Robotics and Autonomous Systems 112 (2019) 211–220

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Open-sector rapid-reactive collision avoidance: Application in aerial robot navigation through outdoor unstructured environments

Jake A. Steiner, Xiang He, Joseph R. Bourne, Kam K. Leang *

Design, Automation, Robotics and Control (DARC) Lab, Department of Mechanical Engineering, University of Utah Robotics Center, University of Utah, Salt Lake City, UT, 84112, United States

HIGHLIGHTS

- Reactive-based collision avoidance method that does not require maps.
- Low computational method allows the robot to quickly react to unknown obstacles.
- Experimental results show a smooth trajectory at a relatively high speed (3 m/s).

ARTICLE INFO

Article history:
Received 7 May 2018
Received in revised form 14 September 2018
Accepted 19 November 2018
Available online 1 December 2018

ABSTRACT

A new reactive collision avoidance method for navigation of aerial robots (such as unmanned aerial vehicles (UAVs)) in unstructured urban/suburban environments is presented. Small form-factor aerial robots, such as quadcopters, often have limited payload capacity, flight time, processing power, and sensing capabilities. To enhance the capabilities of such vehicles without increasing weight or computing power, a reactive collision avoidance method based on open sectors is described. The method utilizes information from a two-dimensional laser scan of the environment and a short-term memory of past actions and can rapidly circumvent obstacles in outdoor urban/suburban environments. With no map required, the method enables the robot to react quickly and navigate even when the enivornment changes. Furthermore, the low computational requirement of the method allows the robot to quickly react to unknown obstacles that may be poorly represented in the scan, such as trees with branches and leaves. The method is validated in simulation results and through physical experiments on a prototype quadcopter system, where results show the robot flying smoothly around obstacles at a relatively high speed (3 m/s).

1. Introduction

Local reactive collision avoidance capability is needed on robots traveling in unknown environments to quickly react to and circumvent obstacles. Due to their increasing versatility, small, lightweight, and agile aerial robots have many exciting applications in outdoor environments including environmental monitoring, surveillance, and package delivery [1–4]. All of these applications may require the robot to autonomously traverse areas with no prior knowledge of the obstacles in the environment. This is where the collision avoidance and motion planning methods described herein are created for: autonomously navigating through complex environments to a goal using no prior knowledge of the scenario. Since a constant communication link to a ground station cannot usually be guaranteed, all the navigation must be done rapidly on the vehicle to enable it to react quickly to unknown, possibly changing, environments. For useful application, the robot needs to not just maintain collision-free flight, but also fly long enough to complete a meaningful mission, *e.g.,* search, rescue, or environmental monitoring. The central limiting factors on flight time are weight and energy storage. These two factors restrict the sensing capabilities and computation power that a robot can utilize. The sensor used is further restricted by the environment in which the aerial robot operates. In outdoor environments, sensors need to work under bright light and shaded/dark areas. Thus, the navigation method needs to be computationally efficient and compatible with sensors that may be less than optimal. Navigation can be accomplished with two main methods. The first is with a user sending waypoints to a local planner, commonly using reactive collision avoidance, to navigate between waypoints. The second is sending waypoints to a hierarchical planner utilizing a global planner and using a local planner that can react quickly to inaccuracy in a map as describe in [5,6].

Many reactive collision avoidance methods are limited in their ability to find their way around obstacles without a global planner. Two types of sensors are often used for collision avoidance: cameras [5,7–9] and scanning laser rangefinders [5,9–13]. Due

* Corresponding author.
E-mail address: kam.k.leang@utah.edu (K.K. Leang).

https://doi.org/10.1016/j.robot.2018.11.016
0921-8890/

to cameras being passive sensors, they rely on the lighting in the environment. In shadows and in bright light the lack of contrast in the image can cause the obstacles to be obscured [6]. A scanning laser rangefinder is an active sensor and can sense small obstacles with more robustness in an outdoor environment with less reliance on lighting. A two-dimensional (2D) scanning laser rangefinder is often used for robot collision avoidance due to its accuracy and weight. When flying outdoors around obstacles such as trees, the laser scan can rapidly change as branches come in and out of view of the 2D scan as the robot maneuvers. Reactive collision avoidance based on artificial potential fields may produce a rapidly changing repelling force from obstacles in the above scenario. This oscillation could cause flight instability, leading to a collision. Reactive potential field has been successfully used on some aerial robots as described in [10] with a 2D scanning laser rangefinder and in [5] with a three-dimensional (3D) scanning laser rangefinder, ultrasonic sensors, and stereo cameras. In both scenarios, a hierarchical planner is relied upon to maneuver around obstacles. Another class of reactive collision avoidance is gap-based methods such as the method described in [11] which has been shown to work in highly cluttered indoor structured environments with ground robots moving at relativity slow speeds.

The reactive collision avoidance method described herein, named open sector (OS), is a reactive collision avoidance method created for use on multirotor aerial robots traveling in outdoor unstructured environments with a 2D scanning laser rangefinder. A collision avoidance method for unstructured environments means it cannot rely on any features in the environment to successfully navigate (such as lane lines on a road [14]). The OS method is able to navigate around urban/suburban environments with natural and man-made obstacles and reach waypoints without the need of a global planner for many complicated scenarios. For more complex urban environments, the OS method could be used to travel between sparsely placed waypoints planned by a global planner. It is demonstrated that the OS method can enable a robot to smoothly travel around obstacles at a high speed (3 m/s) and react quickly to changing obstacles since no map is saved of the environment. The novelty of the OS reactive collision avoidance method described herein is incorporating the effects of inertia to the target direction of travel by creating a virtual target that is modified with a short-term memory of past action, along with choosing an action vector to encourage smooth and safe travel around obstacles. The contribution of this work is demonstrating the efficacy of the OS method through simulation and physical experimental results. The OS method is computationally light weight and can run on aerial robots with a 2D scanning laser rangefinder, such as the robot pictured in Fig. 1. In the figure, the robot is shown to navigate around complex environments without oscillation at high speeds (2 m/s in a simulated complex environment and 3 m/s on a physical system in an outdoor area with trees). The method can smoothly navigate around poorly represented/rapidly changing representation of trees and other objects in outdoor unstructured environments.

The remainder of this paper is organized as follows. Section 2 focuses on summarizing related prior work in comparison to the proposed approach. The collision avoidance algorithm is described in Section 3. Simulation and experimental results and discussions are found in Section 4. Section 5 present conclusions and acknowledgments, respectively.

2. Related prior work

Navigation through environments can be accomplisshed through two main approaches: global map-based methods using a planner and local reactive collision avoidance where no obstacle information is saved. Global methods can start with a known map of the environment or build a global or local map while traveling through the area. If a map of the area is known, then a path can be found with several map and geometric motion planning methods; for example, search based methods such as breadth first search and A* [12,15], probabilistic methods such as rapidly exploring random trees [16] and probabilistic road maps [17], vector field histogram method(VFH) [18] and model predictive control methods [19,20]. These methods either require a map representation to be built while flying, which can be computationally expensive and be detrimentally affected by drift and inaccuracies in localization, or a map may need to be known before-hand, which in many scenarios is not possible. Another drawback of map-based methods is if the environment changes or errors are in the map, there is latency in updating or correcting the map with the sensor information. Map-based collision avoidance and planning methods are used in [9] on a light-weight quadcopter through indoor and outdoor environments.

Fig. 1. Example of an aerial robot navigating around buildings and trees in an outdoor environment.

Local-reactive collision avoidance methods can react more quickly to changes in the environment and do not rely on a correct representation in a map. There have been many reactive collision avoidance methods proposed, where a long standing commonly used method is artificial potential fields (PF) in which objects apply a repulsive force on the agent and the goal applies an attractive force [21]. This method can be used on a known map or reactively. It is used for its simplicity of implementation and can work well as a local planner between closely placed waypoints from a hierarchical planner. However, the PF method has several inherent drawbacks, such as becoming trapped in local minima and oscillating in narrow passages [21]. Several works have addressed these problems. To avoid traps, an approach described in [22] proposes adding a random low-magnitude velocity vector to influence the motion of the agent long enough to move it given that the current velocity command would otherwise be zero. Others have addressed the local minima and the oscillating problem by adding another force to avoid past positions [23]. Even with these modifications, complex environments can still cause the PF method to fail. Since the potential field method is the addition of vectors, it does not guarantee the robot travels in an obstacle-free area. Tunning parameters can help for certain scenarios but it is difficult to find a general set of parameters that work well in a variety of cases. Another approach to local-reactive methods on mobile robots is gap-based methods, where gaps in the environment are determined from a 2D laser scan. Gaps are found by checking for discontinuities in the laser scan ranges. The nearness diagram (ND) method [24] is one of the earliest reactive approaches to navigation with gaps. Oscillating behaviors and irrational deflections towards free space are challenges with ND. Several works have solved many of these limitations on ground robots. In [25] a tangential based method is proposed for solving these drawbacks. The robot kinematic

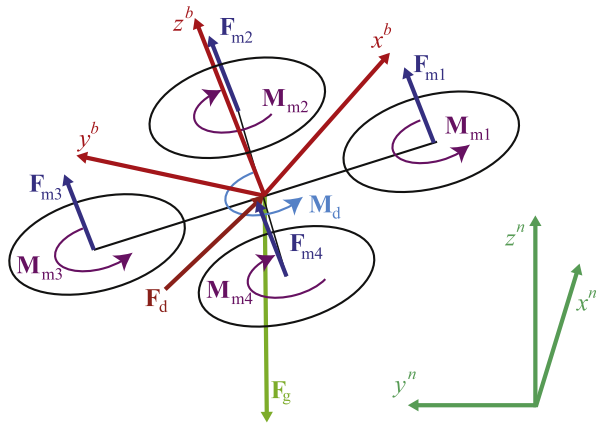


Fig. 2. Free body diagram of quadcopter aerial vehicle.

constraints are taken into consideration with gap-based navigation in the admissible gap method [11]. The gap-based methods rely on a relatively clean laser scan and has only been shown to work on slow moving ground robots. To use this method on a noisy outdoor laser scan a fair amount of post processing of the scan would be needed. This would lead to a reduction in the top safe speed of travel by lowering the reaction time of the robot.

The OS reactive collision avoidance method finds open sectors to travel in similar to VFH. Instead of using a probability threshold of obstacles in a sector from a histogram map to determine obstacle-free areas, a threshold is set on the range of the readings in the laser scan. An open sector has no obstacle within the threshold and is large enough to travel through, while a closed sector has an obstacle in the threshold, or has no obstacles, but is too narrow to travel through. The algorithm uses the distances in the closed sectors to determine where to travel in the open sector to produce a smooth trajectory. Adding inertia to the target with past actions enables the robot to navigate around complex obstacles without the drawbacks of increasing the inertia of the robot, as with avoiding the past in PF methods. Creating a virtual target from incorporating past actions was implemented on a quadcopter UAV using a PF method in [13].

3. Technical approach

In this section, the quadcopter dynamics and open sector (OS) local collision avoidance and navigation method for traversing an initially unknown environment are described. Open sectors are angle arcs with no obstacle within a certain range and large enough for the robot to travel through. To guide the robot to travel around trap scenarios that would inhibit potential field methods, such as U-shaped obstacles [26], a short-term memory of the past actions is used. The past actions influence the robot's decisions by modifying a virtual target. Once an open sector is chosen to travel through, the decision of where to travel uses an idea similar to [25] and encourages tangential motion when traveling around an obstacle. Navigation is done in the plane orthogonal to the yaw axis of the vehicle.

3.1. Quadcopter dynamics

A quadcopter is considered as the mobile robot system and the dynamics are described below [27]. First, the body frame $\mathcal{F}^b$ is defined with its origin at the center of gravity, x^b to the front, y^b to the left, and z^b pointing up. The inertial frame $\mathcal{F}^n$ is defined as a North-West-Up coordinate system. A free body diagram of the quadcopter is shown in Fig. 2. The force dynamics that determine the translational velocity in the inertial frame $\mathbf{v}^n = [\dot{x}, \dot{y}, \dot{z}]^T$ is given by

$$m\dot{\mathbf{v}}^n = \mathbf{R}_b^n(\mathbf{F}_d^b + \mathbf{F}_m^b) + \mathbf{F}_g^n, \tag{1}$$

where m is the mass of the vehicle, $\dot{\mathbf{v}}^n$ is the acceleration of the vehicle, $\mathbf{F}_d^b$ is the force due to drag, $\mathbf{F}_m^b$ is the sum of propeller thrust ($\mathbf{F}_m = \sum_{i=1}^4 \mathbf{F}_{m_i}$), and $\mathbf{F}_g^n$ is the gravity force. The rotation matrix $\mathbf{R}_b^n$ rotates the forces from the body frame (superscript b) to the inertial frame (superscript n). To determine the angular velocity in the body frame $\omega^b = [p, q, r]^T$, are given by

$$\mathbf{J}\dot{\omega}^b = \mathbf{M}_d + \mathbf{M}_m, \tag{2}$$

where $\mathbf{J}$ is the inertia matrix for the robot, $\mathbf{M}_d$ is caused by drag, and $\mathbf{M}_m$ is the sum of torques from the propulsion ($\mathbf{M}_m = \sum_{i=1}^4 \mathbf{M}_{m_i}$).

To determine the kinematics of the robot, which relates the angular velocity $[\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ in $\mathcal{F}^b$ to $[p, q, r]^T$ in $\mathcal{F}^n$, the following equation is used:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{V} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \quad \text{where } \mathbf{V} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix}. \tag{3}$$

Additional details about this model can be found in [28].

3.2. Collision avoidance

The collision avoidance algorithm uses a 2D scan of the environment centered on the robot around the z^b axis. The scan is processed to determine reasonable directions of travel that will be collision-free. The best collision-free direction is chosen that will circumvent obstacles and reach the goal.

3.2.1. Virtual wall

The input of the OS method is the scan data from the scanning laser rangefinder, which has a 90° blind spot in the scan. While traveling between waypoints, it is important the scanning laser rangefinder is pointing in the direction of travel. A proportional–integral–derivative (PID) controller is used for yaw control to keep the angle of the true velocity opposite of the blind spot in the scanning laser rangefinder. To keep the robot collision-free, it is important to tune the aggressiveness of the yaw controller. If the controller is too slow, the robot will back up into an obstacle behind it before it can sense the obstacle. Too aggressive of a controller could cause the aerial robot to become unstable.

The blind spot of the scan can be handled in several ways in the algorithm. The simplest being to count the blind spot as not an option for the commanded direction of travel. However, this leads to the robot executing large turns when needing to reverse direction. It was determined that using a virtual wall inserted into the laser scan between the last and first range readings allowed the robot to be more agile and travel more smoothly through the environment. If the yaw controller is adequately responsive, in most cases the robot will yaw around to see an obstacle previously in the blind spot and be able to avoid before collision. Occasionally when the laser scan is not pointed in the direction of travel, the virtual wall approach may cause a collision with a small obstacle in the blind spot. In the system described, the improved performance outweighed the associated risks. Demonstration of the virtual wall can be seen in Fig. 3.

3.2.2. Open sectors

The open sectors are determined by cycling clockwise once through the scan ranges starting from directly behind the robot ($\theta = 2\pi$). In an open sector, no obstacle can be within a threshold distance. This distance is referred to as the look-ahead distance d_a and has a large effect on the behavior of the robot. The length of

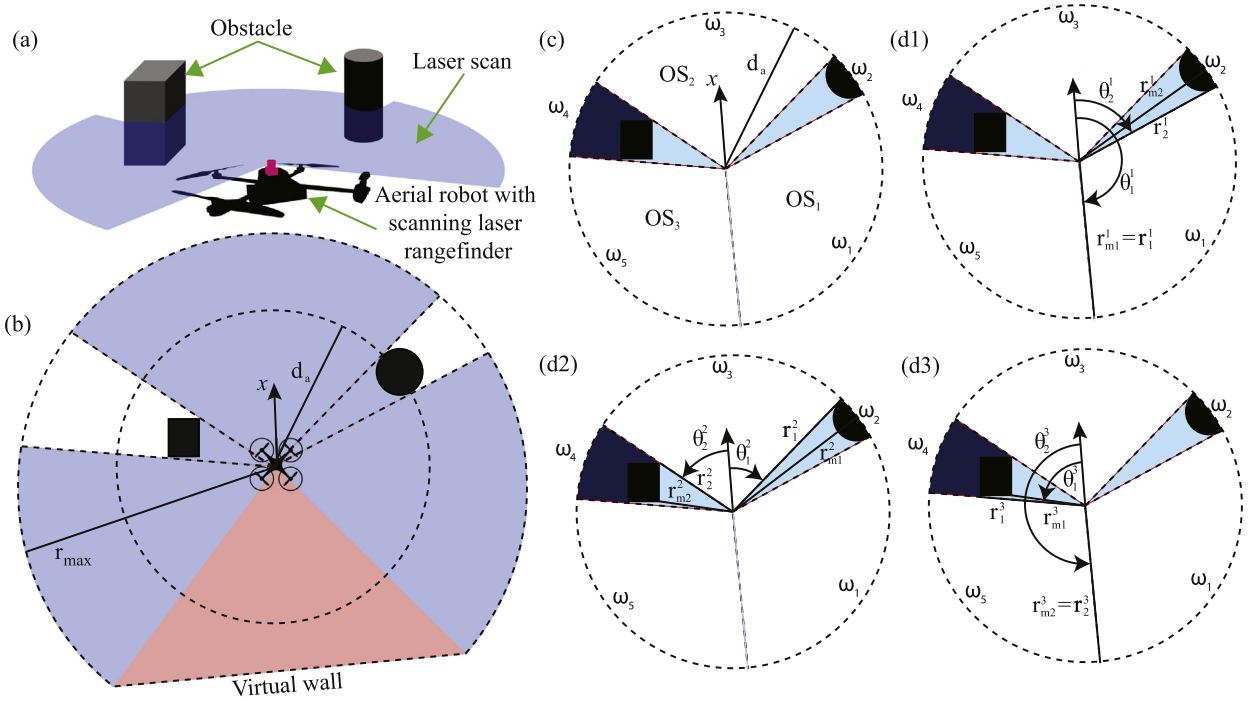


Fig. 3. Graphical representation of the open sector (OS) collision avoidance method: (a) Shows an aerial robot traveling through an environment with two obstacles in the laser scan. (b) Shows the laser scan projecting to the maximum sensed distance r_{max}. The raw laser scan is modified by adding a virtual wall in the blind spot between the first and last scan ranges. The OS algorithm only considers objects within the look-ahead distance d_a, shown as the smaller circle, in the modified laser scan. Any range reading greater then d_a is ignored. (c) Shows how the scan (only considering ranges less than d_a) is segmented into sectors. Sectors ω_2 and ω_4 are closed due to obstacles. Sectors ω_1, ω_3, and ω_5 are open sectors OS_1, OS_2, and OS_3, respectively. (d) Shows the variables saved for open sectors 1, 2, and 3. The superscript corresponds to the OS the variable is related to. The information saved along with each open sector are the angle at the start and stop, θ_1, θ_2; the range at the start and stop angles, r_1, r_2; and the minimum range in the adjacent closed sectors r_{m1}, r_{m2}.

d_a depends on the robot dynamics, the sensor used, and the environment. For example, when an aerial robot flies near the ground, the scanning laser rangefinder will often inadvertently sense the ground while maneuvering; thus, d_a should be short enough so the ground is not usually included as an obstacle. Also, the larger d_a, the less explorative the robot will be. It will consider possibly open passageways closed before getting close enough to determine. Alternatively, a small look-ahead distance will cause the robot to incorrectly reason about actions when traveling around an obstacle. Before the open sectors are processed to determine the best to sector to travel through, narrow open sectors that will likely be too small to travel through are closed. Each sector is checked for a minimum arc angle as well as a minimum Cartesian distance between the readings at the start and stop of the open sector. If it is too small, then the sector is considered closed. The information saved along with each open sector are the angle at the start and stop, θ_1, θ_2, respectively; the range at the start and stop angles, r_1 and r_2, respectively; and the minimum range in the adjacent closed sectors r_{m1} and r_{m2}, respectively.

3.2.3. Virtual target

In previous work [13] the following method was used to determine a virtual target for a PF-based method. The virtual target ϑ used in OS increases the robots ability to navigate around traps compared to always trying to head for a target vector $\mathbf{T}$ pointing directly towards the goal. There are two factors that influence the direction of the virtual target: the goal location and the past actions. The first factor, the goal location, must be included if a goal is to be reached, while the latter can be scaled empirically in the environment the robot is working in. To incorporate the past actions into the virtual target, a vector from m past actions

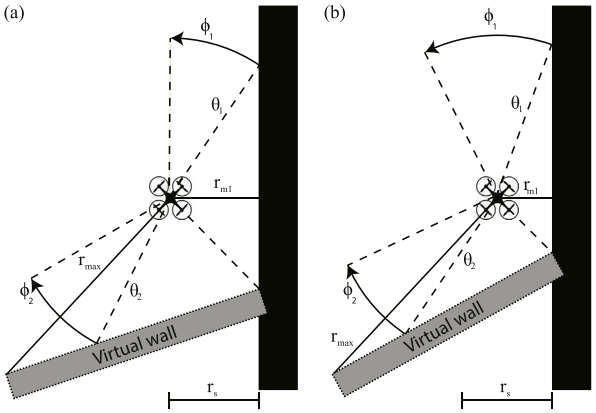


Fig. 4. Examples of safety boundaries, ϕ, when traveling near an obstacle. Safety boundaries keep the robot from navigating to close to obstacles. The virtual wall method of handling the blind spot of the scanning laser rangefinder is shown. Two scenarios are portrayed: (a) when the minimum distance in the adjacent closed sector r_{m1} is equal to the minimum desired distance from obstacles r_s and the safety boundaries do not allow the robot to command a velocity closer than tangential to the wall, and (b) when $r_{m1} < r_s$ and the safety boundaries force the robot to travel farther away from the wall.

is created using

$$\mathbf{A} = \sum_{i=k-m}^{k} \mathbf{V}_i, \tag{4}$$

where $\mathbf{V}_i$ is the action vector at step i and k is the current step. The virtual target angle θ_ϑ is found from the target vector angle $\theta_\mathbf{T}$ by

creating a weighted average of $\theta_{\mathbf{A}}$ and $\theta_{\mathbf{T}}$,

$$\theta_{\vartheta} = \Psi(\theta_{\mathbf{T}}, \theta_{\mathbf{A}})u + \theta_{\mathbf{T}}, \tag{5}$$

where $u \in (0, 1]$ determines the impact $\theta_{\mathbf{A}}$ has on the virtual target and $\Psi(\theta_x, \theta_y)$ returns the smaller angle from θ_x to θ_y and is defined as follows:

$$\Psi(\theta_x, \theta_y) = \begin{cases} -2\pi + \gamma, & \text{if } \gamma > \pi \\ 2\pi + \gamma, & \text{if } \gamma < -\pi \\ \gamma, & \text{otherwise,} \end{cases} \tag{6}$$

with $\gamma = \theta_y - \theta_x$. Incorporating past actions into the virtual target enables the robot to successfully navigate around many obstacles without getting trapped.

3.2.4. Action vector

Once the virtual target vector angle θ_{ϑ} is found, the best open sector to travel through is found by determining the nearest open sector to the virtual target. After the open sector to travel through is determined, safety boundaries are calculated. Safety boundaries are used to encourage the robot to travel tangential to obstacles and maintain a safe distance. The graphical representation of the safety boundaries for traveling along a flat wall are shown in Fig. 4. Since traveling directly toward the start θ_1 or stop θ_2 angles of the sector would cause oscillating behavior and cutting corners too close, safety boundaries are calculated to keep the robot traveling a safe distance from an object. A desired safety radius r_s that the user would like the robot to stay away from objects is set. The following equations are used to calculate the safety boundaries for the start θ_{sb1} and stop θ_{sb2} of the sector:

$$\theta_{sb1} = \theta_1 + \phi_1 \text{ and } \theta_{sb2} = \theta_2 + \phi_2, \tag{7}$$

where

$$\phi_1 = \begin{cases} \pi/2 - \arccos(r_s/r_1), & \text{if } r_{m1} > r_s \\ \pi/2 - \arccos(r_s/d_a) + k(r_s - r_{m1}), & \text{otherwise,} \end{cases} \tag{8}$$

and

$$\phi_2 = \begin{cases} -\pi/2 + \arccos(r_s/r_2), & \text{if } r_{m2} > r_s \\ -\pi/2 + \arccos(r_s/d_a) - k(r_s - r_{m2}), & \text{otherwise,} \end{cases} \tag{9}$$

In the equation above, k determines how aggressively the robot avoids an object once it is within r_s distance. Two scenarios determine how the safety boundaries are calculated. First, when no obstacle is within r_s the safety boundaries are set to encourage tangential motion along the obstacle. Second, if an obstacle gets closer than r_s the safety boundary on the side of the obstacle is increased to encourage motion away from the obstacle.

Once the desired sector to travel through and the safety boundaries are found, the direction of the action vector sent to the robot is determined using Algorithm 1, where the boolean function $\zeta(\theta_x, \theta_y, \theta_z)$ returns *true* if θ_z is contained in the positive angle arc from θ_x to θ_y and *false* otherwise, $\eta(\theta_x, \theta_y)$ determines the positive angle from θ_x to θ_y, and $\xi(\theta_x, \theta_y, \theta_z)$ determines if θ_z is closer to θ_x or θ_y and is defined as:

$$\xi(\theta_x, \theta_y, \theta_z) = \begin{cases} 1, & \text{if } \eta(\theta_x, \theta_z) < \eta(\theta_z, \theta_y) \\ 2, & \text{otherwise.} \end{cases} \tag{10}$$

If the virtual target lies within the open sector, it is then checked to see if it is outside the safety boundaries. If it is, then the action vector angle is equal to the virtual target angle, $\theta_{\mathbf{V}} = \theta_{\vartheta}$, and the robot is free to move straight along the virtual target. If the virtual target lies within the closed sector or in the safety boundaries, the side of the sector (θ_1 or θ_2) which the target is closer to is found. The action vector angle is then set to the angle of the safety boundary on the side closer to the target. In the case where the start and stop safety boundaries, θ_{sb1} and θ_{sb2}, overlap in the open sector the action vector is set to the edge of the safety boundary with the larger ϕ. A graphical representation of how the action vector is chosen is shown in Fig. 5.

Algorithm 1: Determining action vector direction.

Result: Actions vector angle $\theta_{\mathbf{V}}$

```
if ζ(θ1, θ2, θϑ) then
    if ζ(θsb1, θsb2, θϑ) then
        θV = θϑ
    else
        if η(θ1, θ2) < |φ1|+|φ2| then
            if φ2 > φ1 then
                θV = θsb2
            else
                θV = θsb1
            end
        else if ξ(θ1, θ2, θϑ) == 1 then
            θV = θsb1
        else
            θV = θsb2
        end
    end
else
    if ξ(θ1, θ2, θϑ) == 1 then
        θV = θsb1
    else
        θV = θsb2
    end
end
if ζ(θ2, θ1, θV) then
    if ξ(θ1, θ2, θV) = 1 then
        θV = θ1
    else
        θV = θ2
    end
end
```

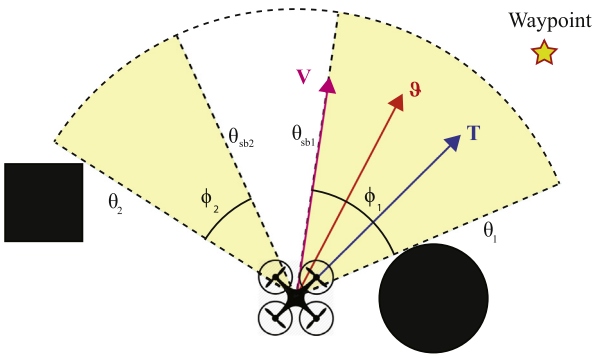


Fig. 5. The target vector **T**, virtual target ϑ, and the resulting action vector **V** shown for an open sector with safety boundaries ϕ_1 and ϕ_2. For this example the virtual target lies withing the safety boundary formed from adding ϕ_1 to θ_1. The action vector **V** will be the nearest angle to ϑ outside of the safety boundaries, this angle is θ_{sb1}.

3.3. Implementation

For safe real world implementation, there are a few cases that need to be handled to ensure stable collision-free navigation. The

modified action vector $\mathbf{V}'$ is used as the command to the robot velocity controller.

In an unstructured environment an obstacle may suddenly appear in the scan or a disturbance, such as a gust of wind, may push the robot near an obstacle. In this case the robot will take emergency action to avoid. If a range reading is less than a certain radius r_e, where $r_e < r_s$, and emergency action is needed, then a more aggressive avoidance maneuver is applied by directly incorporating a vector pointing away from the obstacle into the modified action vector. The following repulsive vector is incorporated into the modified action vector to ensure motion away from the obstacle,

$$\mathbf{f} = \sum_{i=1}^{q} (r_s - \|\mathbf{r}_i\|) \frac{-\mathbf{r}_i}{\|\mathbf{r}_i\|}, \tag{11}$$

where q is the number of range vectors r whose magnitude is less than r_s. In this case, the angle of $\mathbf{V}'$ is found with the following equation, $\theta_{\mathbf{V}'} = \theta_{\mathbf{V_e}}$ where $\mathbf{V_e}$ is found with,

$$\mathbf{V_e} = \frac{\mathbf{f}}{\|\mathbf{f}\|} + \begin{pmatrix} \cos(\theta_{\mathbf{V}}) \\ \sin(\theta_{\mathbf{V}}) \end{pmatrix}. \tag{12}$$

The other two scenarios where the action vector from OS needs to be modified is when there are no open sectors in the scan or when the waypoint is close. When there are no opens sectors in the scan the OS method will fail to find an action vector. When the distance to the waypoint is less than look-ahead distance, d_a, used in OS, the waypoint could be in a closed sector but still reachable by the robot if it were to stop in front of the obstacle. The OS method does not allow travel towards a closed sector while PF will allow the robot to attempt to reach the waypoint close to the obstacle. To allow the aerial robot to travel in these situations the algorithm switches to reactive PF method as described in [13] called PF-IPA. The virtual target is the attractive force and the repulsive force is found with,

$$\mathbf{w} = \sum_{i=1}^{p} \frac{-a\mathbf{r}_i}{\|\mathbf{r}_i\|^b}, \tag{13}$$

where $\mathbf{r_i}$ is the range vector of each reading in the scan and p is the number of readings in the scan. Constants a and b are parameters that can be adjusted to change the behavior of $\mathbf{w}$. The angle of $\theta_{\mathbf{V}'}$ is then found with the following equations, $\theta_{\mathbf{V}'} = \theta_{\boldsymbol{\tau}}$ where

$$\boldsymbol{\tau} = \frac{\mathbf{w}}{\|\mathbf{w}\|} + \frac{\vartheta}{\|\vartheta\|}. \tag{14}$$

The modified action vector angle for all scenarios is found using,

$$\theta_{\mathbf{V}'} = \begin{cases} \theta_{\boldsymbol{\tau}}, & \text{if } d_{wp} < d_a \text{ or no open sectors found} \\ \theta_{\mathbf{V_e}}, & \text{if any range is } < r_e \\ \theta_{\mathbf{V}}, & \text{otherwise,} \end{cases} \tag{15}$$

where d_{wp} is th distance to the waypoint from the robots current position. Once the angle of the action vector is found, the magnitude is determined. The user sets a desired speed v_d. If the true velocity of the robot is in the open sector then the $\|\mathbf{V}\| = v_d$, otherwise the true velocity is in a closed sector and the robot reduces the magnitude to a set safe speed v_s. A moving average filter is also placed on the magnitude of the action vector to ensure smooth flight.

4. Results and discussions

The OS reactive collision avoidance was tested in simulation and on a real system in an outdoor environment. For the simulation and physical experiment, the OS algorithm was set to run at 40 Hz to match the speed of the scanning laser rangefinder.

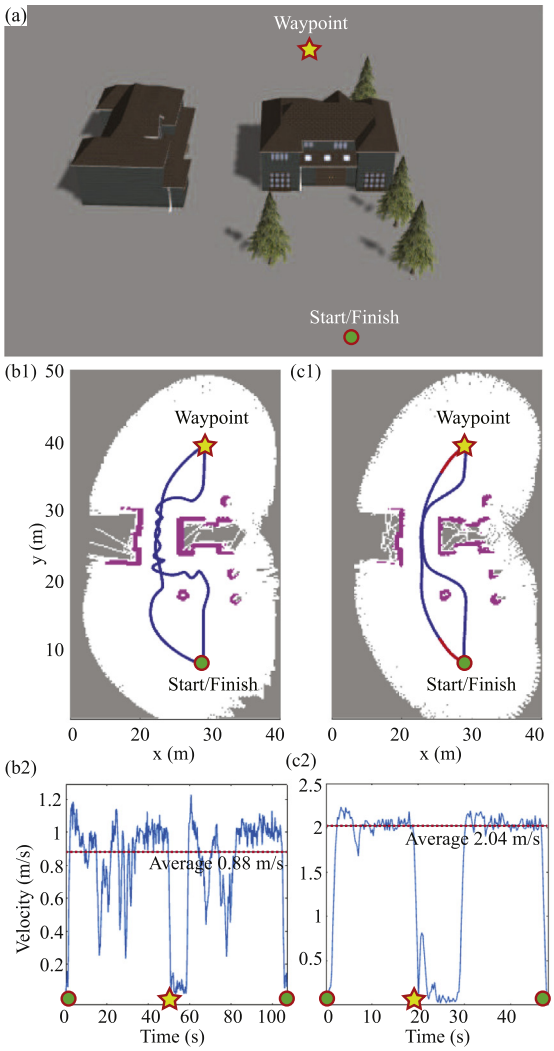


Fig. 6. Simulated results for potential field-based reactive collision avoidance (PF-IPA) [13] and open sector (OS) reactive collision avoidance. The robot starts at the start position (circle) traveled to the waypoint (star) then returned to the start position. PF-IPA results: (b1) a post-processed map of trajectory (blue curve) and the obstructions (purple dots) and (b2) the velocity of the robot during simulation using PF-IPA. The desired velocity is 1 m/s and the average velocity between waypoints is 0.88 m/s (dashed line). OS results: (c1) the trajectory of the robot using OS collision avoidance and (c2) the velocity during the simulation using OS. The command velocity is 2 m/s and the average velocity between waypoints is 2.04 m/s. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Simulation

To evaluate the performance and tune the OS collision avoidance algorithm, simulations were performed in the Gazebo robot simulator [29]. The OS method was tested in a simulated environment with a few trees and a passageway. The PF-IPA method was also tested in the same environment for comparison. Fig. 6 shows the environment, trajectory, and velocity of the quadcopter during the test. The PF-IPA oscillated when traveling through the passageway and when approaching the flat walls of the buildings as seen in Fig. 6(b1). The OS method, in comparison, did not oscillate through the passageway or when approaching the buildings. This is due to the look-ahead distance in OS considering obstacles within a distance d_a from the robot. The look-ahead distance d_a was set to 7 m, the number of past actions incorporated into the virtual target, m in Eq. (4), was set to 90, and the weighting of the

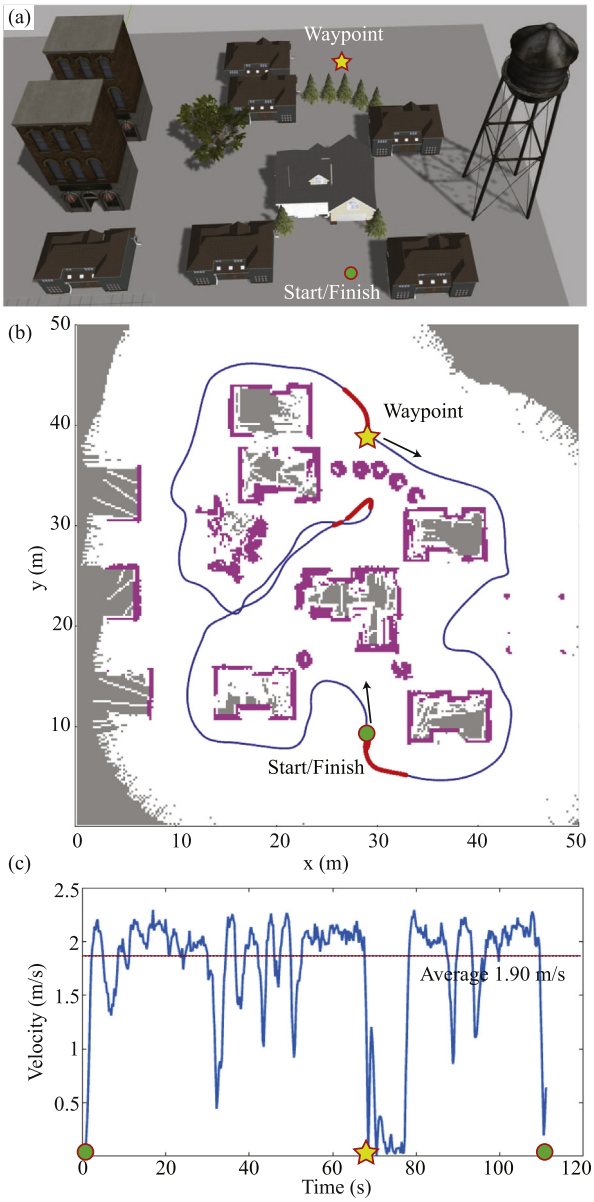


Fig. 7. Simulation of OS reactive collision avoidance in an outdoor environment. The desired velocity was set to 2 m/s. The robot started at the start position (circle) traveled to the waypoint (star) then returned to the start position. (a) Shows the simulated environment in the Gazebo robot simulator along with the waypoints. (b) Shows a post-processed map showing trajectory (blue and red curve) and the obstruction (purple dots). The blue part of the trajectory curve was using OS and the red part shows when PF-IPA was used (c) Shows the velocity during the simulation. After reaching the WP (star) the UAV holds position for 10 s before continuing to the finish position. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

past action vector, u in Eq. (5), was set to 0.6 for the simulations. The trajectory and velocity are smooth around the obstacle and the velocity remained near the desired speed of 2 m/s the entire section between waypoints. Near the waypoints, when the distance to the goal location is less than the look-ahead distance set in OS, the methods switches to PF-IPA. This is shown on the trajectory plot with a red line.

The OS method is tested more intensively in a simulated environment with buildings, trees, and a large trap situation in Fig. 7. Incorporating past actions allowed the robot to navigate out of large trap situations, for example near the center of the map. The

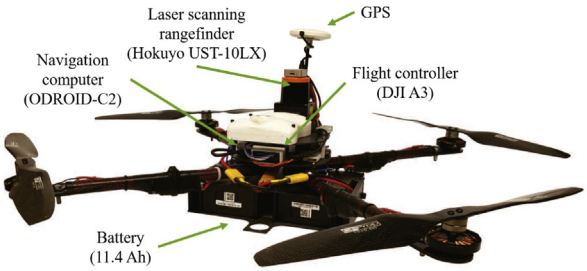


Fig. 8. The prototype aerial robot Enif equipped with a 2D scanning laser rangefinder for obstacle detection and a GPS for localization. Navigation is run on the onboard navigation computer. The Enif UAV has a flight time of 40 min.

two scenarios where the algorithm switches to a PF-IPA method are demonstrated in the simulation. The parts of the trajectory that were planned with PF-IPA are shown with a red line in Fig. 7(b). The method changes to PF-IPA when entering the large trap in the center of the map. No open sectors are found in the scan. The robot smoothly transitions to PF-IPA then back to OS once it detects open sectors to travel in. At the waypoint the robot holds its position for 10 s before continuing to the finish position. For the above simulation, the average velocity while traveling between waypoints was 1.9 m/s. The robot successfully navigated out of several trap situations while maintaining a high velocity and a smooth trajectory.

4.2. Experiments

Physical experiments are performed using the Enif custom-designed autonomous chemical senssing UAV platform [13] shown in Fig. 8. The Enif UAV is an aerial quadcopter vehicle designed for environmental monitoring and chemical sensing. The Enif system is comprised of a UAV and a ground station. Through the ground station the user can select waypoints and set parameters such as desired speed, v_d. The navigation and collision avoidance run on the onboard navigation computer (ODROID-C2) which sends the action vector $\mathbf{V}$ to the flight controller (DJI A3). The 2D scanning laser rangefinder used is the Hokuyo UST-10LX. The scanning rangefinder power usage is 2.94 W, the weight is 160 g, and the accuracy is $\pm$40 mm. The ODROID-C2 has a 1.5 GHz ARM processor and 2 GB of RAM. The OS method uses 28% of the CPU and 96 MB of the RAM. For localization, a GPS is used.

The OS method is tested on a real aerial robot system in several outdoor environments with buildings and natural obstacles. The method is compared to the PF-IPA method presented in [13]. Fig. 9 shows the trajectory of the robot as it travels through an outdoor environment with trees using both the OS method and the PF-IPA method. The height above ground was set to 2 m, the desired speed for the OS algorithm was set to 3 m/s, the look-ahead distance d_a was set to 7 m, the number of past actions incorporated in to the virtual target, m in Eq. (4), was set to 50, and the weighting of the past action vector, u in Eq. (5), was set to 0.7. The map shown was created off-line after the experiment from laser scan data collected during the flight. As can be seen, the OS method produces a smooth trajectory for the robot with no oscillation. The PF-IPA method significantly oscillates when traveling between narrow passageways, for example between obstacles T4 and T5 in the figure. The PF-IPA method was stable up to a desired velocity of 1 m/s, while the OS method was able to travel through the environment at a speed of 3 m/s with a much smoother trajectory. The OS method considers obstacle much earlier than the PF-IPA method, leading to a smoother trajectory and allowing for a higher velocity without collision.

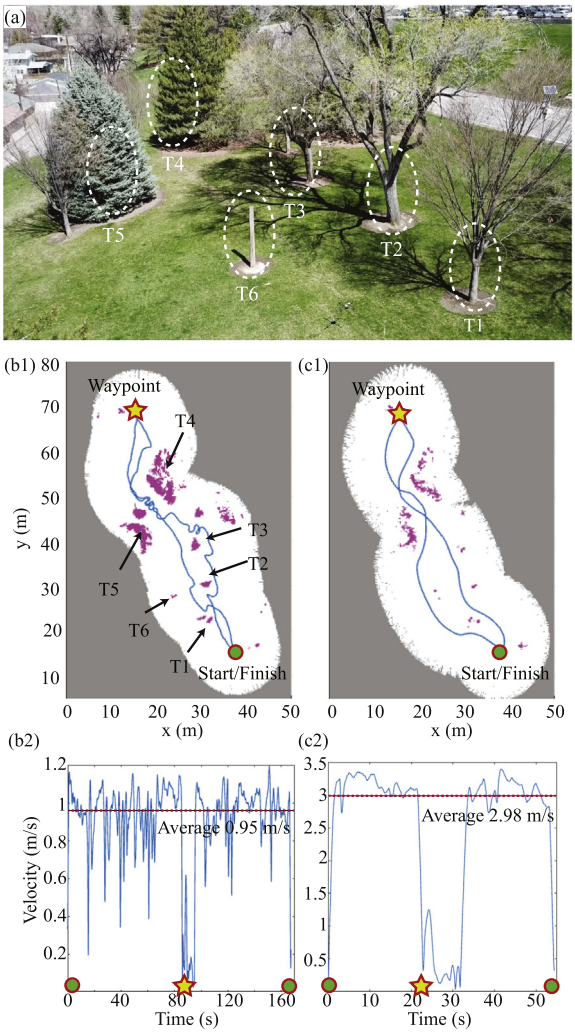


Fig. 9. Comparing open sector (OS) collision avoidance to a potential field method from [13] (PF-IPA) in an outdoor environment with trees using a quadcopter aerial robot. The robot started at the start position (circle) and traveled to the waypoint (star) then returned to the start position. (a) A photo of the environment with main features labeled. (b) The trajectory and velocity of the robot while traveling through an outdoor environment with trees using PF-IPA. Post-processed map showing trajectory (blue solid curve), and the obstructions (purple dots, acquired by the robot). The desired velocity was 1 m/s. (C) The trajectory and velocity of the robot while traveling through the same outdoor environment using OS collision avoidance. The desired velocity was 3 m/s.

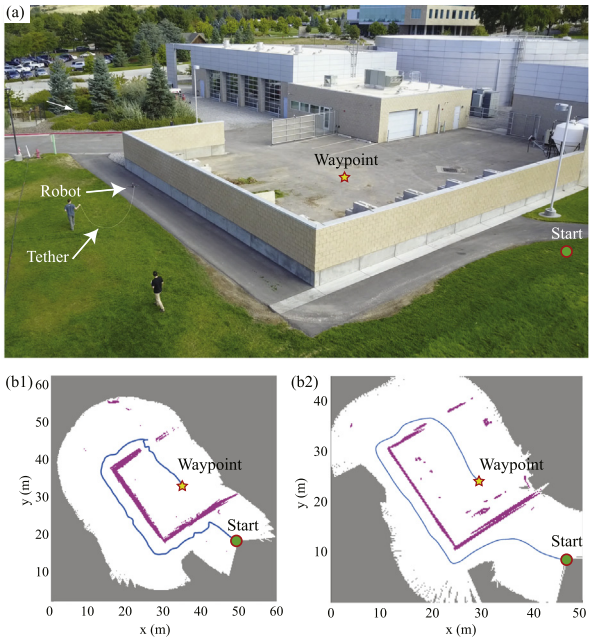


Fig. 10. A quadcopter aerial robot traveling to a waypoint in a low structure using a potential field-based method (PF-IPA) from [13] and the open sector (OS) method: (a) an aerial view of the environment (rope tether attached to robot for safety); (b1) The trajectory of the robot traveling to the waypoint using PF-IPA at a desired speed of 1 m/s. (b2) The trajectory of the robot traveling through the environment using OS at a desired speed of 2 m/s.

The OS method is also tested in an outdoor environment with a structure. The OS method is compared to the results of the PF-IPA method presented in [13] in Fig. 10. The robot successfully reaches the waypoint placed inside a roofless structure with four walls and a gateway opening. The average speed for the OS method was 2 m/s. The PF-IPA method results presented in [13] had an average speed of 1 m/s and oscillation when traveling along the walls and when entering the gateway opening. The OS method produces a smooth trajectory tangentially along the walls of the obstacle and keeps a relatively constant distance to the wall, even when traveling around corners.

Lastly, the OS method was used to navigate the aerial robot through a tree environment with waypoints over a large distance. The trajectory is overlaid over a satellite image in Fig. 11. The distance between the start location and waypoint 1 is approximately 160 m, waypoint 2 is approximately 120 m from waypoint 1, and the finish location is approximately 280 m from waypoint 2. At each waypoint the quadcopter held position for 10 s. The desired speed was set to 2 m/s and the average speed by the robot was measured at 2.03 m/s.

The OS collision avoidance method enables the quadcopter to navigate through multiple outdoor environments with trees and building while traveling between speeds of 2 to 3 m/s. There was comparatively no oscillation compared to potential field methods in narrow passageways and along walls. The velocity remained relatively constant between waypoints even through complex environments. The look-ahead distance and robot dynamics determine how fast the robot can travel through an environment. The look-ahead distance needs to be long enough for the robot to come to a stop and turn around if a wall is encountered at the end of a passageway. With a larger look-ahead distance, sectors may be considered closed before getting close enough to determine. Since the target is modified based on past commands, which are in open sectors, this could lead the robot to never investigate close enough to consider possible openings. In a cluttered environment a large look-ahead distance would cause poor performance, possibly leading to no open sectors to be found in the scan and the robot switching to PF-IPA for navigation. If the look-ahead distance is increased, attention needs to be paid to how often, usefully as the quadcopter tilts, the ground is seen in the scan. This could be remedied by placing the scanning laser rangefinder on a gimbal, this was left off the Enif quadcopter due to weight and flight time constraints. Incorporating past actions gives the robot a short-term memory that enables the robot to navigate out of traps situations without having to detect that it is in a trap or build a map. Incorporating more past actions gives the target more inertia. This will cause the robot to make larger turns around corners. It does not, however, add inertia to the robot, so if a new obstacle is encountered it will react just as agile regardless of how long it has been traveling in that direction, unlike avoiding the past in potential field methods.

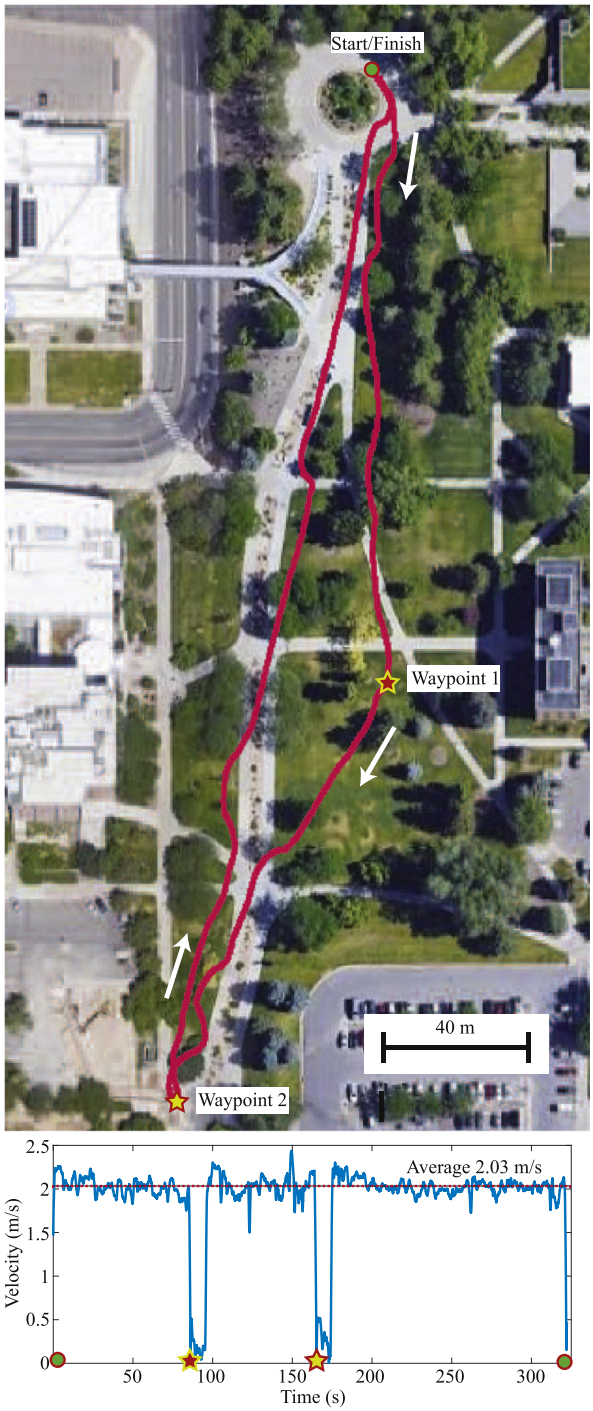


Fig. 11. The trajectory of a quadcopter aerial vehicle through an outdoor environment using open sector (OS) reactive collision avoidance. The robot starts at the start position (circle), travels to waypoint 1 (star), then to waypoint 2, and back to the start location (circle). The average speed is 2.0 m/s. The robot holds position at each waypoint for 10 s.

5. Conclusions and future work

This paper focused on a reactive collision avoidance method for aerial robot navigation in unstructured urban/suburban environments. The open sector method locates angle arcs in the scan where no range reading is less than a threshold and large enough for the robot to travel through safely. The target vector is modified to create a virtual target that incorporates a short-term memory of past actions. The best open sector is chosen with a virtual target and enables the robot to rapidly and smoothly travel around complex obstacles. Simulation and experimental results on an environmental monitoring quadcopter UAV (Enif) validated the algorithm. The robot was able to rapidly navigate through unstructured outdoor environments at speeds between 2 and 3 m/s with a smooth trajectory. The open sector method was able to escape traps that would block potential field-based methods.

Future work will include using a model of the robot to only consider open sectors that can be reached from the current state. Also, open sector may be able to be extended for use with 3D collision avoidance using a point cloud obtained from a scanning laser rangefinder or multiple cameras.

Acknowledgments

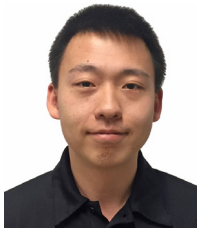
This work is supported, in part, by the University of Utah, USA, the National Science Foundation's Partnership for Innovation Program, USA (Grant No. 1430328), and U.S. ARMY STTR Program, USA grant No. W9132T-16-C-0001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] V. Kumar, N. Michael, Opportunities and challenges with autonomous micro aerial vehicles, Int. J. Robot. Res. 31 (11) (2012) 1279–1291.
[2] M.A. Goodrich, B.S. Morse, C. Engh, J.L. Cooper, J.A. Adams, Towards using unmanned aerial vehicles (UAVs) in wilderness search and rescue: Lessons from field trials, Interact. Stud. 10 (3) (2009) 453–478.
[3] P.K. Freeman, R.S. Freeland, Agricultural UAVs in the U.S.: potential, policy, and hype, Remote Sens. Appl. Soc. Environ. 2 (2015) 35–43.
[4] A. Bhardwaj, L. Sam, Akanksha, F.J. Martin-Torres, R. Kumar, UAVs as remote sensing platform in glaciology: Present applications and future prospects, Remote Sens. Environ. 175 (2016) 196–204.
[5] M. Nieuwenhuisen, D. Droeschel, M. Beul, S. Behnke, Obstacle detection and navigation planning for autonomous micro aerial vehicles, in: International Conference on Unmanned Aircraft Systems, Florida, 2014, pp. 1040–1047.
[6] S. Scherer, S. Singh, L. Chamberlain, M. Elgersma, Flying fast and low among obstacles: Methodology and experiments, Int. J. Robot. Res. 27 (5) (2008) 549–574.
[7] H. Oleynikova, D. Honegger, M. Pollefeys, Reactive avoidance using embedded stereo vision for MAV flight, in: IEEE International Conference on Robotics and Automation, Washington, 2015, pp. 50–56.
[8] K. Schmid, P. Lutz, T. Tomić, E. Mair, H. Hirschmüller, Autonomous vision-based micro air vehicle for indoor and outdoor navigation, J. Field Robot. 31 (4) (2014) 537–570.
[9] K. Mohta, K. Sun, S. Liu, M. Watterson, B. Pfrommer, J. Svacha, Y. Mulgaonkar, C.J. Taylor, V. Kumar, Experiments in fast, autonomous, GPS-denied quadrotor flight, in: International Conference on Robotics and Automation, ICRA, 2018.
[10] J. Jackson, D. Wheeler, T. McLain, Cushioned extended-periphery avoidance: A reactive obstacle avoidance plugin, in: International Conference on Unmanned Aircraft Systems, ICUAS, Virginia, 2016, pp. 399–405.
[11] M. Mujahed, D. Fischer, B. Mertsching, Admissible gap navigation: A new collision avoidance approach, Robot. Auton. Syst. 103 (2018) 93–110.
[12] J.Q. Cui, S. Lai, X. Dong, B.M. Chen, Autonomous navigation of UAV in Foliage Environment, J. Intell. Robot. Syst., Theory Appl. 84 (1–4) (2016) 259–276.
[13] X. He, J.R. Bourne, J.A. Steiner, C. Mortensen, K.C. Hoffman, C.J. Dudley, B. Rogers, D.M. Cropek, K.K. Leang, Autonomous chemical sensing aerial robot for urban / suburban environmental monitoring (Under Review), IEEE Syst. (2018).
[14] S. Kolski, D. Ferguson, M. Bellino, R. Siegwart, Autonomous driving in structured and unstructured environments, in: IEEE Intelligent Vehicles Symposium, 2006.
[15] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, Syst. Sci. Cybern. 4 (2) (1968) 100–107.
[16] S.M. LaValle, Rapidly-exploring random trees: A new tool for path planning, Tech. rep., vol. 129, Iowa State University, Ames, IA, USA, 1998, 98–11.
[17] L.E. Kavraki, P. Svestka, J. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robot. Autom. 12 (4) (1996) 566–580.
[18] J. Borenstein, Y. Koren, The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Trans. Robot. Autom. 7 (3) (1991) 278–288.

[19] J. Wu, H. Wang, N. Li, P. Yao, Y. Huang, Z. Su, Y. Yu, Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by adaptive grasshopper optimization algorithm, Aerosp. Sci. Technol. 70 (2017) 497–510.
[20] D. Shim, H. Chung, H.J. Kim, S. Sastry, Autonomous exploration in unknown urban environments for unmanned aerial vehicles, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2005, p. 6478.
[21] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1) (1986) 90–98.
[22] R.C. Arkin, Motor schema-based mobile robot navigation, Int. J. Robot. Res. 8 (4) (1989) 92–112.
[23] T. Balch, R. Arkin, Avoiding the past: a simple but effective strategy for reactive navigation, in: Proceedings IEEE International Conference on Robotics and Automation, Georgia, 1993, pp. 678–685.
[24] J. Minguez, L. Montano, Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios, IEEE Trans. Robot. Autom. 20 (1) (2004) 45–59.
[25] M. Mujahed, D. Fischer, B. Mertsching, Tangential gap flow (TGF) navigation: A new reactive obstacle avoidance approach for highly cluttered environments, Robot. Auton. Syst. 84 (2016) 15–30.
[26] Y. Koren, J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, in: International Conference on Robotics and Automation, California, 1991, pp. 1398–1404.
[27] D. Guo, H. Wang, K.K. Leang, Nonlinear vision-based observer for visual servo control of an aerial robot in global positioning system denied environments, J. Mech. Robot. 10 (6) (2018) 061018.
[28] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, O. Von Stryk, Comprehensive simulation of quadrotor UAVs using ROS and Gazebo, in: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Tsukuba, Japan, 2012, pp. 400–411.
[29] N. Koenig, A. Howard, Design and use paradigms for Gazebo, an open-source multi-robot simulator, IEEE/RSJ Int. Conf. Intell. Robot. Syst. 3 (2004) 2149–2154.

Jake A. Steiner received his B.S. in mechanical engineering from the California State University Maritime Academy in 2016. He is currently a Ph.D. candidate working in the Design Automation Robotics and Control (DARC) lab at the University of Utah in Salt Lake City, Utah. His research interests include motion planning, artificial intelligence, and soft robotics.



Xiang He is currently a Ph.D. candidate working in the Design Automation Robotics and Control (DARC) lab at the University of Utah in Salt Lake City, Utah. He received his B.S. and M.S. in Automation Science and Electrical Engineering from the Beihang University in 2012 and 2015. His research interests include robotics, control theory, motion planning, multi-rotor ground effect and environmental sensing.

Joseph R. Bourne received his B.S. in mechanical engineering from the University of Utah in 2015. He is currently a Ph.D. candidate working in the Design, Automation, Robotics & Control (DARC) Lab at the University of Utah in Salt Lake City, Utah. He is also affiliated with the University of Utah Robotics Center. His research interests include coordinated motion planning, control, Bayesian estimation, and infotaxis with application in autonomous mobile robots for environmental monitoring.

Kam K. Leang received the B.S. and M.S. degrees in mechanical engineering from the University of Utah, Salt Lake City, in 1997 and 1999, respectively, and the Ph.D. degree from the University of Washington, Seattle, in December 2004. He is an Associate Professor in the Mechanical Engineering Department at the University of Utah, where he joined in July 2014 and directs the DARC (Design, Automation, Robotics and Control) Lab. He is also a member of the University of Utah Robotics Center. Between 2008 and 2014, he was at the University of Nevada, Reno. His current research interests focus on three main areas: (1) modeling and control for highspeed nanopositioning and scanning probe microscopy, (2) modeling, control, and manufacturing of electroactive polymers for applications in soft robotics, and (3) design, control, and motion planning of mobile robotic systems, including aerial robotic systems. He currently serves as an Associate Editor for Mechatronics (Elsevier), the International Journal of Intelligent Robotics and Applications (IJIRA), and Frontiers in Mechanical Engineering (Nature Publishing). He has been involved with conference organization and editorialship, including the American Control Conference (ACC), IEEE International Conference on Robotics and Automation (ICRA), and IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). He is also a member of the ASME.