Collision Avoidance for an Unmanned Aerial Vehicle in the Presence of Static and Moving Obstacles

Andrei Marchidan [*] and Efstathios Bakolas [†]
The University of Texas at Austin, Austin, TX, 78712

A new collision avoidance procedure for an unmanned aerial vehicle in the presence of static and moving obstacles is presented. The proposed procedure is based on a new form of local parametrized guidance vector fields, called collision avoidance vector fields, that produce smooth and intuitive maneuvers around obstacles. These vector fields are generated from a decomposition of UAV kinematics and a proximity-based velocity modulation. The proposed kinematic decomposition encodes both collision avoidance and constant speed motion for the UAV. As such, the resulting maneuvers follow nominal collision-free paths, which we refer to as streamlines of the collision avoidance vector fields, with constant speed. Next, in accordance with the computed guidance vector fields, different collision avoidance controllers that generate collision-free maneuvers are developed. Furthermore, it is shown that any tracking controller with convergence guarantees can be used with the avoidance controllers to track the streamlines of the collision avoidance vector fields. Finally, numerical simulations demonstrate the efficacy of the proposed approach and its ability to avoid collisions with static and moving pop-up threats in different practical scenarios.

Nomenclature

UAV	=	unmanned aerial vehicle
$\mathbb{R}^n$	=	set of n-dimensional real vectors
$\mathbb{S}^1$	=	unit circle homeomorphic to the closed set $[-\pi, \pi]$
$\mathbb{Z}$	=	set of integers
CAVF	=	collision avoidance vector field
$\emptyset$	=	empty set
$\langle \cdot, \cdot \rangle$	=	dot product
$\angle(\cdot, \cdot)$	=	angle between two vectors measured counterclockwise or clockwise for positive or negative values, respectively
V	=	UAV speed

[*]PhD Candidate, Department of Aerospace Engineering and Engineering Mechanics, andrei.marchidan@utexas.edu
[†]Assistant Professor, Department of Aerospace Engineering and Engineering Mechanics, AIAA Senior Member, bakolas@austin.utexas.edu

x	=	UAV position on the x-axis of the inertial frame
y	=	UAV position on the y-axis of the inertial frame
$\boldsymbol{p}$	=	UAV position vector
ψ	=	UAV heading
ψ_d	=	UAV desired heading
$u(t)$	=	UAV steering control input at time t
$\xi_{[t_i,T]}$	=	UAV position trajectory at time interval $[t_i, T]$
T	=	free final time
r_o	=	obstacle radius
r_s	=	sensing range
$\mathcal{S}(t)$	=	sensing space at time t
$\mathcal{J}(t)$	=	index set for all registered obstacles at time t
$\mathcal{O}(t)$	=	set of all inadmissible UAV positions at time t
$\mathcal{F}(t)$	=	set of all admissible UAV positions at time t
$(\cdot)^{\mathsf{C}}$	=	complement of a set operator
$h(\boldsymbol{p})$	=	collision avoidance vector field
$\boldsymbol{p}_o$	=	position of registered obstacle
x_o	=	obstacle position on the x-axis of the inertial frame
y_o	=	obstacle position on the y-axis of the inertial frame
$h_r(\boldsymbol{p})$	=	collision avoidance vector field for an obstacle in the relative obstacle frame
$\boldsymbol{e}_x$	=	x-axis unit vector
$\boldsymbol{e}_y$	=	y-axis unit vector
$\boldsymbol{e}_r$	=	unit vector for the line-of-sight to the obstacle
$\boldsymbol{e}_\theta$	=	unit vector for the obstacle tangent, perpendicular to $\boldsymbol{e}_r$
$h_s(\boldsymbol{p})$	=	collision avoidance vector field for a static obstacle
${}^{i}(\cdot)$	=	inertial frame superscript
${}^{b}(\cdot)$	=	obstacle moving frame superscript
r	=	radial distance of an agent from an obstacle's center
θ	=	angle between the x-axis and the line-of-sight between $\boldsymbol{p}$ and $\boldsymbol{p}_o$
β	=	angle between the desired trajectory direction and obstacle line-of-sight
V_o	=	obstacle speed
θ_o	=	obstacle direction with respect to x-axis

V_b	=	UAV speed in the moving obstacle frame
ψ_b	=	UAV desired heading in the moving obstacle frame
$h_d(\boldsymbol{p})$	=	collision avoidance vector field for a dynamic obstacle
ϕ	=	angle between the line-of-sight and the UAV velocity
$u_s(t)$	=	collision avoidance control input for a static obstacle
$u_d(t)$	=	collision avoidance control input for a dynamic obstacle
$u_m(t)$	=	collision avoidance control input for multiple obstacles
$u_t(t)$	=	tracking control input for multiple obstacles CAVFs
Δ	=	agent distance to an obstacle
w	=	CAVF weight associated with an obstacle
σ_Δ	=	sum of distances to each obstacle whose CAVF is active
ϵ_m	=	predefined weight threshold value

I. Introduction

Unmanned Aerial Vehicles (UAVs) have been widely used by both military and civil entities in missions ranging from surveillance to search-and-rescue to convoy protection to pay-load delivery for rescue situations or even for commercial businesses. In accomplishing these tasks, the autonomous vehicle is required to navigate without supervision in environments populated with both static and moving obstacles. Specifically, while it is following planned trajectories that are in accordance with high-level specifications, such as flying through different waypoints or maintaining a specific course given by a path planning protocol, the UAV must be able to perform maneuvers to avoid pop-up threats or obstacles that may not have been considered for the original trajectory plan. Thus, the need for decentralized, reactive, computationally inexpensive and fast collision avoidance arises.

Flight control systems for unmanned aerial vehicles are well-studied and mature methodologies that provide feedback controllers can guarantee accurate tracking of reference pose. As such, it is a common assumption in the path-planning community for UAVs to assume constant altitude operation and to approximate the system with a planar kinematic Dubins model [1]. This implies that the planning search space reduces to two geometric dimensions and that navigation depends only on speed and steering or heading control. With these assumptions, the collision avoidance problem can be tackled in different ways, depending on additional mission requirements.

Some of the more notable early works tackled the collision avoidance problem by creating a graph in the autonomous agent's free configuration space, considering only the geometric requirement of finding an obstacle-free path between two points. Then, obstacle-free paths are found by applying different search algorithms that attempt to connect graph nodes while, at the same time, may optimize different metrics. Some of the most used algorithms are Dijkstra's [2], A* [3, 4] and D* or D*-lite algorithms [5]. The configuration space graphs are usually created using either basic sampling techniques that depend on feature density or more complex techniques that employ more advanced space discretizations: i.e., decomposition into cells [6], Voronoi diagrams [7], projections [8] or retractions [9]. Graph-based search algorithms, however, rely on search space granularity for speed and accuracy and, therefore, are not very suitable for real-time applications. To speed up the process of sampling and searching for paths, a sampling-based technique that quickly generates obstacle-free paths was developed by Lavalle [10] and was later extended to more complex dynamics [11–13]. These new approaches, relying on randomly sampling the free configuration space, sped up the search algorithms, however, they require both re-planning in the presence of moving obstacles and a large number of samples and collision checks for very cluttered environments, making them computationally costly.

Other popular approaches for collision avoidance rely on curve parametrization for trajectory generation [14–16]. These geometric techniques integrate dynamic and path constraints by performing waypoint parameter optimization in order to define splines, polynomial, logistic or Bézier curves, or even clothoids, that are feasible for more complex systems to follow. Such constraints may relate to speed, path curvature, path length, obstacle avoidance, and many other mission requirements. Solving these problems, however, requires high computational resources due to their nonlinear nature, which may lead to the appearance of local minima or configurations where the autonomous agent gets stuck. Moreover, none of these parameter optimization techniques are able to account for moving obstacles without further assumptions, simplifications, or re-planning.

A similar class of algorithms used for collision avoidance are optimization-based algorithms that aim to solve nonlinear programs with different NLP (nonlinear programming) solvers [17, 18] by applying direct or indirect numerical methods. Indirect methods require deriving the necessary optimality conditions and finding the correct adjoint variables to satisfy these conditions, whereas direct methods use a discretization of the state and input variables in order to reduce the optimal control problem into an NLP problem. Both of these nonlinear programming techniques cannot guarantee convergence to a feasible solution and demand good initial guesses for their decision variables, which still makes them too computationally intensive for real-time applications.

To avoid the computational complexity of the mentioned techniques, a different approach that generates obstacle-free paths by using artificial potential fields was developed in [19]. In this method, obstacles are associated with repelling forces and target destinations are associated with attractive forces. Then, by considering these to be the only forces acting on the autonomous agent, the motion plan is generated through gradient descent on the artificial potential field. The method's simplicity comes with two drawbacks: the existence of local minima and the lack of a mechanism to handle input constraints, which may lead to infeasible commands that the agent must follow. The first problem was ammended with the introduction of navigation functions [20] and harmonic potential fields [21, 22]. However, due to the high computational cost of generating harmonic potentials and the required parameter tuning for navigation functions, these approaches are hard to implement on real-time systems and can only be applied for moving obstacles through re-planning. Borrowing further from the theory of harmonic potentials and combining it with concepts from hydrodynamics, a new class of motion planners that uses streamlines as feedback motion plans was developed [23–26]. This new approach is able to consider convex static obstacles and obstacles that move with a much lower speed than the agent speed for collision avoidance. While this method solves the problem of local minima and of the high computational costs required for generating these motion plans, it does not guarantee that the required acceleration or speed control inputs are feasible. Thus, in an effort to include kinematic constraints, new fluid motion planners were introduced in [27] and [28] for curvature and speed constraints. These new planners, however, are unable to handle moving obstacles with any guarantees.

This paper proposes a new approach that uses a parameterized vector field for generating constant speed collision avoidance maneuvers around static and moving obstacles. The idea of a parametrized vector field has been used in past works to generate guidance laws for tracking different motion patterns. The vector fields were obtained using Lyapunov functions that guarantee convergence to the desired motion plans [1, 29] or by using a parametric consistency condition for the agent nonholonomic constraints [30, 31]. In contrast, the parametrized vector fields presented herein are generated directly from the decomposition of agent kinematics into normal and tangential components with respect to the obstacle boundary. In this way, the proposed approach is more closely related to the decomposition presented in [25], where the original dynamical system is modulated. The difference is that the proposed decomposition is used to encode motion plan behaviors according to different goals, such as collision avoidance and constant speed motion, by decoupling the modulation and constraining the normal and tangential velocity components with respect to the obstacle boundary to simultaneously account for different requirements. To the best of our knowledge, our approach of generating vector fields is new, since it uses the original agent kinematics and artificially modulates velocity components with a new parametric proximity-based eigenvalue function. The parameters can be used to adjust the agent's behavior around the obstacle, by performing the transition between obstacle-free motion and collision avoidance from different distances and with different intensities. A steering controller is described for the static and moving obstacle case, respectively, and an ad-hoc algorithm is proposed for dealing with multiple obstacles. Then, depending on the obstacles' velocity and their proximity, the UAV may be required to switch between controllers due to the appearance of new obstacles along its path. As a consequence, collision avoidance may not be guaranteed since the switch may not lead to motion continuity between obstacle-free motion and collision avoidance or, in other words, may bring new initial conditions that would not correspond to the desired collision avoidance vector field. Therefore, the use of a tracking controller with proven convergence guarantees is proposed. Tracking the desired inputs leads to the desired motion plan, provided by the collision avoidance vector fields, and obstacle avoidance is guaranteed for the provided problem specifications while maintaining constant agent speed.

This paper is organized as follows. Section II defines the collision avoidance problem. Section III provides a new formulation of a guidance vector field that accomplishes collision avoidance. Section IV combines these vector fields

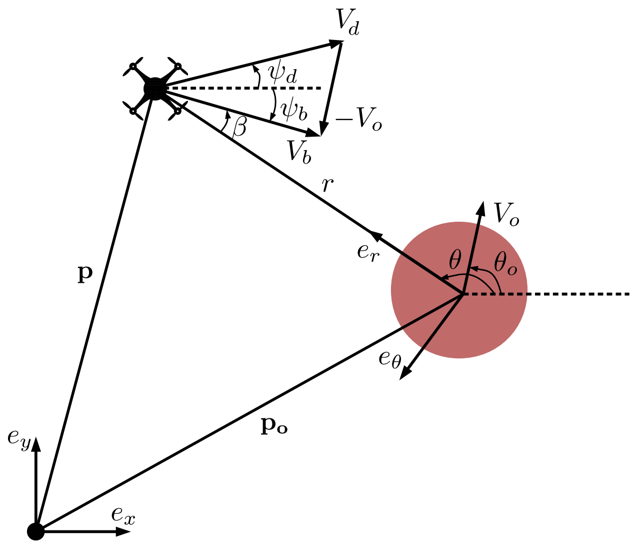


Fig. 1 Snapshot at the initial planning time t_i illustrating the coordinate systems and the state variables used in the collision avoidance problem.

appropriately in order to yield a feasible controller that satisfies the collision avoidance problem requirements. Section V illustrates and discusses numerical simulations. Finally, Section VI provides concluding remarks.

II. Formulation of the Collision Avoidance Problem

This section provides the system state space model and the mathematical formulation of the collision avoidance problem. As in previous works [1, 32], the UAV is assumed to have access to low-level control systems capable to provide attitude stabilization, altitude-hold and velocity tracking accurately, such that the omission of these control dynamics would not result in significant performance errors. Therefore, consider a UAV modeled as a point-mass system flying in a horizontal plane with constant forward speed V and direction ψ, relative to the x-axis of the inertial frame of motion which is fixed to the plane, as illustrated in Figure 1. The UAV's full state is expressed by its position $\boldsymbol{p} := [x, y]^T \in \mathbb{R}^2$ and orientation $\psi \in \mathbb{S}^1$. The equations of motion are given by the following kinematic model:

$$\begin{aligned} \dot{x}(t) &= V\cos\psi(t), \qquad & x(t_i) &= x_i, \\ \dot{y}(t) &= V\sin\psi(t), \qquad & y(t_i) &= y_i, \\ \dot{\psi}(t) &= u(t), \qquad & \psi(t_i) &= \psi_i, \end{aligned} \tag{1}$$

where u is the steering rate input and $[x_i, y_i]^T \in \mathbb{R}^2$ and $\psi_i \in \mathbb{S}^1$ are the vehicle position and heading, respectively, at the initial planning time t_i. The trajectory of the UAV obtained by integrating system (1) forward in time is denoted by $\xi_{[t_i,T]} : [t_i, T] \to \mathbb{R}^2$, where $T > t_i$ is the free final time and $\xi_{[t_i,T]} := \{\boldsymbol{p}(t) : t \in [t_i, T]\}$.

Further, assume that the UAV is initially traveling at constant speed according to a high-level planning module that provides a constant desired steering angle, denoted by $\psi_d \in [-\pi, \pi]$, and therefore $\psi_i = \psi_d$. This assumption implies only that, for the planning period, the UAV's goal is to maintain the same heading and any deviation from a predefined path would be reduced by the high-level planner, which is outside the scope of this paper.

The UAV is moving in an environment that may be populated with static and moving obstacles that can represent no-fly zones, other UAVs, airplanes, or physical obstacles. Thus, for planning purposes, assume that each obstacle is bounded by a circular region of a specific radius $r_o > 0$, determined by the largest obstacle dimension. If the user desires to incorporate vehicle size and shape into the planning procedure, the collision avoidance maneuvers may be planned in the obstacle configuration space, as defined in [33], by using augmented obstacle boundaries and the same point-mass model for the UAV. Moreover, assume that the UAV has a limited sensing range modeled as a circular region of radius r_s, determined by its sensor capabilities, centered at the UAV's geometric center. As such, the UAV sensing region at time t may be modeled in accordance with its sensing range:

$$\mathcal{S}(t) = \{\mathbf{p} \in \mathbb{R}^2 : \|\boldsymbol{p}(t) - \mathbf{p}\| \le r_s\}, \tag{2}$$

Due to the sensor limitations, the obstacles may be viewed as pop-up motion constraints, since they are registered by the UAV only when they enter the circular sensing region. Let $\mathcal{J}(t) = \{1, 2, \dots, n(t)\}$ be the index set for all registered obstacles at time t, where $n(t)$ is the total number of obstacles found. Then, the set of all inadmissible UAV positions is defined by:

$$\mathcal{O}(t) = \{\mathbf{p} \in \mathbb{R}^2 : \|\boldsymbol{p}_o^j(t) - \mathbf{p}\| \le r_o^j, \ \forall j \in \mathcal{J}(t)\}, \tag{3}$$

where $\boldsymbol{p}_o^j(t) := [x_o^j(t), y_o^j(t)]^T \in \mathcal{S}(t)$ is the position of obstacle j inside the sensing region and r_o^j is the obstacle radius. Therefore, the UAV motion is constrained to the free space inside the sensing region $\mathcal{S}(t)$, which is defined as the set of all admissible UAV positions, i.e.,

$$\mathcal{F}(t) = \mathcal{S}(t) \cap \mathcal{O}(t)^{\mathsf{C}}. \tag{4}$$

The sensing region and the admissible and inadmissible sets are illustrated in Figure 2. The collision avoidance problem is stated as follows.

Problem 1. Consider a UAV whose equations of motion are given in (1), with a sensing radius r_s, moving through an environment with static and moving obstacles, such that $\mathcal{O}(t) \neq \emptyset$. Find the steering rate input $u(t)$, such that the trajectory generated by (1), $\xi_{[t_i,T]}$, avoids collisions with any registered obstacles, while, at the same time, maintains

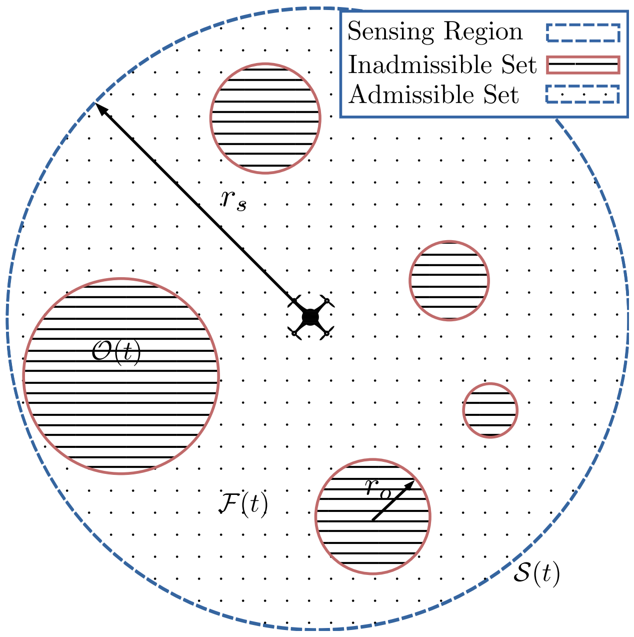


Fig. 2 Diagram illustrating the spaces involved in the collision avoidance problem.

the same course that it had before initiating the collision avoidance procedure, i.e.,

$$\xi_{[t_i,T]} \in \mathcal{F}(t), \ \forall t \in [t_i, T] \text{ and } \psi(T) = \psi_d.$$

To solve this problem, a novel approach that determines inputs through a special class of vector fields that ensure collision avoidance and their corresponding controllers is proposed in the following sections. Note that the approach described in Section III may be used for a different system model and as such will be presented generally for any autonomous agent.

III. Collision Avoidance Vector Fields

Consider an autonomous agent moving according to system (1) around an obstacle of radius r_o. Since collision avoidance requires a study of the relative motion between the agent and the obstacle, two coordinate frames are considered: one inertial frame, fixed to a point in space and determined by the unit vectors $\boldsymbol{e}_x$ and $\boldsymbol{e}_y$; and one moving frame, fixed at the obstacle geometric center and determined by unit vectors $\boldsymbol{e}_r$ and $\boldsymbol{e}_\theta$, as illustrated in Figure 1. Note, that in the moving frame, the direction of $\boldsymbol{e}_r$ is identical with the line-of-sight direction between the obstacle and the agent.

To avoid collision with an obstacle, the agent's trajectory must not penetrate its boundary at any point in time. This requirement is enforced by ensuring that the agent's velocity along the line-of-sight to the obstacle is non-negative at the obstacle boundary. A guidance law that achieves this behavior is developed using a vector field with radial velocities that are non-negative at the obstacle boundary. The concept of a collision avoidance vector field (CAVF) is defined next.

Definition III.1. The collision avoidance vector field of a circular obstacle of radius r_o centered at $\boldsymbol{p}_o \in \mathbb{R}^2$ and moving with velocity $\boldsymbol{V}_o$ is a spatially dependent vector field $h(\cdot) : \mathbb{R}^2 \to \mathbb{R}^2$ with the property that

$$\exists\, \alpha \geq 0 \text{ such that } \langle h(\boldsymbol{p}_r), \boldsymbol{e}_r \rangle \geq \alpha \geq \langle \boldsymbol{V}_o, \boldsymbol{e}_r \rangle,\ \forall \boldsymbol{p}_r \in \mathbb{R}^2 \text{ for which } \|\boldsymbol{p}_r - \boldsymbol{p}_o\| = r_o. \tag{5}$$

Any agent whose equations of motions determine a vector field that matches the collision avoidance vector field given in Definition III.1 will move away from the obstacle boundary in an attempt to avoid collisions with the particular obstacle.

A. Local Collision Avoidance Vector Field for a Single Obstacle

This section proposes an approach for generating parametrized collision avoidance vector fields around one static obstacle when the agent is supposed to move with constant speed, V, along a constant direction, determined by ψ_d. Consider a static obstacle of radius r_o located at $\boldsymbol{p}_o = [x_o, y_o]^T \in \mathcal{S}(t)$. Let $h_r(\boldsymbol{p}) = [\dot{r}, r\dot{\theta}]^T$ be a spatially dependent vector field in the relative frame, where $\boldsymbol{p} \in \mathbb{R}^2$ is a point in space, $r = \|\boldsymbol{p} - \boldsymbol{p}_o\|$, and $\theta \in \mathbb{S}^1$ is the angle that the line-of-sight between points $\boldsymbol{p}$ and $\boldsymbol{p}_o$ makes with the inertial x-axis of a coordinate system centered at $\boldsymbol{p}_o$, as illustrated in Figure 1. Since the obstacle is stationary, collision avoidance is achieved if (5) is satisfied for $\boldsymbol{V}_o = 0$, which implies that $\dot{r} \geq 0$ at the obstacle boundary $r = r_o$. Therefore, consider the following system:

$$\begin{aligned} \dot{r} &= -\lambda(r,\theta) V \cos\beta, \\ \dot{\theta} &= -\operatorname{sgn}(\sin\beta)\frac{1}{r}\sqrt{V^2 - \dot{r}^2}, \end{aligned} \tag{6}$$

where V is the agent's speed and $\lambda(\cdot,\cdot) : [r_o, \infty] \times \mathbb{R} \to [0,1]$ is a continuous function with the property that $\lambda(r_i, \theta) = 1$ and $\lambda(r_o, \theta) \geq 0$, $\forall \theta \in \mathbb{S}^1$. Note that, since the obstacle is not moving, $\boldsymbol{V}_o = 0$ and $\boldsymbol{V}_b$ is identical to $\boldsymbol{V}$, as seen in Figure 1 $\beta = \angle([\cos\psi_d, \sin\psi_d]^T, -\boldsymbol{e}_r) \in [-\pi, \pi]$ is the angle between the desired trajectory direction in the inertial frame ψ_d and the line-of-sight to the obstacle. The collision avoidance vector field for a static obstacle in the inertial frame is denoted by $h_s(\boldsymbol{p})$ and defined through the following frame transformation of the relative collision avoidance vector field:

$$h_s(\boldsymbol{p}) = \boldsymbol{R}(-\theta) h_r(\boldsymbol{p}), \tag{7}$$

where $\boldsymbol{R}(-\theta)$ is the standard rotation matrix:

$$\boldsymbol{R}(-\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{8}$$

Next, define $\lambda(r,\theta)$ as a velocity modulation function that is described by the continuous function (9), i.e.,

$$\lambda(r,\theta) = \begin{cases} -\frac{2}{\pi}\Big(\vartheta(\theta,\psi_d)(1-\gamma(r)) - \frac{\pi}{2}\Big), & \text{if } r_o \le r \le r_i,\ \ \vartheta(\theta,\psi_d) \in (0,\pi/2] \\ \gamma(r), & \text{if } r_o \le r \le r_i,\ \ \vartheta(\theta,\psi_d) \in (\pi/2,\pi] \cup [-\pi,-\pi/2) \\ +\frac{2}{\pi}\Big(\vartheta(\theta,\psi_d)(1-\gamma(r)) + \frac{\pi}{2}\Big), & \text{if } r_o \le r \le r_i,\ \ \vartheta(\theta,\psi_d) \in [-\pi/2,0] \\ 1, & \text{if } r_i < r,\ \ \theta \in \mathbb{S}^1, \end{cases} \tag{9}$$

where

$$\gamma(r) = \frac{a\big[(r-r_i)-(r_o-r)\big]}{\sqrt{(r-r_i)^2(r_o-r)^2 + \big[2a(2r-r_i-r_o)\big]^2}} + 0.5 \tag{10}$$

and

$$\vartheta(\theta,\psi_d) = \operatorname{atan2}(\sin(\theta-\psi_d),\cos(\theta-\psi_d)) \in [-\pi,\pi]. \tag{11}$$

The form of $\gamma(r)$ was inspired by the sigmoid-type functions with the algebraic form $\frac{x}{\sqrt{1+x^2}}$, where x is replaced with a function of r that satisfies $\lim_{r\to r_o} x(r) = -\infty$ and $\lim_{r\to r_i} x(r) = \infty$. With minor modifications to this sigmoid function, the desired properties for modulating the approach velocity are obtained such that $\lambda(r_i,\theta) = 1$ and $\lambda(r_o,\theta) \ge 0,\ \forall\ \theta \in \mathbb{S}^1$.

Parameters $a > 0$ and $r_i \in (r_o, r_s)$ can be chosen such that the vector field smoothness and reactivity are influenced to achieve different motion patterns. In other words, the distance at which the CAVF becomes active is determined by r_i and the transition between obstacle-free motion and collision avoidance is performed more abruptly or gradually depending on the choice of parameter a, as seen in Figure 3. Note further that r_i defines a region of influence around the obstacle and any agent inside this region of influence is considered to perform collision avoidance maneuvers. These parameters affect the behavior of λ and consequently of the steering law. Note that r_i acts as a compressive or extensive factor of the collision avoidance trajectory, whereas a determines the behavior of a sigmoid function, similar to the hyperbolic tangent. Thus, if r_i is large, the agent has more space to steer around the obstacle and the generated maneuver is smoother than in the case of a smaller r_i. In contrast, if a is large, the agent performs a more aggressive maneuver than in the case of a smaller a. The choice of parameters is left to the user's desired application and agent behavior.

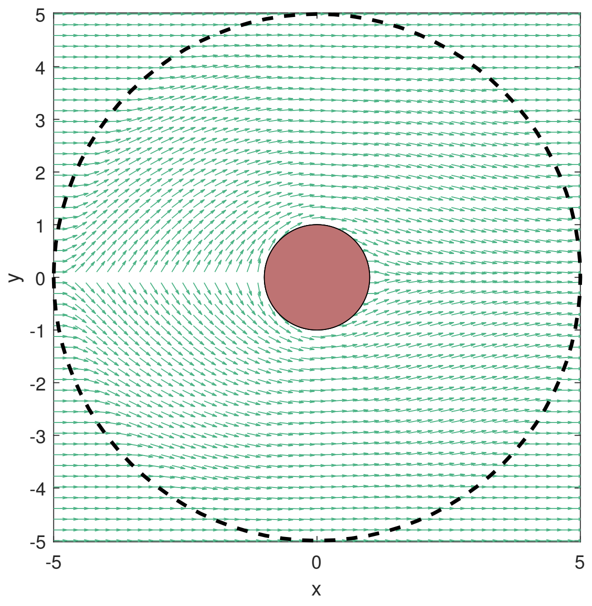
Furthermore, function λ is used to generate a vector field that becomes tangential at the obstacle's boundary and that circulates around the obstacle while trying to maintain the original heading ψ_d. As such, the provided function λ artificially scales the vector field as $r \to r_o$, since $\gamma(r_o) = 0$ and $\lambda(r_o, \theta) \geq 0$ implies that $\dot{r} \geq 0$ at $r = r_o$ whenever the agent approaches the obstacle. Therefore, $\alpha = 0$ satisfies (5) and the equations of motion given in (7) describe a CAVF. Moreover, circulation is achieved through a smooth transition, determined by λ, from obstacle-free motion to tangential motion that reverses direction as the obstacle is cleared. An obstacle is considered cleared when the agent velocity points away from the obstacle, $\langle \boldsymbol{V}, -\boldsymbol{e}_r \rangle < 0$, where $\boldsymbol{V}$ is the agent velocity vector. One sample vector field is illustrated in Figure 5, where $\psi_d = 0$.

Note that the CAVF for a single static obstacle given in (7) presents a singularity whenever $\sin\beta = 0$. This singularity defines a switching line, illustrated in Figure 5 by the line determined by $y = 0$. The switching line represents the set of agent states from which two actions may be performed to avoid collisions and, as a result, two equivalent solutions to the collision avoidance problem exist. This singularity appears due to the $\operatorname{sgn}(\cdot)$ function, which is discontinuous at 0. It will be shown in the following sections how this issue may be resolved practically. Further, note that the agent's speed does not influence the property given in (5) and, as such, may be changed throughout the agent's motion without compromising the collision avoidance properties of the system.

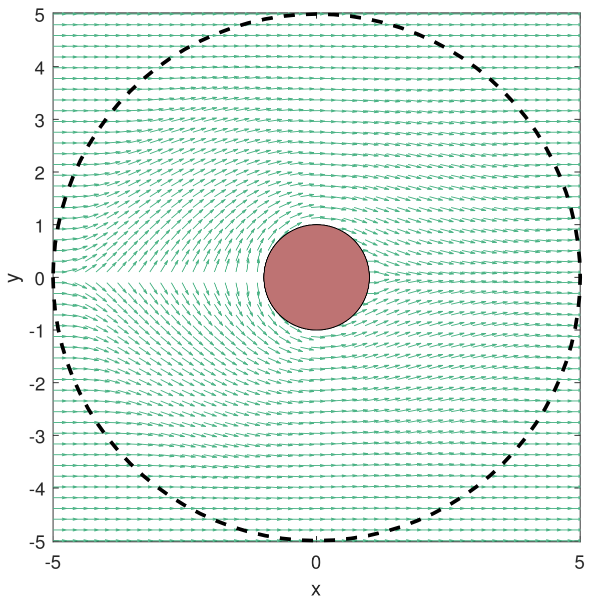
Next, the results presented for a static obstacle are extended to the case when the obstacle is moving with constant velocity, denoted by $\boldsymbol{V}_o = V_o[\cos\theta_o, \sin\theta_o]^T$, where $V_o > 0$ and $\theta_o \in \mathbb{S}^1$ are the obstacle's speed and heading angle with respect to the inertial x-axis, respectively. The constant-velocity assumption is reasonable for the problem of local collision avoidance presented in this paper, since obstacles are considered only as soon as they appear in the agent's sensing radius, which may be too restrictive for avoiding collisions. This limitation is determined by the agent's hardware and is expected in realistic situations when full knowledge of the environment is not available. Moreover, the obstacles are modeled as noncooperative agents which also supports the assumption that their speed and direction are constant for the duration of the collision avoidance maneuver.

To perform collision avoidance with a moving obstacle, the agent should not reach the obstacle's boundary with a radial speed smaller than $|\langle \boldsymbol{V}_o, \boldsymbol{e}_r \rangle|$. Therefore, we propose an approach that generates a similar parametrized CAVF around the moving obstacle, instantaneously in time, while trying to maintain as much as possible the original vector field direction, ψ_d, in the inertial frame. Since the parametrized CAVF of system (7) generates a zero radial velocity, applying this in the moving frame leads to a radial speed of $|\langle \boldsymbol{V}_o, \boldsymbol{e}_r \rangle|$ in the inertial frame, thus satisfying the requirement for collision avoidance.

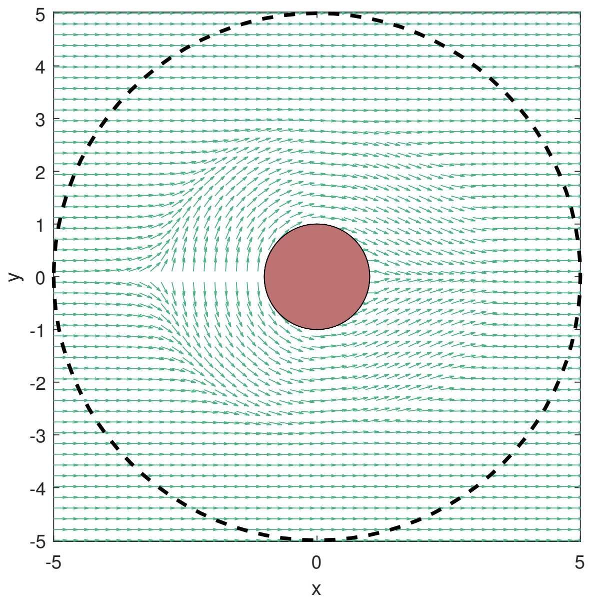
Let $\boldsymbol{V}_b$ be the agent's velocity in the relative moving frame, as illustrated in Figure 1. To generate a similar



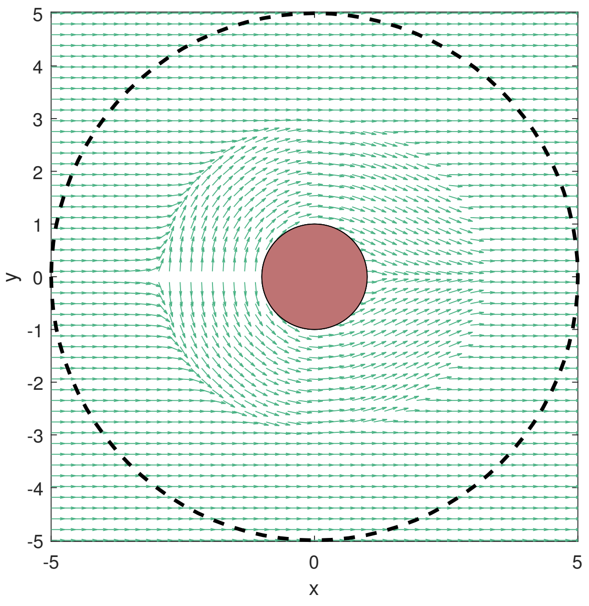
(a) $a = 0.1$



(b) $a = 0.3$



(c) $a = 3$



(d) $a = 10$

Fig. 3 Snapshots of a static obstacle CAVF, obtained for varying a values, $\psi_d = 0,\ r_o = 1,\ r_i = 5$

parametrized CAVF, ${}^b\boldsymbol{V}_b$ must have the same form as $h_s(\boldsymbol{p})$ from (7). Therefore, ${}^b\boldsymbol{V}_b = [\dot{r}, r\dot{\theta}]^T$, with

$$\begin{aligned} \dot{r} &= -\lambda(r,\theta)V_b\cos\beta \\ \dot{\theta} &= -\operatorname{sgn}(\sin\beta)\frac{1}{r}\sqrt{V_b^2 - \dot{r}^2}, \end{aligned} \tag{12}$$

where $\beta = \angle([\cos\psi_b, \sin\psi_b]^T, -\boldsymbol{e}_r) = \pi - (\theta - \psi_b)$ as seen in Figure 1. To maintain continuity between the obstacle-free

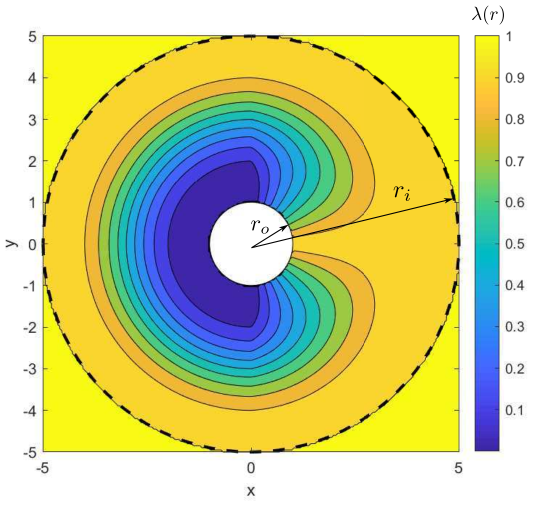


Fig. 4 Contours of $\lambda(r, \theta)$, when $a = 1$, $\psi_d = 0$, $r_o = 1$, $r_i = 5$

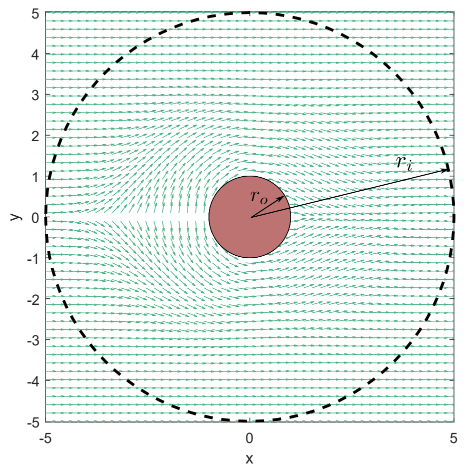


Fig. 5 Snapshot of a static obstacle CAVF, obtained using $\lambda(r, \theta)$ **with** $a = 1$, $\psi_d = 0$, $r_o = 1$, $r_i = 5$ **in Eq. 7**

vector field and the collision avoidance vector field, it is necessary to have

$$\psi_b = \operatorname{atan2}(V \sin \psi_d - V_o \sin \theta_o,\ V \cos \psi_d - V_o \cos \theta_o), \tag{13}$$

since the corresponding equations of motion to (12) in the inertial frame are obtained from ${}^i\boldsymbol{V} = {}^i\boldsymbol{V}_b + {}^i\boldsymbol{V}_o$. The relative heading ψ_b takes into account the obstacle's linear motion and preserves the agent's heading in the inertial frame at the interface between obstacle-free motion and collision avoidance.

Next assume that the agent's speed in the inertial frame is constant, denoted by V. Then, the magnitude of the relative velocity, V_b, is found by solving: $\|{}^i\boldsymbol{V}_b + {}^i\boldsymbol{V}_o\| = V$. As such, system (12) may be implemented and the CAVF for a moving obstacle, $h_d(\cdot)$, is

$$h_d(\boldsymbol{p}) = \boldsymbol{R}(-\theta)\, {}^b\boldsymbol{V}_b + {}^i\boldsymbol{V}_o, \tag{14}$$

where $\boldsymbol{R}(-\theta)$ is the standard rotation matrix, defined in (8).

If perfect tracking of the CAVF (14) is achieved, then $\dot{r} \geq 0$ in the moving frame or $\langle h_d(\boldsymbol{p}), \boldsymbol{e}_r \rangle \geq \langle \boldsymbol{V}_o, \boldsymbol{e}_r \rangle$ whenever $r = r_o$, which implies that the particle will not penetrate the obstacle boundary, thus avoiding collision. A sample vector field is illustrated in Figure 6 for a moving obstacle. The generated CAVF is rotated around the obstacle with angle ψ_b, which depends on both the obstacle's and agent's velocity, in order to allow for maneuvers that align with the obstacle motion and still return to the original heading ψ_d. Moreover, the CAVF presents strictly positive radial speed almost everywhere on the obstacle boundary to avoid the possible collision. As a result, the CAVF guides an agent found near the obstacle's boundary along its direction of motion, by steering it almost in the same direction as the obstacle's velocity to avoid the immediate collision, after which the CAVF steers the agent around the obstacle to return to the original course ψ_d.

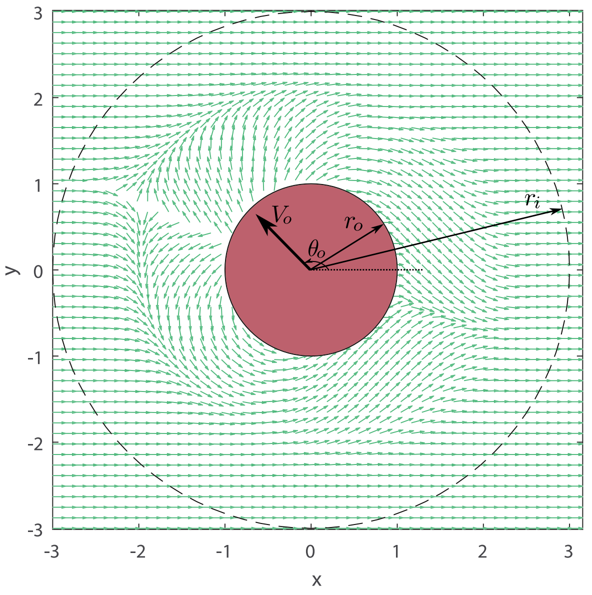


Fig. 6 An instantaneous snapshot of the moving obstacle CAVF at $t = 0$**, in the inertial frame, obtained using** $\lambda(r, \theta)$ **with** $a = 1$, $\psi_d = 0$, $r_o = 0.3$, $r_i = 4$, $V_o = 0.9$, $\theta_o = 2.35$ **in Eq.** (14)**. The thicker vector represents the obstacle velocity vector, whereas the thinner arrows show the scalar radii of the obstacle boundary and region of influence.**

The CAVF methodology is able to provide motion plans for an autonomous agent with guarantees on collision avoidance for static obstacles and obstacles moving with constant velocity. In comparison to the commonly used potential field, vector field, or navigation function techniques for collision avoidance, the CAVF is constructed directly from agent kinematics and is able to provide constant speed plans without post-processing. In contradistinction, current dynamical system approaches or vector field techniques, like the fluid-flow fields resulting from the application of the circle theorem [25, 30, 31], or simple distance-based potential functions [19], provide motion plans that may not take into account agent kinematics or speed constraints without trajectory planning or heuristics. As such, the CAVF is constructed in accordance with the aim of control, rather than only providing a collision-free geometric path.

B. Mixed Collision Avoidance Vector Fields for Multiple Obstacles

Suppose next, that multiple obstacles are identified by the agent's sensors and that there is a minimum separation distance between them denoted by δ. Furthermore, suppose that the collision avoidance parameters are chosen such that the obstacles' radii of influence lead to overlapping CAVFs. In this case, following the streamlines of one CAVF may lead to collision with other obstacles intersecting them. As such, the CAVFs must be mixed in a judicious way so that collision avoidance is still guaranteed. A mixed CAVF is now defined with respect to multiple obstacles.

Definition III.2. Suppose there are n obstacles identified by the index set $\mathcal{J}(t) = \{1, 2, \ldots, n(t)\}$ with overlapping radii of influence, where each one is located at $\boldsymbol{p}_o^j \in \mathcal{S}(t)$ and has radius $r_o^j > 0$, for $j \in \mathcal{J}(t)$ such that $\|\boldsymbol{p}_o^j - \boldsymbol{p}_o^i\| > r_o^j + r_o^i + \delta$, $\forall j, i \in \mathcal{J}(t)$, with $j \neq i$. Their mixed collision avoidance vector field is defined as a spatially dependent vector field $h(\cdot) : \mathbb{R}^2 \to \mathbb{R}^2$ with the property that for all $j \in \mathcal{J}(t)$ there exists $\alpha^j \geq 0$ such that $\langle h(\boldsymbol{p}_r), \boldsymbol{e}_r^j \rangle \geq \alpha^j$, $\forall \boldsymbol{p}_r \in \mathbb{R}^2$ that satisfies $\|\boldsymbol{p}_r - \boldsymbol{p}_o^j\| = r_o^j$.

To generate one such CAVF, a method that mixes CAVFs by computing their weighted sum, while preserving magnitude, is proposed in Algorithm 1. The weights associated with each vector field, denoted by w^j, are determined by the agent's distance to each obstacle in $\mathcal{J}(t)$, which is denoted in Algorithm 1 by Δ^j, $\forall j \in \mathcal{J}(t)$. The idea behind this proximity metric comes from the fact that the agent may have to switch between two different motion plans or follow a combination of the two: its obstacle-free motion plan or the collision avoidance maneuver proposed by an isolated CAVF, depending on how close it is to a particular obstacle boundary. As such, if the agent is outside any obstacle's radius of influence, it will continue its original motion plan without any influence from the CAVFs. As the agent enters the radius of influence of one or more obstacles and approaches the obstacles' boundaries, it must perform collision avoidance with respect to all influencing obstacles. Therefore, the proximity metric maps the agent's position into the weight range set $[0, 1]$, where a zero weight value is associated with obstacle-free motion whenever the agent is outside the obstacle radius of influence, whereas a weight value of one is associated with tangential motion around a single isolated obstacle whenever the agent is at that particular obstacle's boundary. This mapping is given in lines 2-13 of Algorithm 1. Here, we assume that obstacles cannot overlap.

Next, consider the case when the agent is closest to obstacle $k \in \mathcal{J}(t)$. Then, let the value of the weight associated with that obstacle be $w^k \geq w^j$, $\forall j \in \mathcal{J}(t)$. If this weight value is within a predefined weight threshold denoted ϵ_m, $w^k > \epsilon_m$, where $0 \ll \epsilon_m < 1$, the mixed CAVF will be identical to the CAVF of obstacle k; otherwise, all weights are normalized. This procedure is illustrated in lines 14-24 of Algorithm 1 and is required to perform the avoidance maneuver only with respect to one obstacle, since otherwise, mixing CAVFs may not guarantee collision avoidance.

Proposition III.1. Algorithm 1 generates a mixed CAVF.

Proof. Consider $n(t)$ obstacles with overlapping radii of influence determined by the index set $\mathcal{J}(t) = \{1, 2, \ldots, n(t)\}$. Suppose that $\boldsymbol{p} \in \mathcal{S}(t)$ is such that $\|\boldsymbol{p} - \boldsymbol{p}_o^j\| \leq r_i^j$, $\forall j \in \mathcal{J}(t)$. Whenever $\|\boldsymbol{p} - \boldsymbol{p}_o^k\| = r_o^k$ for some $k \in \mathcal{J}(t)$, applying Algorithm 1 will result in $w^k = 1$ and $w^j = 0$, $\forall j \neq k$, which implies that $h(\boldsymbol{p}) = h^k(\boldsymbol{p})$. Therefore, by Definition III.2, $h(\boldsymbol{p})$ is a mixed CAVF. □

The application of Algorithm 1 generates a mixed CAVF that presents n singularities or switching lines, as seen in the sample mixed CAVF illustrated in Figure 7. These switching lines suggest only that the agent does not have enough information on which direction should move, due to the symmetry of the obstacles and the existence of two feasible

Algorithm 1 Mixing Collision Avoidance Vector Fields

```
Inputs: $r$, $V$, $\psi_d$, $r_o^j$, $V_o^j$, $\theta_o^j$, $a^j$, $r_i^j$ $\forall j \in \mathcal{J}(t)$
Outputs: $w^j$, $h(\boldsymbol{p})$

 1: $\sigma_\Delta = 0$
 2: for each obstacle $j$ do
 3:     Compute $h^j(\boldsymbol{p})$ using (7) or (14)
 4:     Compute $\Delta^j = \begin{cases} r - r_o^j, & \text{if } r - r_i^j < 0 \\ -1, & \text{otherwise.} \end{cases}$
 5:     $\sigma_\Delta = \sigma_\Delta + \Delta^j(\Delta^j > 0)$
 6: for each obstacle $j$ do
 7:     if $\sigma_\Delta = \Delta^j$ then
 8:         $w^j = 1$
 9:     else
10:         if $\Delta^j > 0$ then
11:             $w^j = 1 - \Delta^j/\sigma_\Delta$
12:         else
13:             $w^j = 0$
14: $[w^k, k] = \max_j w^j$
15: $\text{sum}_w = \sum_j w^j$
16: if $w^k > \epsilon_m$ then
17:     $w^j = 0, \forall j \neq k$
18:     $w^k = 1$
19: else
20:     if $\text{sum}_w = 0$ then
21:         $w^j = 1, \forall j$
22:     else
23:         for each weight $j$ do
24:             $w^j = w^j/\text{sum}_w$
25: $h(\boldsymbol{p}) = \sum_j w^j h^j(\boldsymbol{p})$
```

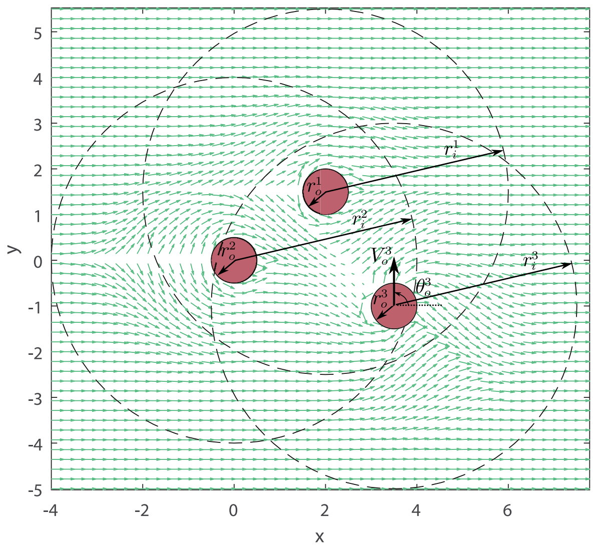


Fig. 7 An instantaneous snapshot of the mixed obstacles CAVF at $t = 0$, in the inertial frame, for two static obstacles and one moving obstacle, with the following parameters: $a = 1$, $r_o = 0.3$, $r_i = 4$, $V_o = 0.9$, $\theta_o = \pi/2$. The thicker vector represents the obstacle velocity vector, while the thinner arrows show the scalar radii of the obstacle boundary and region of influence.

collision avoidance maneuvers. In these situations, the flight control system may choose one of the two equivalent solutions to the collision avoidance problem, that is, to turn left or right, by applying a small user-defined correction that forces the agent to follow the CAVF on one side of the switching line. More details about the form of this practical control correction are provided in the next section.

Moreover, when multiple moving obstacles have overlapping radii of influence, Algorithm 1 generates a mixed CAVF that may be vanishing at certain configurations due to the vector field mixture. These configurations, however, are guaranteed to be away from the obstacle boundary by the application of the weight threshold ϵ_m which induces a single-authority region around the obstacle for its respective collision avoidance maneuvers. Furthermore, these situations are only temporary since they would result from perfectly symmetric configurations of the obstacles with respect to the agent and its obstacle-free motion plan. As such, the UAV will still be guaranteed to perform collision avoidance under these conditions.

IV. UAV Vector Field Controller

This section presents an approach for using the CAVFs from Section III to generate a solution to the collision avoidance problem described in Section II. The approach will be presented incrementally. First we demonstrate how the local CAVF for a stationary obstacle can be converted into a steering control input. Then, we show how the CAVF for a moving obstacle extends naturally the method proposed for a stationary obstacle, using the relative kinematics between the UAV and the obstacle. Lastly, we show how a simple tracking controller may be used for the mixed CAVFs while providing at the same time convergence guarantees to the desired collision avoidance maneuvers.

A. Collision Avoidance Controller for a Single Obstacle

Consider a UAV moving around a static obstacle of radius r_o, located at $\boldsymbol{p}_o = [x_o, y_o]^T$. Suppose that the UAV starts performing the collision avoidance maneuver when it is at a distance $r_i > r_o$ from the obstacle geometric center, that is, $\|\boldsymbol{p}(t_i) - \boldsymbol{p}_o\| = r_i$. Assume that the UAV's heading matches the desired heading ψ_d, if it is at a distance greater than r_i, and no steering is required, which implies $u(t) = 0$. Converting the equations of motion (1) of a UAV into polar coordinates results in the following form:

$$\begin{aligned} \dot{r}(t) &= -V\cos\phi(t), & r(t_i) &= r_i, \\ \dot{\theta}(t) &= -\frac{V}{r(t)}\sin\phi(t), & \theta(t_i) &= \operatorname{atan2}\left(y(t_i) - y_o, x(t_i) - x_o\right), \\ \dot{\phi}(t) &= u_s(t), & \phi(t_i) &= \phi_i, \end{aligned} \tag{15}$$

where $r(t) = \|\boldsymbol{p}(t) - \boldsymbol{p}_o\|$ is the relative distance between the UAV and the obstacle geometric center, $\theta(t) \in \mathbb{S}^1$, is the angle between the inertial x-axis and the line-of-sight to the obstacle, $\phi(t) \in \mathbb{S}^1$, is the angle between the line-of-sight and the current agent velocity vector, and $u_s(t) \in \mathbb{S}^1$ is the new control input. By inspecting Figure 1,

$$\beta(t) = \angle({}^i\boldsymbol{V}(t_i), -\boldsymbol{e}_r(t)),\ \forall t \in [t_i, t_f], \tag{16}$$

where $t_f > t_i$ is the time at which the UAV exits the region of influence of the considered obstacle. Therefore, the following initial condition for $\phi(t)$ is obtained: $\phi_i = \beta(t_i) = \angle({}^i\boldsymbol{V}(t_i), -\boldsymbol{e}_r(t_i))$. This initial condition guarantees continuity at the maneuver transfer between the high-level plan and the low-level avoidance of the CAVF by making sure that the collision avoidance velocity vector is aligned with the obstacle-free velocity vector. Next, to determine the control input $u_s(t)$ required to achieve the behavior of (7), consider the following relation, which results from the equivalency between system (7) and system (15):

$$V\cos\phi(t) = \lambda(r,\theta)V\cos\beta(t). \tag{17}$$

Differentiating (17) with respect to time and using the definition of β provided in (16) yields

$$\dot{\phi}(t) = \frac{\dot{\lambda}(t)\cos\left(\theta(t) - \psi_d\right) - \lambda(t)\dot{\theta}(t)\sin\left(\theta(t) - \psi_d\right)}{\sin\phi(t)}. \tag{18}$$

Here, the notation $\lambda(t)$ denotes the implicit time-dependence of λ and $\dot{\lambda}(t)$ denotes the total derivative of λ; in particular, $\lambda(t) := \lambda(r(t), \theta(t))$ and $\dot{\lambda}(t) := \frac{\partial\lambda}{\partial r}\frac{dr}{dt} + \frac{\partial\lambda}{\partial\theta}\frac{d\theta}{dt}$ and, since $\dot{\phi}(t) = u_s(t)$, the control input must be equal to the right hand side of (18). Moreover, $\dot{\phi}(t)$ is singular whenever $\sin\phi(t) = 0$, corresponding to the case when the UAV is moving along the line-of-sight to the obstacle and may avoid collision by either turning left or right. As such, the control input that guarantees collision avoidance exists but it is not unique. Hence a small user-defined correction may be performed whenever $\phi(t) = k\pi$, where $k \in \mathbb{Z}$, by replacing $1/\sin\phi(t)$ in (18) with $k_\vartheta = 1/\sin\vartheta$, where $0 < |\vartheta| \ll \pi/2$. This change corresponds to a small deviation from the current path, which triggers the collision avoidance on one side or another of the obstacle. Thus, we have the following control input for system (15):

$$u_s(t) = \begin{cases} k_\vartheta \left(\dot{\lambda}(t)\cos\left(\theta(t)-\psi_d\right) - \lambda(t)\dot{\theta}(t)\sin\left(\theta(t)-\psi_d\right)\right), & \text{if } \phi(t) = k\pi \\ \dfrac{1}{\sin\phi(t)}\left(\dot{\lambda}(t)\cos\left(\theta(t)-\psi_d\right) - \lambda(t)\dot{\theta}(t)\sin\left(\theta(t)-\psi_d\right)\right), & \text{otherwise.} \end{cases} \tag{19}$$

The control input (19) may be mapped into inertial coordinates by a simple transformation between system (1) and system (15) given by ${}^b\boldsymbol{V}(t) = \boldsymbol{R}(\theta(t))^i\boldsymbol{V}(t)$. From this transformation, we have the following relation:

$$V\cos\psi(t) = -V\cos(\phi(t)+\theta(t)),$$
$$V\sin\psi(t) = -V\sin(\phi(t)+\theta(t)),$$

from which, the inertial control input for collision avoidance is

$$u(t) = \dot{\phi}(t) + \dot{\theta}(t) = u_s(t) + \dot{\theta}(t). \tag{20}$$

Therefore, if the initial conditions of system (1) are such that (17) is satisfied, then the application of (20) generates paths that follow perfectly the streamlines of the static obstacle CAVF, as illustrated in Figure 8. The resulting trajectories show how the UAV should maneuver around the obstacle to avoid collision and return to its original heading course. Note also that the farther the agent moves away from the switching line, the less it is required to maneuver in order to avoid collision, as can be observed in the top and bottom trajectories.

Next, consider a UAV located near a moving obstacle. Suppose, as before, that the UAV starts performing the collision avoidance maneuver when it reaches a distance r_i from the obstacle geometric center. Converting system (1) into a relative polar coordinate frame yields

$$\begin{aligned} \dot{r}(t) &= -V_b(t)\cos\phi(t), & r(t_i) &= r_i, \\ \dot{\theta}(t) &= -\frac{V_b(t)}{r(t)}\sin\phi(t), & \theta(t_i) &= \operatorname{atan2}\left(y(t_i)-y_o, x(t_i)-x_o\right), \\ \dot{\phi}(t) &= u_d(t), & \phi(t_i) &= \phi_i, \end{aligned} \tag{21}$$

where $V_b(t)$ is the UAV speed in the relative frame of motion, which satisfies the following set of equations obtained

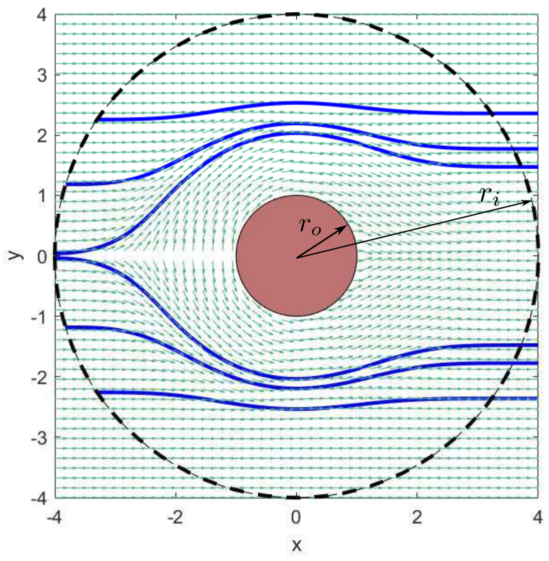


Fig. 8 System (1) **trajectories (blue) generated using** (20)**, for a CAVF (red vector field) with the following parameters** $a = 1,\ r_i = 3,\ r_o = 1,\ \psi_d = 0$**.**

from mapping system (21) into the inertial frame and equating it to system (1):

$$\begin{aligned} V\cos\psi(t) &= -V_b(t)\cos(\phi(t)+\theta(t)) + V_o\cos\theta_o, \\ V\sin\psi(t) &= -V_b(t)\sin(\phi(t)+\theta(t)) + V_o\sin\theta_o. \end{aligned} \tag{22}$$

To maintain continuity at the boundary of the CAVF, between the collision avoidance maneuver and obstacle-free motion, the UAV's velocity must satisfy (22) and, as such, the initial condition for $\phi(t)$ is

$$\phi_i = \text{atan2}\left(V_o\sin\theta_o - V\sin\psi_d,\ V_o\cos\theta_o - V\cos\psi_d\right) - \theta(t_i). \tag{23}$$

Next, to determine the control input $u_d(t)$ required to achieve similar motion patterns that CAVF (14) provides, the following relation, which results from the equivalency between system (21) and system (12), is considered:

$$V_b(t)\cos\phi(t) = \lambda(r,\theta)V_b(t)\cos\beta(t), \tag{24}$$

where $\beta(t) = \angle({}^i\boldsymbol{V}_b(t_i), -\boldsymbol{e}_r(t))$. Differentiating (24) with respect to time and using the definition of β given in (16), yields

$$\dot\phi(t) = \frac{\dot\lambda(t)\cos\left(\theta(t)-\psi_b\right) - \lambda(t)\dot\theta(t)\sin\left(\theta(t)-\psi_b\right)}{\sin\phi(t)}. \tag{25}$$

As previously noted, the right-hand side of (25) is not well-defined whenever $\phi(t) = k\pi$, for $k \in \mathbb{Z}$. Therefore, the following non-singular control input may be used to generate paths that follow the streamlines of the dynamic CAVF, as long as the initial conditions agree with (23):

$$u_d(t) = \begin{cases} k_\vartheta \left(\dot{\lambda}(t)\cos\left(\theta(t) - \psi_b\right) - \lambda(t)\dot{\theta}(t)\sin\left(\theta(t) - \psi_b\right)\right), & \text{if } \phi(t) = k\pi \\ \dfrac{1}{\sin\phi(t)}\left(\dot{\lambda}(t)\cos\left(\theta(t) - \psi_b\right) - \lambda(t)\dot{\theta}(t)\sin\left(\theta(t) - \psi_b\right)\right), & \text{otherwise.} \end{cases} \tag{26}$$

Further, to have continuity between the control input for obstacle-free motion and for collision avoidance, the agent's desired heading in the moving frame must compensate for the obstacle motion. As such, making the direction of agent's motion in the moving frame correspond to

$$\psi_b = \phi_i + \theta(t_i) = \operatorname{atan2}(V_o \sin\theta_o - V\sin\psi_d, V_o\cos\theta_o - V\cos\psi_d), \tag{27}$$

achieves the required continuity between $u_d(t)$ at $r(t) = r_i$ and $u(t)$ at $r(t) > r_i$.

As presented in the Appendix VII.A, the connection between the inertial and polar equations of motion is exploited to obtain the following relation for the inertial control input that leads to collision avoidance of moving obstacles:

$$u(t) = \dot{\psi} = (u_d(t) + \dot{\theta})\frac{V_b^2 - V_bV_o\cos(\phi + \theta - \theta_o)}{V^2} - \dot{V}_b\frac{V_o\sin(\phi + \theta - \theta_o)}{V^2}, \tag{28}$$

where

$$\dot{V}_b = \left(1 + \frac{V_o^2}{VV_b}\sin(\phi + \theta - \theta_o)\sin(\psi - \theta_o)\right)^{-1}\left(\left(u_d(t) + \dot{\theta}\right)\frac{V_o}{V}\left(V_b - V_o\cos(\phi + \theta - \theta_o)\right)\sin(\psi - \theta_o)\right). \tag{29}$$

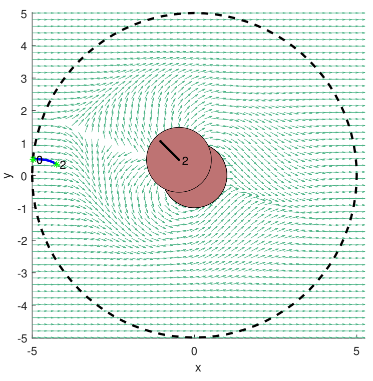
Proposition IV.1. Equation (29) is non singular if the UAV speed is greater than the obstacle speed, i.e., $V > V_o$.

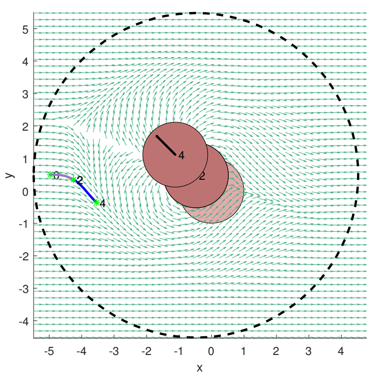
Proof. The evolution of the agent's speed in the moving frame is singular whenever the expression $1 + \frac{V_o^2}{VV_b}\sin(\phi + \theta - \theta_o)\sin(\psi - \theta_o)$ tends to 0. This expression can be equivalently written as

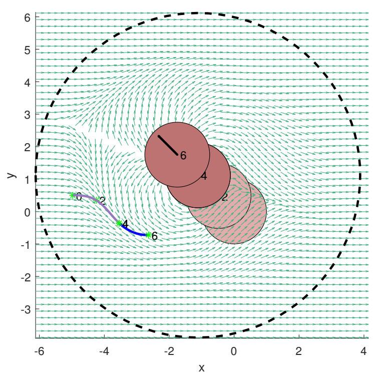
$$\left(\frac{V}{V_o}\right)\left(\frac{V_b}{V_o}\right) = -\sin(\phi + \theta - \theta_o)\sin(\theta_o - \psi). \tag{30}$$

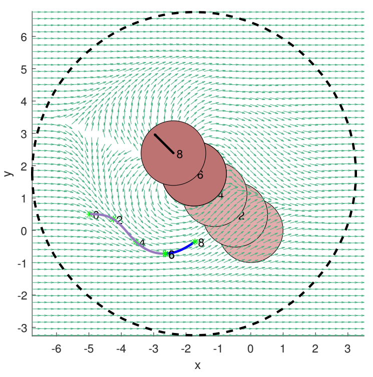
Moreover, applying the sine law in the triangle formed by the vector addition of $\boldsymbol{V_b}$ and $\boldsymbol{V_o}$, the following relation is obtained:

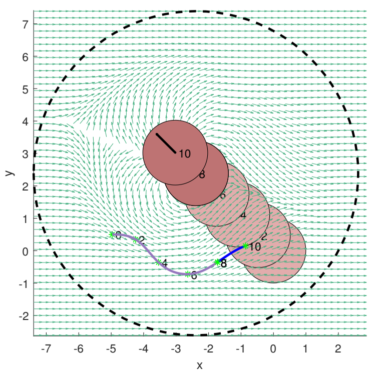
$$\frac{V_b}{\sin(\theta_o - \psi)} = \frac{V}{\sin(\phi + \theta - \theta_o)}, \tag{31}$$


(a) $t \in [0, 2]$ **sec,** $\tau = 0$ **sec**


(b) $t \in [2, 4]$ **sec,** $\tau = 2$ **sec**


(c) $t \in [4, 6]$ **sec,** $\tau = 4$ **sec**


(d) $t \in [6, 8]$ **sec,** $\tau = 6$ **sec**


(e) $t \in [8, 10]$ **sec,** $\tau = 8$ **sec**

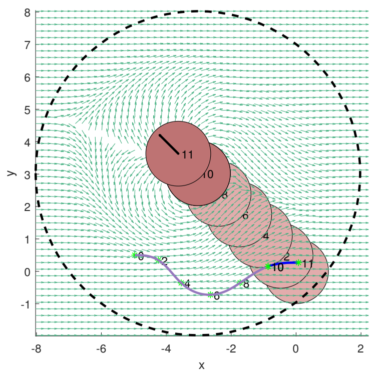

(f) $t \in [10, 11]$ **sec,** $\tau = 10$ **sec**

Fig. 9 Trajectories of system (1) **(blue) generated using input** $u(t)$ **defined in** (28)**, for a CAVF (green vector field) with the following parameters** $a = 1,\ r_i = 3,\ \psi_d = 0,\ V_o = 0.9,\ \theta_o = 2.35$**. The CAVF corresponds to the vector field generated at the given times, denoted by** τ**.**

which, in view of (30), results in a parametric condition that leads to the singularity

$$\left(\frac{V}{V_o}\right)^2 = \sin^2(\phi + \theta - \theta_o). \tag{32}$$

Condition (32) will not be true as long as $V > V_o$. □

Proposition IV.1 shows the condition under which the UAV control input (28) may be found. Therefore, if the initial conditions of system (1) are such that (24) is satisfied, then the application of (28) will generate paths that follow the streamlines of the dynamic obstacle CAVF, as illustrated in the sample simulation trials from Figure 9. It is important to note that, in the case $V \leq V_o$, the presented approach is not able to guarantee collision avoidance since there may be situations with imminent collisions. One such situation may be described by an agent-obstacle configuration where the agent is at the obstacle's boundary in the obstacle's path. In order to avoid collision, the agent would have to move along the obstacle path, however, since the agent's speed is lower than the obstacle's speed, the agent would not be able to remain outside the obstacle boundary. To remedy such situations another module would be required on top of the collision avoidance approach to identify these imminent collision zones and to artificially bloat the obstacles such that these imminent collision zones are incorporated within the obstacle boundary.

B. Collision Avoidance Controller for Multiple Obstacles

The previous controllers, given in (20) and (28), may be used in the mixing process presented in Algorithm 1 to determine a controller that follows the streamlines of the mixed CAVF, as long as the correct initial conditions are used.

Proposition IV.2. Suppose there are n obstacles identified by the index set $\mathcal{J}(t) = \{1, 2, \ldots, n(t)\}$ with overlapping radii of influence. Then, an agent following the streamlines of the corresponding mixed CAVF provided by Algorithm 1, may use the following control input:

$$u_m(t) = \sum_{j \in \mathcal{J}(t)} w^j u^j(t), \tag{33}$$

where $u^j(t)$ is the control input for the j-th obstacle, which may be of form (20) or (28).

Proof. See Appendix A. □

As noted in Section III, the mixed CAVF presents discontinuities resulting from crossing different regions of influence or the weight threshold imposed by Algorithm 1. Therefore, applying (33) to system (1) will not necessarily result in trajectories that follow the streamlines of the mixed CAVF. As such, an UAV moving around multiple obstacles will require a tracking controller that can achieve convergence to the mixed CAVF in a short amount of time. Specifically, depending on the minimum separation between obstacles, the tracking controller has to guarantee convergence to the vector field within enough time to clear the separation, free from any collision.

Consider the following tracking controller to be used in the case of a UAV moving around multiple obstacles with overlapping regions of influence:

$$u_t(t) = -K(\psi(t) - \psi_{\text{ca}}(t)) + u_m(t), \tag{34}$$

where K is a proportional gain that will be used to enforce tracking convergence, $\psi_{\text{ca}}(t)$ is the mixed CAVF heading and $u_m(t)$ is the controller defined in (33).

Proposition IV.3. For a given error tolerance e_ψ in UAV heading convergence to the mixed CAVF, where $0 < |e_\psi| \ll \pi/2$, and a minimum separation distance between obstacles $\delta > 0$, the proportional gain

$$K = \frac{2V(\log \pi - \log e_\psi)}{\delta} \tag{35}$$

guarantees that controller $u_t(t)$ defined in (34) results in CAVF tracking.

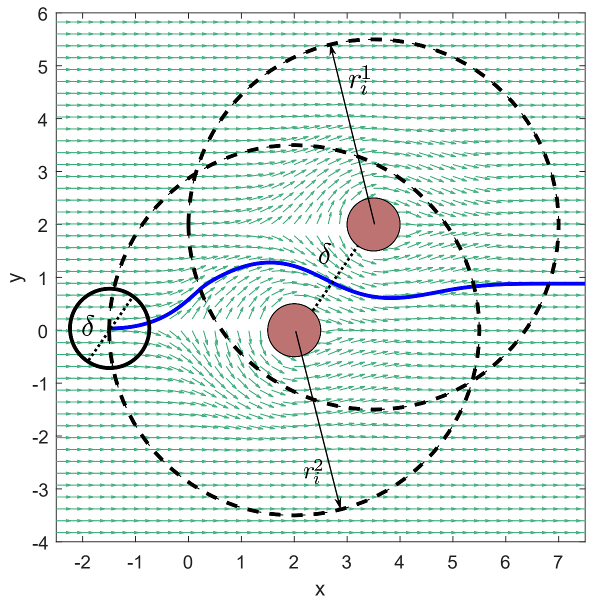


Fig. 10 Application of the tracking controller (34) **to system** (1)**, where the gain** K **depends on the separation distance,** δ**, between two static obstacles with overlapping radii of influence,** r_i^1 **and** r_i^2**.**

Proof. See Appendix VII.C. □

Proposition IV.3 relates the gain of the tracking controller with a desired heading error tolerance and a desired distance within which convergence is achieved. Therefore, picking a gain K that satisfies (47) will guarantee tracking within the given specifications.

An illustrative example is shown in Figure 10, in which two obstacles with overlapping radii of influence are considered and the proposed tracking controller is able to follow the direction imposed by the mixed CAVF obtained with Algorithm 1.

V. Simulation Results

This section uses three simulation scenarios with real world applications to analyze the performance of the proposed methodology for collision avoidance. The first scenario is a UAV moving through an environment with multiple scattered static obstacles. This scenario may represent, for example, a UAV moving at a constant height through a forest in which every tree is modeled as a static cylinder. In the second scenario, a UAV moves through an environment populated with multiple moving obstacles, similar to a busy airspace populated with other UAVs. Lastly, a UAV navigates through a complex environment, populated by both static and moving obstacles.

The UAV model used for these simulations assumes constant speed throughout all of its maneuvers. This assumption can be relaxed since the turning rate control input required to perform collision avoidance (19) and (26) is not directly dependent on the UAV's speed. Therefore, one may use a speed controller in parallel with the steering controllers presented herein to incorporate turning rate constraints. The speed controller must, however, take into account the obstacles' velocities and it must be able to provide enough actuation for collision avoidance to satisfy the requirement that the UAV's speed must be greater than the obstacles' speed, as presented in Section IV.

A. Collision Avoidance Scenario 1: Navigation through a densely populated workspace with static obstacles

Consider a UAV modeled by system (1) moving through a workspace with multiple static obstacles. In a realistic setting, for example, this workspace can represent a small forest patch that contains 12 trees, where each tree is modeled as a static circular obstacle if the UAV is moving at a constant altitude. The UAV's control objective is to pass through the workspace while maintaining an initial Eastward direction, $\psi_d = 0$, and while avoiding collisions. To this end, the proposed methodology for creating a CAVF and tracking its streamlines is applied.

The results of the simulation are illustrated in Figure 12 for the following UAV initial conditions and parameters, respectively: $r_s = 12$ m, $x_i = 0$, $y_i = 1.3$, $\psi_i = \psi_d = 0$, $V = 1$ m/s, $a = 1$ and $r_i^j = 2$ m, $\forall j \in \mathcal{J}(t) = \{1, 2, \ldots, 12\}$. The trajectory was obtained using the control input presented in (34) with a gain $K = 25$, by tracking the mixed CAVF illustrated in Figure 11. The gain was chosen such that it satisfies the condition for the minimum separation between obstacles, which in the given example is $\delta = 0.516$, and for the heading convergence error $e_\psi = 0.01$. The mixed CAVF was obtained by applying Algorithm 1. As the agent moves through the forest patch, it has to adapt to its workspace and switch from avoiding one obstacle to another. These switches can be seen by the short discontinuities in the steering control, illustrated in Figure 13 by the short abrupt changes in the required steering rates, whenever the agent enters or leaves the region determined by an obstacle's radius of influence. The controller defined in (34) is able to make the UAV's heading (illustrated by the black line in Figure 13) follow closely the mixed CAVF heading (illustrated by the green dashed line in Figure 13). Increasing gain K would result in a better convergence but more demand from the UAV's actuators, by requesting more angular speed, which would increase the spikes in $\dot{\psi}$.

B. Collision Avoidance Scenario 2: Navigation through a densely populated workspace with moving obstacles

Consider next the case when a UAV is moving through a workspace populated by multiple moving obstacles. In a realistic setting, this type of workspace could be representative for a high-traffic region of the airspace, where multiple UAVs are trying to perform cooperative or non-cooperative tasks, such as search and rescue, pay-load delivery, or surveillance. For example, suppose the workspace contains 5 other moving UAV's, where each one is modeled as a moving circular obstacle of radius $r_o^j = 0.3$ m, $\forall j \in \mathcal{J}(t) = \{1, 2, \ldots, 5\}$. Each obstacle is moving with constant velocity, at different speeds and headings. The UAV's objective is to move through this workspace while avoiding collision with any incoming obstacles and while maintaining an Eastward general heading, $\psi_d = 0$.

The CAVF for this problem is generated by applying the methodology presented in Section III for multiple moving obstacles with overlapping radii of influence. Therefore, the controller from Section IV is used to track the resulting

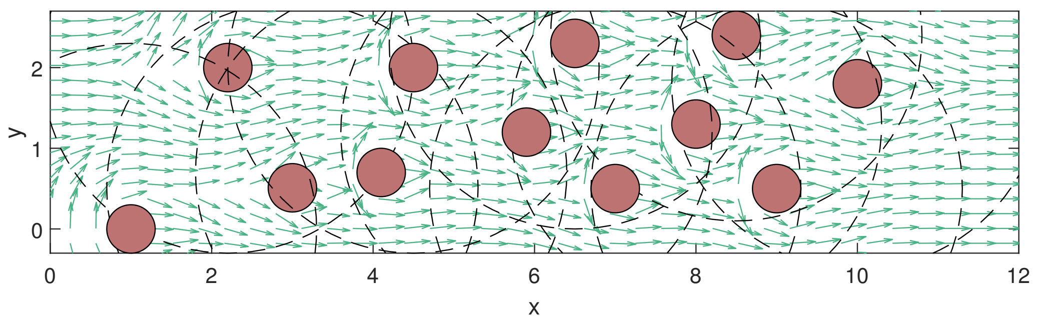


Fig. 11 The resulting mixed CAVF for the given small forest patch and collision avoidance parameters.

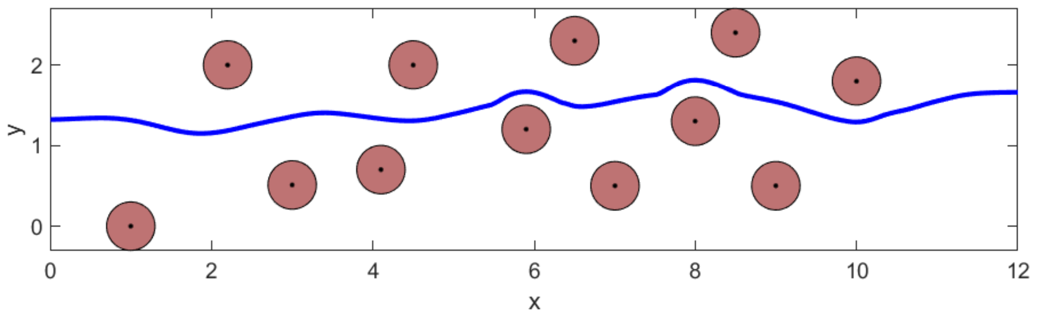


Fig. 12 Collision avoidance for a UAV (blue line) moving through a forest patch with trees modeled as circular static obstacles of radius $r_o^j = 0.3$ m, $\forall j \in \mathcal{J}(t)$.

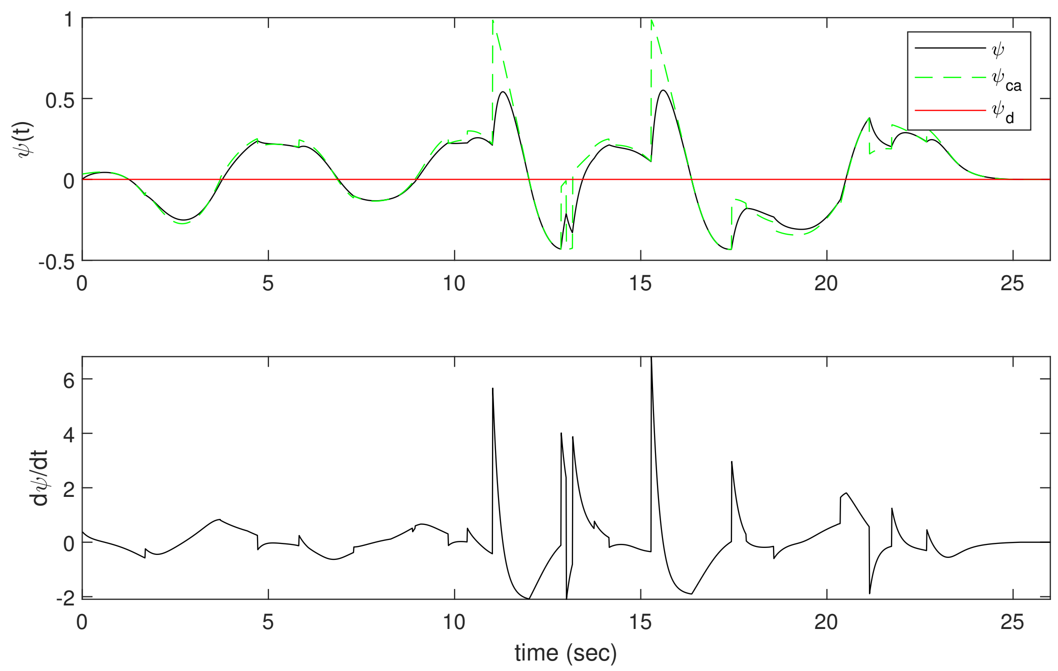


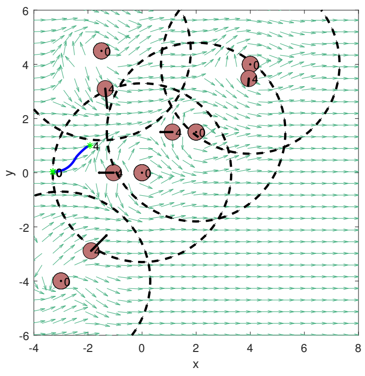
Fig. 13 Steering rate controls with resulting UAV headings.

CAVF. The results of the simulation are illustated in Figure 14 for the following UAV initial conditions and parameters, respectively: $x_i = -3.3,\ y_i = 0,\ \psi_i = \psi_d = 0,\ V = 1$ m/s, $a = 1$ and $r_i^j = 3,\ \forall j \in \mathcal{J}(t)$. The trajectory was obtained using the control input given in (34) with a time-varying gain $K(t) = 11.5\delta(t)^{-1}$, where $\delta(t) = \min_{j\in\mathcal{J}(t)} \left\| r^j(t) - r_o^j \right\|$ is the minimum distance to an obstacle at time t and where $r^j(t) = \left\| \boldsymbol{p} - \boldsymbol{p}_o^j \right\|$ is the distance between the agent's position and the obstacle j's position. The heading convergence error is set to be $e_\psi = 0.01$. The simulation shows an agent performing multiple collision avoidance maneuvers around the moving obstacles. Looking at the steering rate controls in Figure 15, note that the presented controller is able to track with accuracy the desired CAVF heading. The peaks in the profile of steering rate versus time determine different control authority switches for collision avoidance, depending on the obstacle proximities.

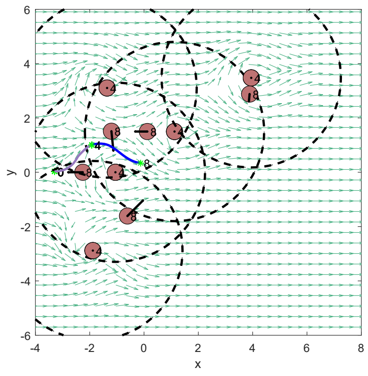
C. Collision Avoidance Scenario 3: Cluttered workspace with multiple static and moving obstacles

In the last simulation scenario, a UAV moves through a workspace that contains both static and moving obstacles. The UAV's goal is to avoid collisions with any obstacle while maintaining an Eastward direction of motion, $\psi_d = 0$. The obstacles have different radii, $r_o^j > 0$ and may move with different speeds $V_o^j \in [0, 1)$, in multiple directions $\theta_o^j \in [0, 2\pi), \forall j \in \mathcal{J}(t) = \{1, 2, \ldots, 7\}$.

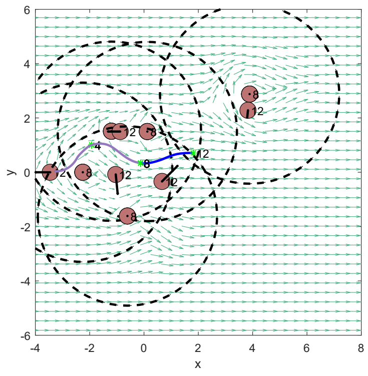
As soon as the UAV registers the obstacles, it generates a mixed CAVF using Algorithm 1. Then, applying the controller given in (34), the UAV is able to track with minimal heading error the desired vector field. The results



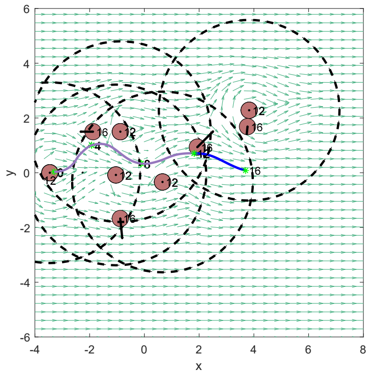
(a) $t \in [0, 4]$ **sec,** $\tau = 0$ **sec**



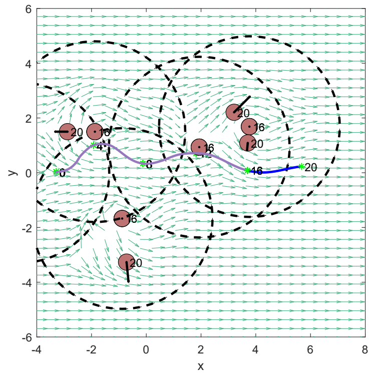
(b) $t \in [4, 8]$ **sec,** $\tau = 4$ **sec**



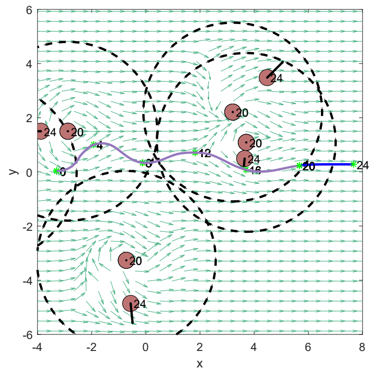
(c) $t \in [8, 12]$ **sec,** $\tau = 8$ **sec**



(d) $t \in [12, 16]$ **sec,** $\tau = 12$ **sec**



(e) $t \in [16, 20]$ **sec,** $\tau = 16$ **sec**



(f) $t \in [20, 24]$ **sec,** $\tau = 20$ **sec**

Fig. 14 Trajectories of system (1) **(blue) generated using input** $u(t)$ **defined in** (28)**, for a CAVF (green vector field) with the following parameters** $a = 1,\ r_i = 3$ m, $\psi_d = 0,\ V_o = 0.9$ m/s, $\theta_o = 2.35$ **generated at time** τ**.**

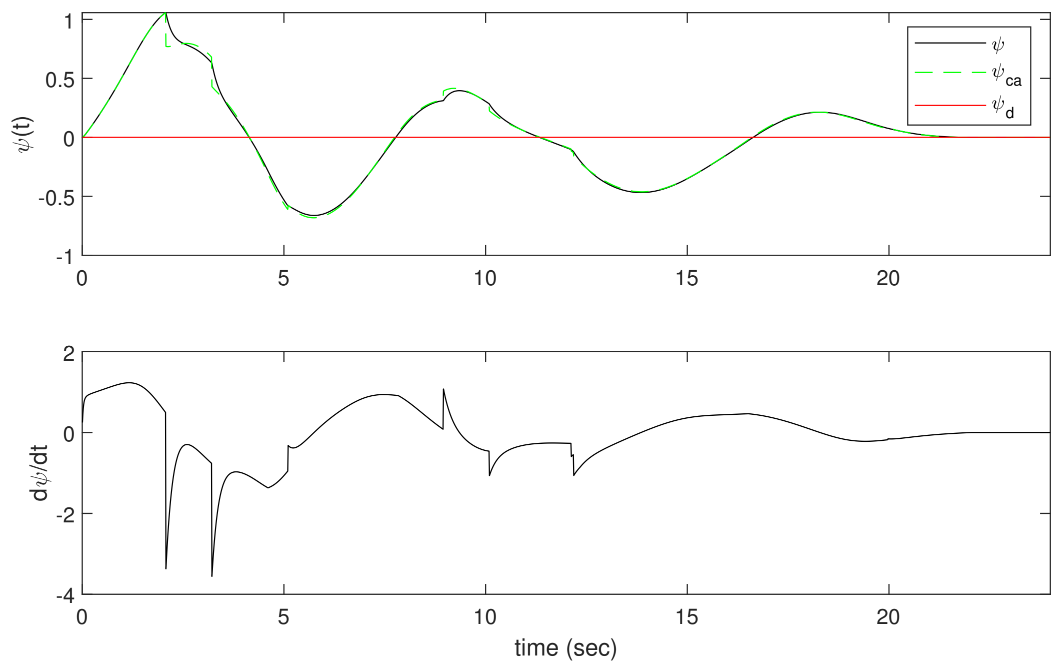


Fig. 15 Steering rate controls with resulting UAV heading versus time.

of the simulation are illustrated in Figure 16 for the following UAV initial conditions and parameters, respectively: $x_i = -3.3$, $y_i = 0$, $\psi_i = \psi_d = 0$, $V = 1$ m/s, $a = 1$ and $r_i^j = 3$, $\forall j \in \mathcal{J}(t)$. The trajectory obtained shows a more aggressive UAV behavior than in the previous simulations due to the immediate danger of colliding with the moving obstacles while intercepting static obstacles in its path. The trajectory was obtained using the control input defined in (34) with a gain $K(t)$ that depends on the minimum distance between the UAV and the detected obstacles, as defined in the previous simulation scenario. Overall, the proposed CAVF generates a guidance field that, when tracked accurately, leads to trajectories free of any collisions with the sensed obstacles.

VI. Conclusion

This paper presents a new methodology for UAV collision avoidance. The approach makes use of a new class of guidance vector fields called collision avoidance vector fields (CAVF) that are determined using a decomposition of unmanned aerial vehicle kinematics and modulation of the UAV's normal velocity component with respect to the obstacle boundary. To perform the collision avoidance maneuvers prescribed by these vector fields with constant speed, a steering law is implemented and a tracking controller guarantees convergence to the desired motion plan. Simulations performed for three scenarios illustrate the efficacy of the presented approach. In particular, the proposed approach based on CAVFs generates motion plans that take into account multiple static and moving obstacles with little computational effort, making their generation appropriate for real-time applications. Furthermore, the proposed

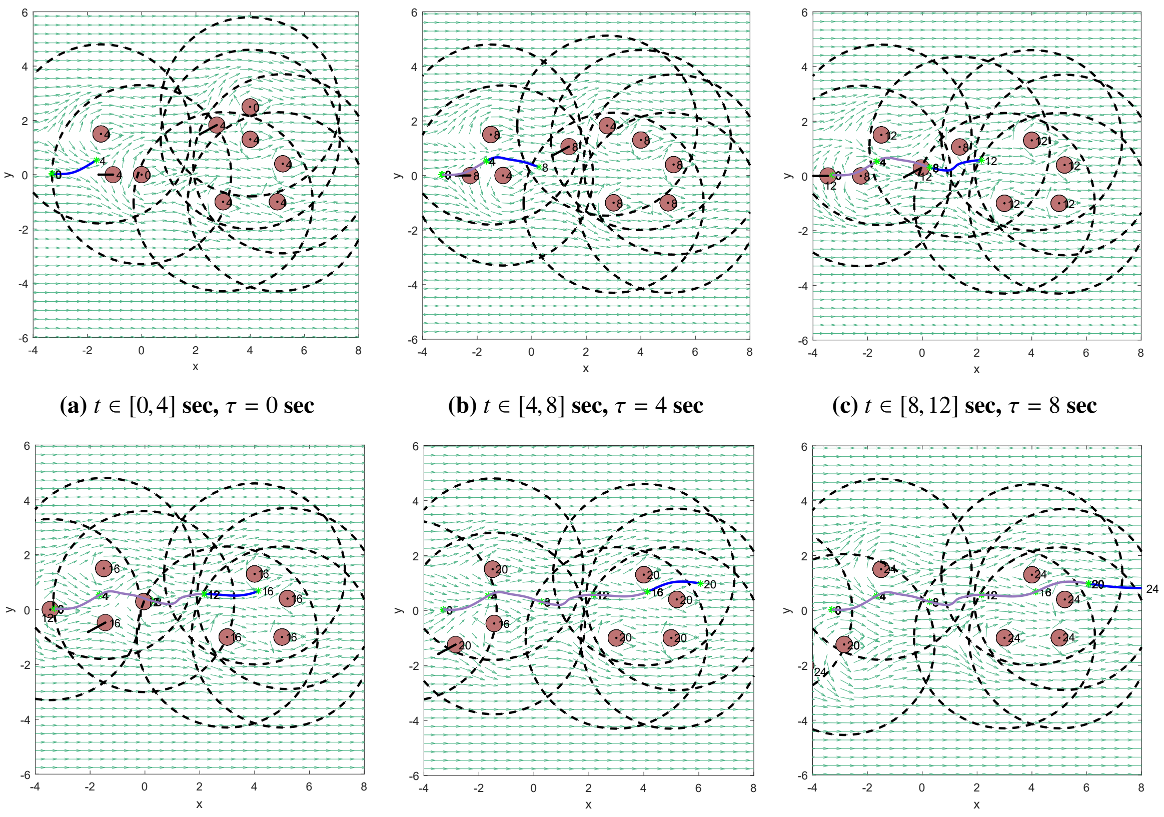

(a) $t \in [0, 4]$ **sec,** $\tau = 0$ **sec**
(b) $t \in [4, 8]$ **sec,** $\tau = 4$ **sec**
(c) $t \in [8, 12]$ **sec,** $\tau = 8$ **sec**
(d) $t \in [12, 16]$ **sec,** $\tau = 12$ **sec**
(e) $t \in [16, 20]$ **sec,** $\tau = 16$ **sec**
(f) $t \in [20, 24]$ **sec,** $\tau = 20$ **sec**

Fig. 16 System (1) **trajectories (blue) generated using** (28)**, for a CAVF (green vector field) with the following parameters** $a = 1,\ r_i = 3$ m, $\psi_d = 0,\ V_o = 0.9$ m/s, $\theta_o = 2.35$ **generated at time** τ**.**

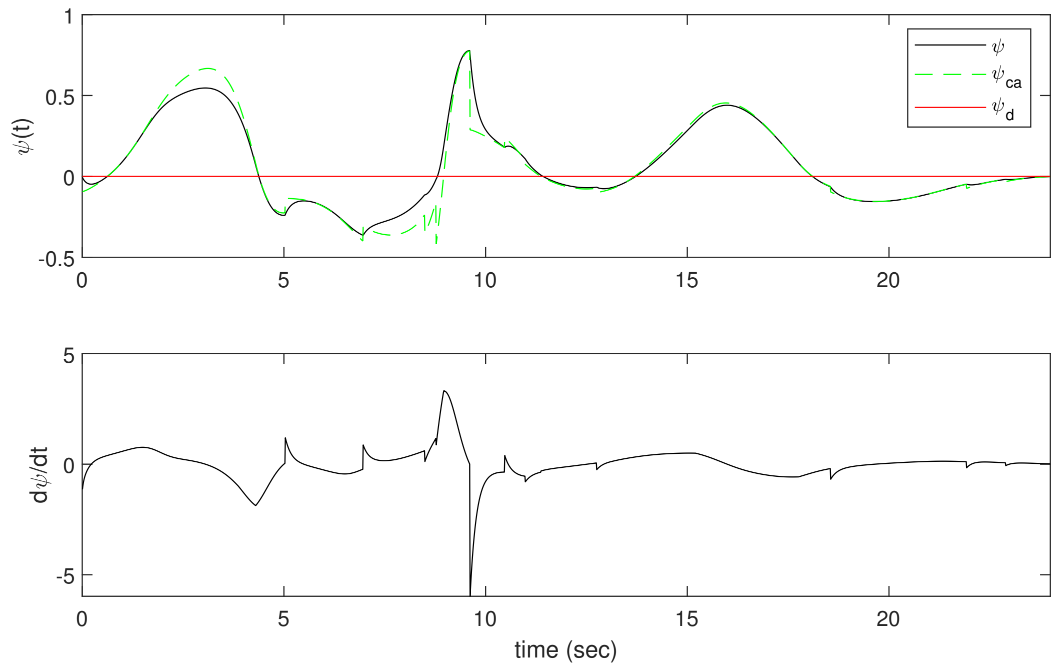


Fig. 17 Steering rate controls with resulting UAV headings.

controllers generate trajectories that follow these motion plans accurately and within specified tolerances.

Further extensions of the CAVF approach include but may not be limited to introducing disturbance models resulting from winds or modifying the approach to provide collision avoidance guarantees in the presence of obstacle state uncertainty. Another venue for extending this methodology is cooperative multi-agent path planning, in which case the assumption of constant velocity will be relaxed and more information would be required for planning.

VII. Acknowledgments

The authors would like to thank the National Science Foundation (Award #1562339) for sponsoring the research work.

Appendix

A. Moving obstacle input derivation

Consider the relation between the polar and inertial forms for the agent velocity, given in (22):

$$V\cos\psi(t) = -V_b(t)\cos(\phi(t)+\theta(t)) + V_o\cos\theta_o, \tag{36}$$

$$V\sin\psi(t) = -V_b(t)\sin(\phi(t)+\theta(t)) + V_o\sin\theta_o. \tag{37}$$

Then, using (36) and (37), the steering angle can be identified uniquely using the atan2 function, i.e.,

$$\psi(t) = \text{atan2}(-V_b(t)\sin(\phi(t)+\theta(t)) + V_o\sin\theta_o, -V_b(t)\cos(\phi(t)+\theta(t)) + V_o\cos\theta_o). \tag{38}$$

First, note that taking the sum of the squares of (36) and (37), results in the equality

$$V^2 = V_b(t)^2 + V_o^2 - 2V_b(t)V_o\cos(\phi(t)+\theta(t)-\theta_o). \tag{39}$$

Then, taking the derivative of (38) and using (39), the steering control input for avoiding a moving obstacle is obtained as given in (28).

Finally, rearranging equations (36) and (37) so that

$$V_b\cos(\phi(t)+\theta(t)) = -V\cos\psi(t) + V_o\cos\theta_o \tag{40}$$

$$V_b\sin(\phi(t)+\theta(t)) = -V\sin\psi(t) + V_o\sin\theta_o \tag{41}$$

and taking the sum of squares of the new equations yields

$$V_b(t)^2 = V^2 + V_o^2 - 2VV_o\cos(\theta_o - \psi(t)). \tag{42}$$

Then, taking the derivative of (42) and using (28), the evolution of $V_b(t)$ is obtained as in (29).

B. Proof for Proposition IV.2

Consider first a mixed CAVF, whose equations of motion are given by $[\dot{x}_m, \dot{y}_m]^T = h_m(\boldsymbol{p})$, resulting from two separate CAVFs generated by isolating each obstacle with their own equations of motion: $[\dot{x}_1, \dot{y}_1]^T = h_1(\boldsymbol{p})$ and $[\dot{x}_2, \dot{y}_2]^T = h_2(\boldsymbol{p})$. The mixed CAVF is given by applying Algorithm 1, i.e.,

$$h_m(\boldsymbol{p}) = w^1 h_1(\boldsymbol{p}) + w^2 h_2(\boldsymbol{p}), \tag{43}$$

from which we get the following relations:

$$\begin{aligned} \dot{x}_m &= w^1\dot{x}_1 + w^2\dot{x}_2, \\ \dot{y}_m &= w^1\dot{y}_1 + w^2\dot{y}_2. \end{aligned} \tag{44}$$

Moreover, it is known that each vector field has the same magnitude V and the following different headings: $\psi_m = \text{atan2}(\dot{y}_m, \dot{x}_m)$ for h_m, $\psi_1 = \text{atan2}(\dot{y}_1, \dot{x}_1)$ for h_1 and $\psi_2 = \text{atan2}(\dot{y}_2, \dot{x}_2)$ for h_2. Then, using the derivative of atan2, the following evolutions for each heading angle are obtained:

$$\begin{aligned} \dot{\psi}_1 &= (\dot{x}_1 - \dot{y}_1)/V, \\ \dot{\psi}_2 &= (\dot{x}_2 - \dot{y}_2)/V, \\ \dot{\psi}_m &= (\dot{x}_m - \dot{y}_m)/V. \end{aligned} \tag{45}$$

Next, using (44) in (45), we obtain

$$\begin{aligned} \dot{\psi}_m &= (w^1\dot{x}_1 + w^2\dot{x}_2 - w^1\dot{y}_1 - w^2\dot{y}_2)/V \\ &= w^1(\dot{x}_1 - \dot{y}_1)/V + w^2(\dot{x}_2 - \dot{y}_2)/V \\ &= w^1\dot{\psi}_1 + w^2\dot{\psi}_2, \end{aligned} \tag{46}$$

which implies that $\dot{\psi}_m = w^1\dot{\psi}_1 + w^2\dot{\psi}_2$ corresponds to the mixed CAVF for two obstacles. In the case in which there are more than two obstacles, the same approach can be carried out similarly or by induction to show that $\dot{\psi}_m = \sum_j w^j\dot{\psi}_j$. Therefore, the control input (33) corresponds to the heading evolution of a mixed CAVF. ■

C. Proof for Proposition IV.3

Let $e(t) = \psi(t) - \psi_{\text{ca}}(t)$ be the heading error between the UAV and the mixed CAVF. Then, by applying (34) to system (1), the error dynamics are $\dot{e}(t) = -Ke(t) + u_m(t) - \dot{\psi}_{\text{ca}}(t)$. Therefore $\dot{\psi}_{\text{ca}} = u_m(t)$, which results in $\dot{e}(t) = -Ke(t)$ and $e(t) = c\exp(-K(t - t_i))$, where c is a constant that depends on the initial heading error, $c = \psi(t_i) - \psi_{\text{ca}}(t_i)$.

Next, suppose that the heading error satisfies $e(t) \leq e_\psi$, $\forall t > t_{\text{track}} > t_i$, where t_{track} will be defined later. Equivalently, $c\exp(-K(t - t_i)) \leq e_\psi$, or, expressed differently, $K(t - t_i) \geq \log c - \log e_\psi$. Therefore, if K satisfies the inequality

$$K \geq \frac{\log c - \log e_\psi}{t_{\text{track}} - t_i}, \tag{47}$$

then $e(t_{\text{track}}) \leq e_\psi$, which implies that the heading error converges with the given tolerance, that is, $e(t) \leq e_\psi$ $\forall t \geq t_{\text{track}}$. Moreover, since the agent travels with constant speed, time t_{track} can be selected such that $t_{\text{track}} = \delta/(2V)$, where δ is the minimum separation distance between obstacles, assumed for mixed CAVF. Then, selecting the gain to be

$$\hat{K} = \frac{\log c - \log e_\psi}{\frac{\delta}{2V} - t_i} \tag{48}$$

guarantees convergence of the agent's heading to the mixed CAVF within a ball of radius $\delta/2$. Note that gain $\hat{K}$ depends on the initial error between the agent heading and the CAVF which is bounded from above by π. Therefore, gain (35) that depends only on the separation distance and error tolerance can be used for any situation instead of (48). ■

References

[1] Frew, E. W., Lawrence, D. A., and Morris, S., "Coordinated standoff tracking of moving targets using Lyapunov guidance vector fields," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 2, 2008, pp. 290–306.

[2] Dijkstra, E. W., "A note on two problems in connexion with graphs," *Numerische mathematik*, Vol. 1, No. 1, 1959, pp. 269–271.

[3] Hart, P. E., Nilsson, N. J., and Raphael, B., "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, 1968, pp. 100–107.

[4] Yang, H., and Zhao, Y., "Trajectory planning for autonomous aerospace vehicles amid known obstacles and conflicts," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 6, 2004, pp. 997–1008.

[5] Stentz, A., "Optimal and efficient path planning for partially known environments," *Intelligent Unmanned Ground Vehicles*, Springer, 1997, pp. 203–220.

[6] Zhu, D., and Latombe, J.-C., "New heuristic algorithms for efficient hierarchical path planning," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 1, 1991, pp. 9–20.

[7] Huttenlocher, D. P., Kedem, K., and Sharir, M., "The upper envelope of Voronoi surfaces and its applications," *Discrete & Computational Geometry*, Vol. 9, No. 3, 1993, pp. 267–291.

[8] Schwartz, J. T., and Sharir, M., "A survey of motion planning and related geometric algorithms," *Artificial Intelligence*, Vol. 37, No. 1-3, 1988, pp. 157–169.

[9] Ó'Dúnlaing, C., and Yap, C. K., "A "retraction" method for planning the motion of a disc," *Journal of Algorithms*, Vol. 6, No. 1, 1985, pp. 104–111.

[10] Kuffner, J. J., and LaValle, S. M., "RRT-connect: An efficient approach to single-query path planning," *Proceedings IEEE International Conference on Robotics and Automation*, IEEE, 2000, pp. 995–1001.

[11] Karaman, S., and Frazzoli, E., "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, Vol. 104, 2010, p. 2.

[12] Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129.

[13] Tedrake, R., Manchester, I. R., Tobenkin, M., and Roberts, J. W., "LQR-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, Vol. 29, No. 8, 2010, pp. 1038–1052.

[14] Upadhyay, S., and Ratnoo, A., "Smooth path planning for unmanned aerial vehicles with airspace restrictions," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 7, 2017, pp. 1596–1612.

[15] Mattei, M., and Blasi, L., "Smooth flight trajectory planning in the presence of no-fly zones and obstacles," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 454–462.

[16] Delingette, H., Hebert, M., and Ikeuchi, K., "Trajectory generation with curvature constraint based on energy minimization," *Proceedings IEEE/RSJ International Conference Intelligent Robots and Systems*, IEEE, 1991, pp. 206–211.

[17] Sun, C., Liu, Y.-C., Dai, R., and Grymin, D., "Two approaches for path planning of unmanned aerial vehicles with avoidance zones," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 8, 2017, pp. 2076–2083.

[18] Frazzoli, E., Mao, Z.-H., Oh, J.-H., and Feron, E., "Resolution of conflicts involving many aircraft via semidefinite programming," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 1, 2001, pp. 79–86.

[19] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots," *Autonomous Robot Vehicles*, Springer, 1986, pp. 396–404.

[20] Rimon, E., and Koditschek, D. E., "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 5, 1992, pp. 501–518.

[21] Connolly, C. I., Burns, J. B., and Weiss, R., "Path planning using Laplace's equation," *Proceedings, IEEE International Conference on Robotics and Automation*, IEEE, 1990, pp. 2102–2106.

[22] Kim, J.-O., and Khosla, P. K., "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 3, 1992, pp. 338–349.

[23] Li, Z., and Bui, T., "Robot path planning using fluid model," *Journal of Intelligent and Robotic Systems*, Vol. 21, No. 1, 1998, pp. 29–50.

[24] Waydo, S., and Murray, R. M., "Vehicle motion planning using stream functions," *Proceedings, IEEE International Conference on Robotics and Automation*, Vol. 2, IEEE, 2003, pp. 2484–2491.

[25] Khansari-Zadeh, S. M., and Billard, A., "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, Vol. 32, No. 4, 2012, pp. 433–454.

[26] Yao, P., Wang, H., and Su, Z., "UAV feasible path planning based on disturbed fluid and trajectory propagation," *Chinese Journal of Aeronautics*, Vol. 28, No. 4, 2015, pp. 1163–1177.

[27] Lau, D., Eden, J., and Oetomo, D., "Fluid motion planner for nonholonomic 3-D mobile robots with kinematic constraints," *IEEE Transactions on Robotics*, Vol. 31, No. 6, 2015, pp. 1537–1547.

[28] Owen, T., Hillier, R., and Lau, D., "Smooth path planning around elliptical obstacles using potential flow for non-holonomic robots," *Robot Soccer World Cup*, Springer, 2011, pp. 329–340.

[29] Lawrence, D. A., Frew, E. W., and Pisano, W. J., "Lyapunov vector fields for autonomous unmanned aircraft flight control," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, 2008, pp. 1220–1229.

[30] Panagou, D., Tanner, H. G., and Kyriakopoulos, K. J., "Control of nonholonomic systems using reference vector fields," *Proceedings, IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 2831–2836.

[31] Panagou, D., "Motion planning and collision avoidance using navigation vector fields," *Proceedings, IEEE International Conference on Robotics and Automation*, 2014, pp. 2513–2518.

[32] González-Arribas, D., Soler, M., and Sanjurjo-Rivo, M., "Robust aircraft trajectory planning under wind uncertainty using optimal control," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 3, 2017, pp. 673–688.

[33] LaValle, S. M., *Planning Algorithms*, Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.