\mathscr{H}A^*$ converges in an average of 4.92 seconds. With the same obstacles, pdf, start/goal, and grid, $\mathscr{BH}A^*$ is able to find a path that traverses between the obstacles and through the edge of the core of the pdf, incurring a cumulative path-load of 5.91 (Fig. (7(a))), and a path length of 117 m. This is with a ξ coefficient of 1.28. $\mathscr{BH}A^*$ is able to converge in an average of 2.76 seconds while exploring a fraction of the grid, affording the decreased run-time even with the overhead costs of backtracking.

For the second case, in Fig. (5(c)) the problem remains the same as in Fig. (5(a)) with the same OBS, and DOM, but with new O and G poses as well as an alternate DAMAGE, still a Gaussian with an alternate location and orientation. $\mathscr{H}A^*$ was able to find a solution, converging in an average across runs of 4.76 seconds. The solution loops in between the obstacles avoiding the center of the pdf incurring a cumulative load of 1.11 (Fig. (7(b))). The path length is 165 m. $\mathscr{BH}A^*$ is able to converge in an average across runs of 5.04 seconds, finding a path that skirts along a contour of the pdf with a path length of 126 m. The ξ coefficient was set to 1.2. The cumulative path-load is 4.26 (Fig. (7(b))). Here backtracking allows $\mathscr{H}A^*$ to search near to the center of the pdf, finding a shorter path length path. In cases where the path is not trivial, i.e., where there is an instance where a node popped from the frontier reaches the resource limit, backtracking allows $\mathscr{H}A^*$ to converge to a solution, while $\mathscr{H}A^*$ in most cases will find a sub-optimal path or fail to converge at all. If there is not a node popped that exceeds the resource constraint $\mathscr{BH}A^*$ and $\mathscr{H}A^*$ will perform exactly the same.

Fig. (6) shows the paths generated through the LARAC algorithm. The LARAC path generation framework requires prior knowledge of the complete graph with edge integrals which in presence of Dubins motion primitives can result in complications arising from the misalignment with the discretization of the heading state. Therefore, the example presented does not utilize Dubins motion primitives and, instead, uses 16 nearest-neighbor grid configuration. Since this method results in an optimal path [4, 5], it serves as a baseline to verify the results of the $\mathscr{BH}A^*$ approach. For the two examples discussed above, the overall path shape and the path load are compared (see Fig. (6) and (7)) and are found to be very close.

In $\mathscr{BH}A^*$, the relaxation parameter ξ can have a major effect on its efficiency. We present the results of of varying the ξ for an individual problem in Table (1). These tests were performed with the same, DOM, $\mathscr{D}$, OBS, and DAMAGE configuration as test represented in Fig. (5(a)). For this O and G, A^* was unable to converge to a solution. The best result produced had a ξ of 1.3 (Table (1)), producing a path with a length of 102 m and a cumulative path-load of 5.567 (Ref. Fig.), converging in 2.27 seconds. Increasing the ξ results in the same path, but with more backtracks required and thus a higher run-time. Decreasing the ξ results in a longer path with a lower cumulative path load and require far more backtracks and a significantly higher run-time. As seen in Figs. (8(a)) and (8(c)), the solution that a ξ of 1.1 produces results in far more of $\mathscr{D}$ being explored. In Fig. (9) the load incurred over that path for various ξ coefficient values presented in table (1).

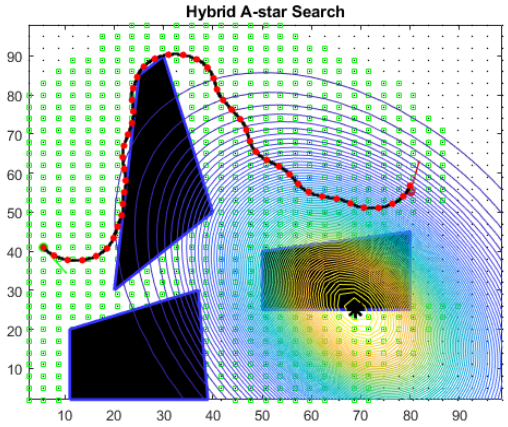
A $\xi = 1$ means that when a node violates an integral constraints, the search will backtrack until it has found a node in the path built from the start to the node to the node in violation that has a path load that is the optimal path-load for that node. What this effect will result in is that the planned path will converge to the minimum load path the more often backtracking is used. Requiring to backtrack to the point where the path load equals the minimum possible load to that point is likely to result in deep searches into the visited tree. The number of nodes removed from the visited (Closed), and frontier (Front) has an upper bound of $O(b^{d+1})$ where b is the branching factor (b = 3 for this case of Dubins primitives) and d is the depth of the backtrack. This means that deep backtracks can get computationally expensive very quickly, particularly considering the data-structure chosen for Front as the cost of removing items from a min-heap is nontrivial. On the other hand a large ξ will result in very little load shedding, and the result of this is shallow backtracks that fail to identify the source of the constraint violation. Backtracking will be used more frequently, resulting in a path that is very close to the resource constraint, but has sacrificed all of its optimality with respect to path length. Every time backtracking is used, optimal path length is being traded for a more optimal path load. This pattern of diminishing marginal returns on ξ is not monotonic, as seen in Table (1). The performance of backtracking is relative to ξ is also very dependant on the structure of $\mathscr{D}$, DAMAGE, and OBS. There will be local minimums in path length or computational cost as ξ increases but they do not translate from problem to problem, for example, 1.3 is best ξ in terms of path length and computation time for the problem presented in Table (1) and Fig. (8), however this ξ does not produce optimal results in other problem configurations such as in the path generated in Fig.(5(c)) where $\xi = 1.2$ produces optimal results. The problem then becomes what is the best possible value for ξ given $\mathscr{D}$ structure that the minimum path cost while ensuring that the resource constraint is not violated. This is ongoing work and will be addressed in the final manuscript.

B. Flight Tests

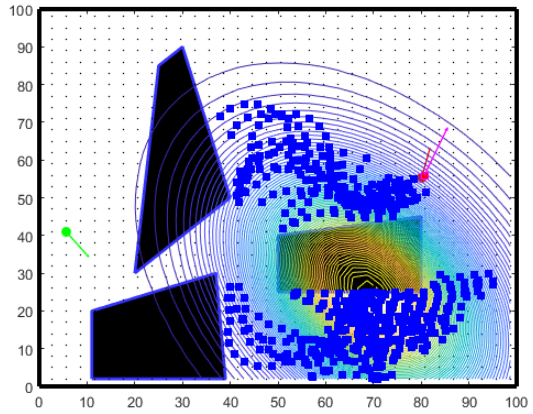
In addition to the numerical studies, flight tests were performed to validate the trackability the computed paths. A multirotor small unmanned aerial system (UAV) platform was used to conduct the tests. Although the agility of multirotor does not limit its trajectories to Dubins like path, the UAV was flown along the computed Dubins path and the test serves as a means to validate the path tracking capabilities and study the deviations in future improvements in path design.

The said multirotor platform, nicknamed 'Hawk', is built using modified DJI F450 quadrotor airframe with DJI 2312E motors (960 rpm/V rating), DJI 430 Lite ESCs (30 A rating) and DJI 9.4×5 propellers. Weighing at about 1.5 kg, it is capable of producing a max of 3200g of thrust and delivers 18-20 minutes flight time with a 4000 mAh 4-cell LiPo battery. Fig.(10) shows the schematic of critical components of the Hawk (Fig.(10(a))) and the assembled vehicle (Fig.(10(b))). Hawk employs the Pixhawk 4 flight controller board [17] hosting the PX4 autopilot firmware [18]. PX4 autopilot uses measurements from the accelerometer, gyroscope, magnetometer, barometer and GPS for an accurate state estimation and uses PID control for position control. Additionally, the UAV is equipped with a downward Lidar sensor to measure distance from ground enabling a soft landing. Using a software MAVLink Router [19] on the ground station computer to generate multiple MAVLink streams [20] from the telemetry module's serial port, the vehicle can communicate with QGroundControl (QGC) [21] on the (a) Ground Station Computer via local UDP connection, and (b) remote pilot's smartphone mounted on the remote control (RC) handset by relaying it via external UDP connection over Wi-Fi (as shown in Fig.(10(a))). PX4's mission mode capability for following series of waypoints was used to track the computed path. For seamless interaction between the UAV and the path planning, a ROS (robot operating system) based mission management was deployed to request path, upload the path to the vehicle and command execution.

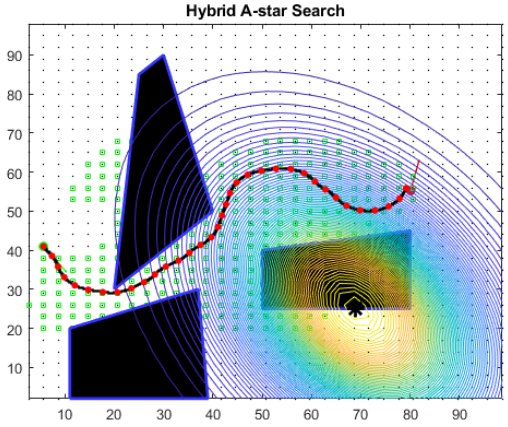
Flight tests were performed for two separate paths, shown in Figs.(5(a)) and (5(c)). The results of the flight tests are displayed in Fig.(11). As seen in the figures, the Hawk was able to follow the planned set of way points with minimal deviation, successfully avoiding any collisions with obstacles. However, the issue is that for the flight path to 'followed', all assumptions made in the planner must be met, the critical assumption being that the Hawk is able to maintain a near constant speed of $v = 3m/s$. The Hawk was unable to do this due to limitations in Pixhawk 4 flight controller's ability to follow sets of way points that are relatively close while maintaining a constant prescribed speed. In the tests the Hawk had an average speed of 1.6 m/s. The result is that when computing the approximate path load of the flown path (using Eq.(5)) the slower speed means that the path load is accumulated much faster than if the Hawk was able to fly as the prescribed speed, meaning that the path load far exceeds the resource constraint.



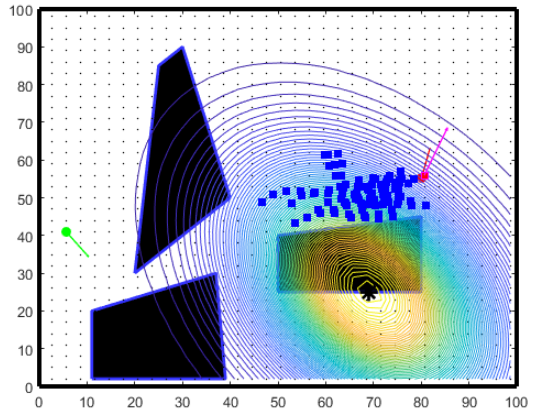
(a) Results of $\mathcal{BH}A^*$ with $\xi = 1.1$



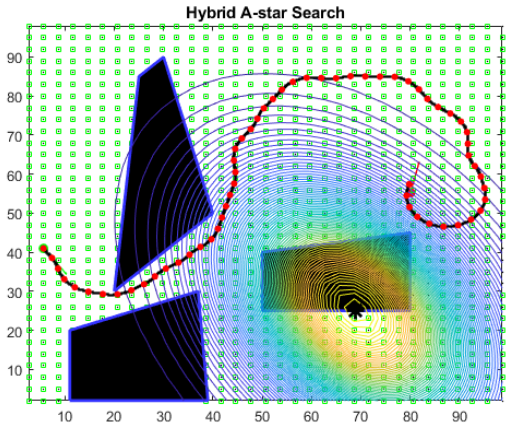
(b) Nodes Removed From Closed During Backtracking with $\xi = 1.1$



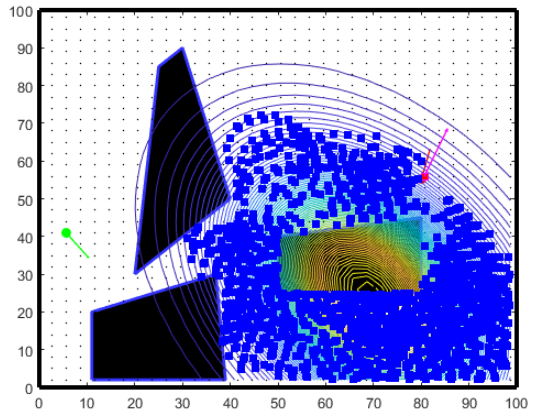
(c) Results of $\mathcal{BH}A^*$ with $\xi = 1.3$



(d) Nodes Removed From Closed During Backtracking with $\xi = 1.3$



(e) Results of $\mathcal{BH}A^*$ with $\xi = 1.6$



(f) Nodes Removed From Closed During Backtracking with $\xi = 1.6$

Fig. 8 Plots of Results of Varying ξ in $\mathcal{BH}A^*$ (Table (1))

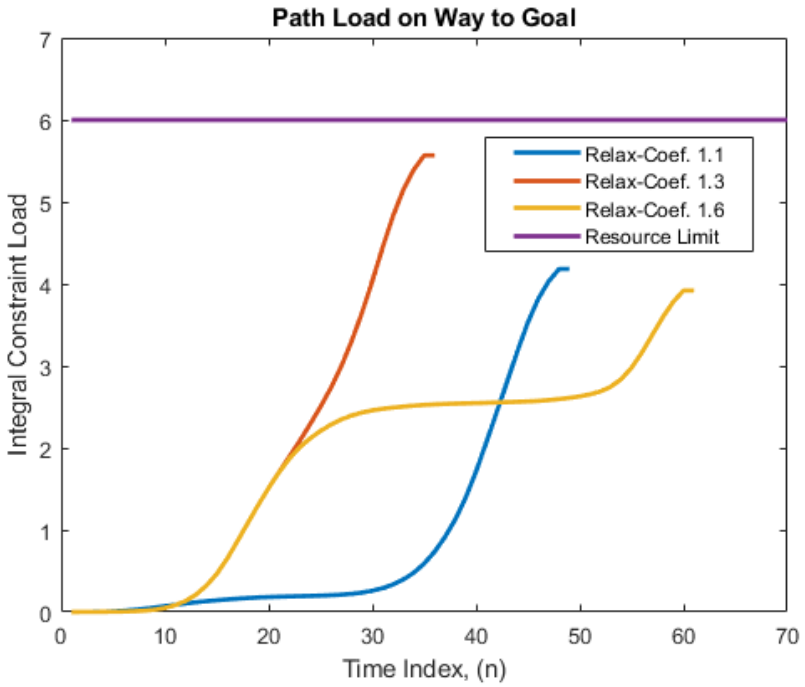


Fig. 9 Path-Load Accumulated by Paths Generated by Using Various ξ For a Set O/G

V. Conclusion

In conclusion, *backtracking hybrid-A* graph search* is able to demonstrate gains in optimality over hybrid A^* graph search. In numerical tests using a model of a simple fire as a damage function $\mathscr{BH}A^*$ is able to produce more optimal paths than $\mathscr{H}A^*$ and that closely resemble paths generated using the Lagrange Relaxation based Aggregate Cost (LARAC) algorithm, which is guaranteed to produce optimal results. Thus paths generated by $\mathscr{BH}A^*$ successfully demonstrate that the load shedding approach of backtracking can produce more optimal results than $\mathscr{H}A^*$ which treats integral constraint violations as point-wise constraint violations. Additionally the added overhead of backtracking is marginal compared to $\mathscr{H}A^*$ if the optimal relaxation parameter is chosen. Flight tests, performed using a multirotor small unmanned aerial system (UAV), demonstrated that paths generated by $\mathscr{BH}A^*$ are easily trackable. Future work includes improving the optimality of the Backtracking Stopping Criterion, as well as modifications to the waypoint following approach for path tracking.

References

[1] Thomas, B. W., Calogiuri, T., and Hewitt, M., "An exact bidirectional A* approach for solving resource-constrained shortest path problems," *Networks*, Vol. 73, 2019, p. 187–205.

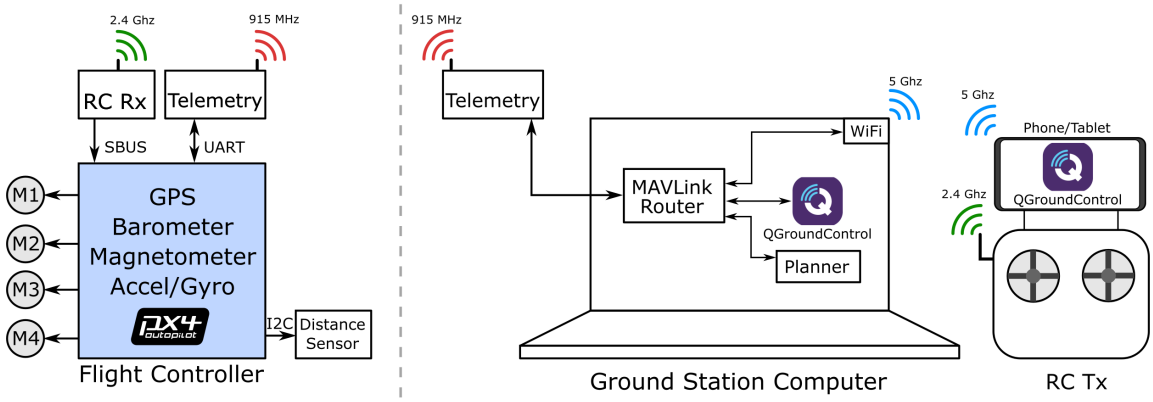
[2] Ahujia, R., Magnanti, T. L., and Orlin, J. B., "Network flows: Theory, algorithms and applications," *New Jersey: Rentice-Hall*, 1993.

[3] Bertsekas, D. P., "Nonlinear Programming," 1999.

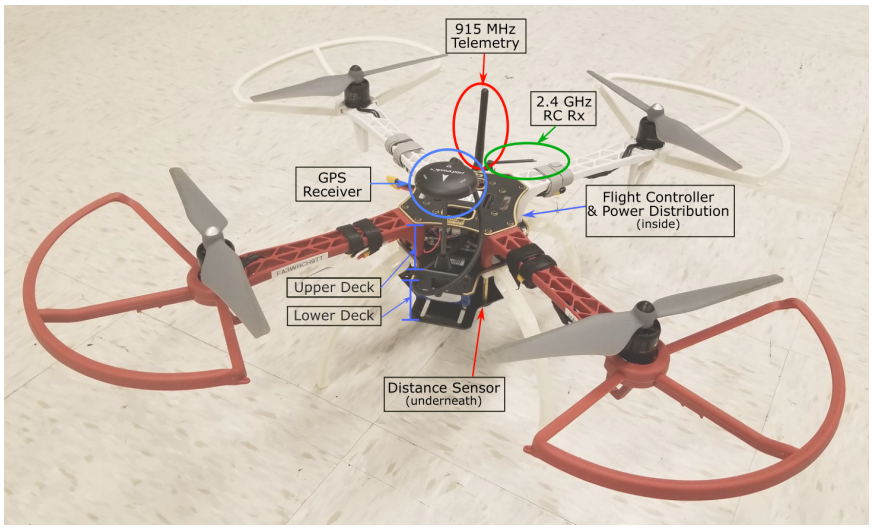
[4] Jüttner, A., Szviatovski, B., Mécs, I., and Rajkó, Z., "Lagrange relaxation based method for the QoS routing problem," *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, Vol. 2, IEEE, 2001, pp. 859–868.

[5] Xiao, Y., Thulasiraman, K., Xue, G., Jüttner, A., and Arumugam, S., "The constrained shortest path problem: algorithmic approaches and an algebraic study with generalization," *AKCE International Journal of Graphs and Combinatorics*, Vol. 2, No. 2, 2005, pp. 63–86.

[6] Dijkstra, E. W., "A note on two problems in connexion with graphs," *Numerische Mathematik*, Vol. 1, 1959, p. 269–271.



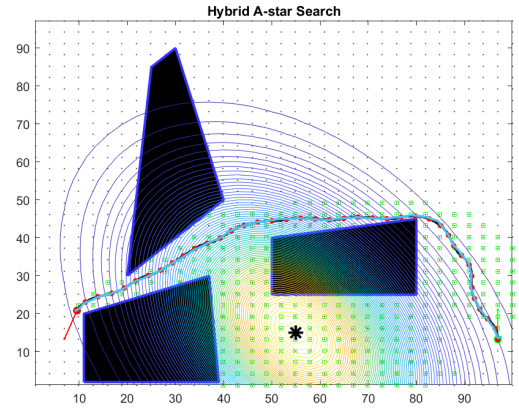
(a) Schematic



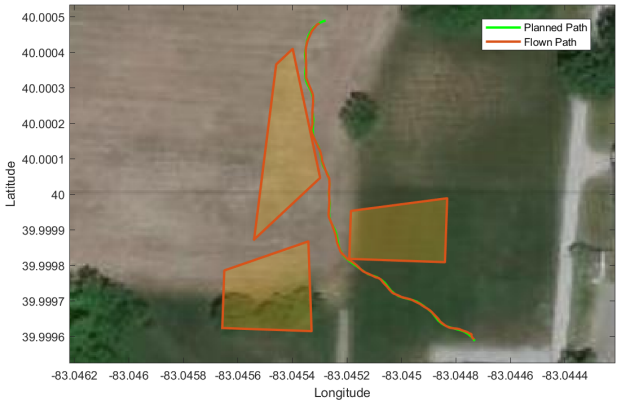
(b) Assembled *Hawk* platform

Fig. 10 *Hawk* UAV built on a modified DJI Flamewheel F450 (450 mm size) X-type Quadrotor platform

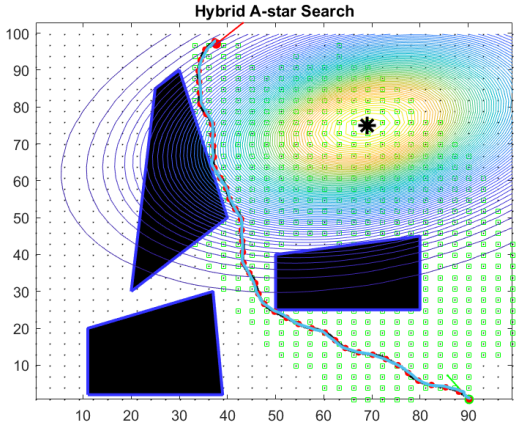
(a) Results of the Flight Test of the Path from (Fig. 5(a)) With Obstacles Superimposed on Local Map



(b) Results of Flight Test Converted From Geodetic to Cartesian and Plotted on 5(a)



(c) Results of the Flight Test of the Path from (Fig. 5(c)) With Obstacles Superimposed on Local Map



(d) Results of Flight Test Converted From Geodetic to Cartesian and Plotted on 5(c)

Fig. 11 Results of Flight Tests Superimposed on Geographic Location and Planner Output

[7] Hart, P., Nilsson, N., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, Vol. SCC-4, No. 2, 1968, p. 100–107.

[8] Daniel, K., Nash, A., Koenig, S., and Felner, A., "Theta*: Any-Angle Path Planning on Grids," *Journal of Artificial Intelligence Research*, Vol. 39, 2010, pp. 533–579.

[9] Dolgov, D., Thrun, S., a, M. M., and Diebel, J., "Path Planning for Autonomous Driving in Unknown Environments," *Experi. Robotics: The 11th Intern. Sympo., STAR 54*, Changshu, Suzhou, China, June 26-30, 2018, p. 55–64.

[10] Petereit, J., Emter, T., Frey, C. W., Kopfstedt, T., and Beutel, A., "Application of Hybrid A* to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor Environments," *Robotik 2012: 7th German Conference on Robotics*, Munich, Germany, 2012.

[11] Shkel, A. M., and Lumel, V., "Classification of the Dubins set," *Robotics and Autonomous Systems*, Vol. 34, 2001, pp. 179–202.

[12] Reeds, J., and Shepp, L., "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, Vol. 145, 1990, p. 367–393.

[13] Banzhaf, H., Berinpanathan, N., Nienhuser, D., and Zollner, J. M., "From G2 to G3 Continuity: Continuous Curvature Rate Steering Functions for Sampling-Based Nonholonomic Motion Planning," *IEEE Intelligent Vehicles Symposium (IV)*, Changshu, Suzhou, China, June 26-30, 2018.

[14] Oliveira, R., Lima, P. F., Cirillo, M., Martensson, J., and Wahlberg, B., "Trajectory Generation using Sharpness Continuous Dubins-like Paths with Applications in Control of Heavy-Duty Vehicles," *European Control Conference (ECC)*, Limassol, Cyprus, June 12-15, 2018.

[15] Drysdale, D., *An introduction to fire dynamics*, John Wiley & Sons, 2011.

[16] Butler, B. W., and Cohen, J. D., "Firefighter safety zones: a theoretical model based on radiative heating," *International Journal of Wildland Fire*, Vol. 8, No. 2, 1998, pp. 73–77.

[17] "Pixhawk 4 flight controller kit," http://www.holybro.com/product/pixhawk-4/.

[18] "PX4 Open Source Autopilot," https://px4.io/.

[19] "MAVLink Router by Intel," https://github.com/intel/mavlink-router.

[20] "MAVLink - MAV Communication Protocol," https://mavlink.io/.

[21] "QGroundControl - Ground Control Station for MAVLink Protocol," http://qgroundcontrol.com/.