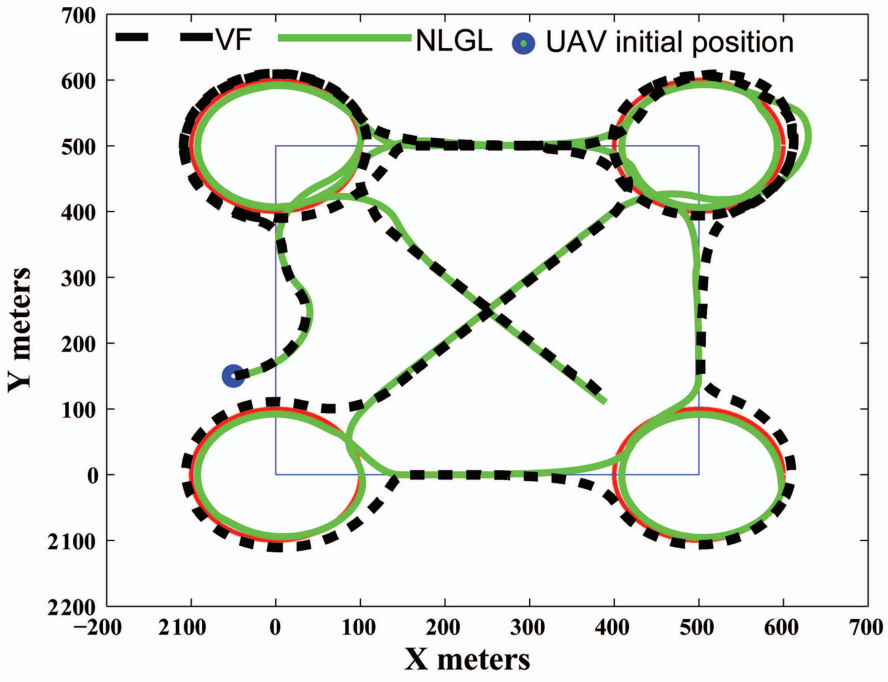
Unmanned Aerial Vehicle Path Following

A SURVEY AND ANALYSIS OF ALGORITHMS FOR FIXED-WING UNMANNED AERIAL VEHICLES



P.B. SUJIT, SRIKANTH SARIPALLI, and JOÃO BORGES SOUSA

Unmanned aerial vehicles (UAVs) are mainly used by military and government organizations, but with low-cost sensors, electronics, and airframes there is significant interest in using low-cost UAVs among aircraft hobbyists, academic researchers, and industries. Applications such as mapping, search and rescue, patrol, and surveillance require the UAV to autonomously follow a predefined path at a prescribed height. The most commonly used paths are straight lines and circular orbits. Path-following algorithms ensure that the UAV will follow a predefined path in three or two dimensions at constant height. A basic requirement

Digital Object Identifier 10.1109/MCS.2013.2287568
Date of publication: 17 January 2014

1066-033X/14/$31.00©2014IEEE

for these path-following algorithms is that they must be accurate and robust to wind disturbances.

There are several types of unmanned vehicles—UAVs, autonomous surface vehicles (ASVs), autonomous underwater vehicles (AUVs), and unmanned ground vehicles (UGVs). The principles of path following for different kinds of unmanned vehicles are similar; however, the dynamics and kinematics associated with each type of vehicle vary. Because of this variation, the corresponding constraints also vary and hence require several types of path-following algorithms. It is difficult for the UAV practitioner to select the correct algorithm because of the many available choices. To assist the practitioners, this article provides a survey of existing algorithms and presents a detailed analysis and comparison of a subset, some of which are widely used in commercial autopilots. The path-following algorithms studied in this article are two dimensional (2D) with constant altitude.

PATH-FOLLOWING PROBLEM

Given a path, the initial location of the aerial vehicle $p(x,y)$, and its heading angle ψ, the path-following problem is to determine the commanded heading angle that accurately tracks the path. Typically, most missions are either straight-line or circular-orbit paths. Straight-line paths can be defined as a set of waypoints W_i and W_{i+1} [as shown in Figure 1(a)]. The angle θ formed by this straight line with respect to the x-y coordinate frame is the line-of-sight (LOS) angle. Figure 1(b) shows the initial geometry for a circular orbit, also called "loiter following." The distance d from the UAV to the path is the "cross-track" error. Let θ_d be the desired angle of the path. In case of a straight-line path, $\theta_d = \theta$, and in case of a loiter path, $\theta_d = \theta_p$. In addition to minimizing the cross-track error, the UAV must minimize the heading error $|\theta_d - \psi|$, where $|\cdot|$ represents the absolute value. The objective of path-following algorithms is to have $d \to 0$ and $|\psi - \theta| \to 0$, as the mission time $t \to \infty$. Formally, the path-following problem using Serret-Ferret frame [1], [2], can be defined as follows.

Definition

Assume the UAV has to follow a path $P(\gamma) \in \mathbb{R}^2$, which is parameterized by $\gamma \in \mathbb{R}$.. The desired speed is $v_P(\gamma) \in \mathbb{R}$, and let $P(\gamma)$ be sufficiently smooth with bounded derivatives. The goal is to design a feedback control law such that the closed-loop signals are bounded, $\|p(t) - P(\gamma(t))\|$ converges to the neighborhood of the origin, and the velocity error $|\dot{\gamma}(t) - v_P(\gamma(t))| < \epsilon$, for $\epsilon > 0$.

CLASSIFICATION OF PATH-FOLLOWING ALGORITHMS

Path following is a basic requirement for any type of unmanned vehicle. A feedback control design for a mobile robot path-following problem is described in [3]. The path-following problem has applications in aerospace, underwater, and ground-based robots. Solution strategies for the path-following problem are either geometric or control theoretic. Several algorithms of each type are described below.

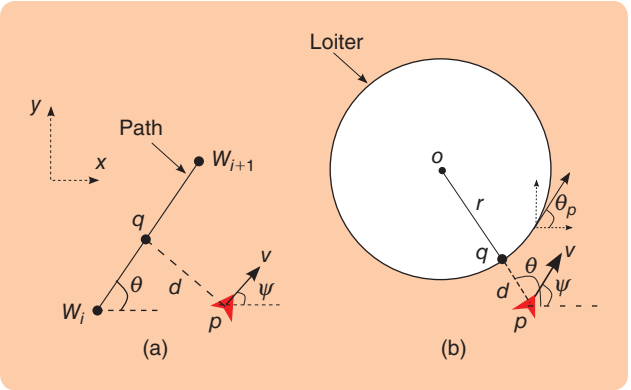


FIGURE 1 (a) The straight-line path defined by waypoints W_i and W_{i+1} that needs to be followed by the unmanned aerial vehicle (UAV) located at p. (b) The UAV located at p needs to follow the loiter circle given by a radius of r from O.

Geometric Methods

Geometric techniques, including pure pursuit [4], LOS [5]–[10], and variants of pursuit and LOS guidance laws [11], [12] are mainly found in the missile guidance literature. The path-following algorithms based on pure pursuit and LOS-based guidance laws use a virtual target point (VTP) on the path. The guidance laws direct the vehicle to chase the VTP, which eventually drives the vehicle onto the path. The distance between the VTP and the UAV position projected on the path is often called "virtual distance." The stability of LOS guidance laws for path following depends significantly on the selection of the virtual distance parameter [8]. Pure pursuit and LOS guidance laws can be combined to create a new guidance law for path following [12]. Instead of using pursuit or LOS guidance, a nonlinear guidance law (NLGL) using VTP was developed in [11]. Most of the guidance laws are developed using Dubin's car model [13].

An optimal path-following algorithm using Dubin's curves in 2D was developed in [14] and modified for three dimensional (3D) in [15]. An alternative to guidance laws is an approach based on vector fields (VFs). A VF-based path-following algorithm was developed in [16] and is intuitive and easy to implement. Since all these techniques use the geometric approach, computing the desired heading angle is quick and they are easy to implement.

Control Techniques

Control techniques, particularly nonlinear control techniques, are popular for path-following applications. They provide a certain degree of robustness to wind disturbances. A common approach used in path following is based on proportional-integral-derivative (PID) control [17] but does not perform as well as an NLGL [11]. A PID

ALGORITHM 1 Maximum lateral acceleration.

Input: u, v_g, $R_{\min}$

$$u = \begin{cases} -v_g^2/R_{\min} & \text{if } u < 0 \text{ and } |u| > v_g^2/R_{\min} \\ v_g^2/R_{\min} & \text{if } u > 0 \text{ and } |u| > v_g^2/R_{\min} \\ u & \text{otherwise} \end{cases}$$

controller with feedforward capability was developed in [18] that performed better than an NLGL.

Several types of control-theoretic techniques have been developed for path-following problems in robotics that include UAVs, AUVs, ASVs, and UGVs. Some of the well-known techniques are linear quadratic regulator (LQR) [19]–[21], sliding mode control [16], [22], model predictive control [23], [24], backstepping control [25]–[28], gain scheduling theory [29], adaptive control [30]–[32], and dynamic programming [33]. Other techniques such as piecewise affine control [34] were developed to follow a predetermined path.

Similar to VFs [16], a Lyapunov VF technique for path-following is developed in [35]. Lyapunov VFs produce globally stable tracking of loiters and are mainly used for such purposes. A modified Lyapunov VF was developed in [36] for tangent-plus-Lyapunov VF guidance and performs better than plain Lyapunov VF guidance [35].

Stability and performance guarantees are key requirements for nonlinear control laws to ensure that the vehicle tracks the path accurately under different path and environmental conditions. There are local performance and stability results for a class of path-following controllers developed from the notion of trimming trajectories [37]. The path-following error approaches zero for any general dynamical system. In [38], known performance guarantees for the path-following problem are provided. Analytical performance of a path-following controller over a trajectory-tracking controller is given in [1]. In trajectory tracking, the path is time parameterized, whereas time is not considered in path following. The trajectory-tracking controller can be reparameterized to produce a path-following controller. This reparameterizing enables the path to be tracked either as a path-following problem or as a trajectory-tracking problem.

This article focuses on path following for fixed-wing UAVs in 2D. Since there are many available techniques, the most common and widely used path-following algorithms were selected and compared on the basis of accuracy, robustness, and control effort. The algorithms compared are the carrot-chasing algorithm, NLGL [11], vector-field (VF)-based path following [16], LQR-based path following [20], and pure pursuit with LOS (PLOS)-based path following [12]. These algorithms were selected due to their simplicity, robustness, and ease of implementation. The carrot-chasing algorithm is

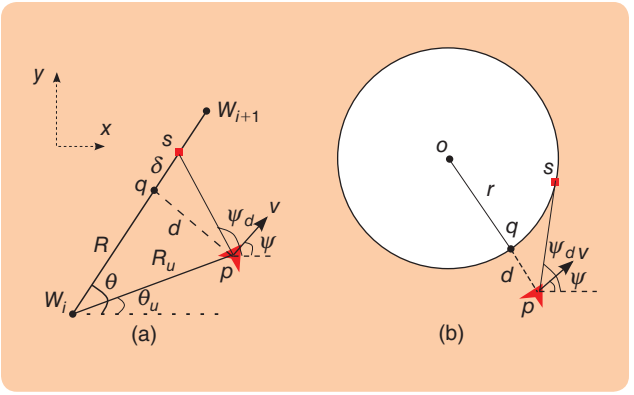


FIGURE 2 (a) The straight-line path that the unmanned aerial vehicle (UAV) has to follow and required angles that have to be computed using the carrot-following algorithm. (b) The loiter circle that the UAV has to follow and the definition of angles required to compute the commanded control using carrot-following algorithm.

the most widely used path-following algorithm. The NLGL and VF approaches are intuitive, easy to understand, and flight tested. The LQR and PLOS approaches are easy to implement.

As the UAV industry is growing rapidly, selecting good path-following algorithms will enable users to have stable platforms that meet desired performance criteria. In most commercial autopilot systems, the type of path-following algorithm that is used is not readily available. However, the VF path-following algorithm has been tested on a Kestrel autopilot [16], [39], and NLGL has been tested on a Piccolo autopilot [11], [40]. Among open-source autopilot systems, ArduPilot [41] uses VF implementation, while Paparazzi autopilot uses the carrot-chasing algorithm [42]. The analysis in this article will allow fixed-wing UAV practitioners using open-source cheap autopilots (such as ArduPilot [41], Paparazzi [42], and OpenPilot [43]) to make informed choices about the correct algorithm to implement for their particular application. Fairly evaluating these algorithms in experiments is difficult since environmental variables such as wind speed can fluctuate from one experiment to the next. By using a kinematic model and simulating various path-following strategies, the advantages and disadvantages of these algorithms are demonstrated and the effect of change in parameters is studied. From these studies, the optimal parameters for each algorithm are obtained and used in a six-degrees-of-freedom (6-DOF) simulation model with wind. Monte Carlo simulations are performed, and a comparison analysis on the performance of these algorithms using the 6-DOF model was performed. This article is an extension of the preliminary results presented in [44].

PATH-FOLLOWING ALGORITHMS

The two most commonly used paths for UAVs are straight lines and loiter paths. Both types of paths are usually defined on the x,y plane, with constant altitude and speed. Under these assumptions, the UAV kinematics are

ALGORITHM 2 Algorithm for straight-line following using the carrot-chasing algorithm.

1: Initialize: $W_i = (x_i, y_i), W_{i+1} = (x_{i+1}, y_{i+1}), p = (x, y), \psi, \delta, v_a, \kappa$

2: $R_u = \| W_i - p \|, \theta = \text{atan2}(y_{i+1} - y_i, x_{i+1} - x_i)$

3: $\theta_u = \text{atan2}(y - y_i, x - x_i), \beta = \theta - \theta_u$

4: $R = \sqrt{R_u^2 - (R_u \sin(\beta))^2}$

5: $(x_t, y_t) \leftarrow ((R + \delta)\cos\theta, (R + \delta)\sin\theta)$ % $s = (x_t, y_t)$

6: $\psi_d = \text{atan2}(y_t - y, x_t - x)$

7: $u = \text{max\_limit}(\kappa(\psi_d - \psi)v_a)$

ALGORITHM 3 Algorithm to loiter using the carrot-chasing algorithm.

1: Initialize: $O = (x_l, y_l), r, p, \psi, \lambda, v_a, \kappa$

2: $d = \| O - p \| - r$

3: $\theta = \text{atan2}(y - y_l, x - x_l)$

4: $(x_t, y_t) \leftarrow (r\cos(\theta + \lambda), r\sin(\theta + \lambda))$ % $s = (x_t, y_t)$

5: $\psi_d = \text{atan2}(y_t - y, x_t - x)$

6: $u = \text{max\_limit}(\kappa(\psi_d - \psi)v_a)$

$$
\begin{aligned}
\dot{x} &= v_g \cos\chi = v_a \cos\psi + v_w \cos\psi_w, \\
\dot{y} &= v_g \sin\chi = v_a \sin\psi + v_w \sin\psi_w, \\
\dot{\chi} &= k(\chi_d - \chi), \\
u &= \text{max\_limit}(\dot{\chi} v_g),
\end{aligned} \tag{1}
$$

where v_a is the UAV airspeed, ψ is the heading angle, v_w is the wind speed, ψ_w is the wind direction, χ_d is the desired direction toward the goal, $\chi = \text{atan2}(\dot{y}, \dot{x})$ is the course angle, $\dot{\chi}$ is the course angle rate, and the ground speed of the UAV is $v_g = \sqrt{v_{gx}^2 + v_{gy}^2}$, where $v_{gx} = v_a \cos\psi + v_w \cos\psi_w$ and $v_{gy} = v_a \sin\psi + v_w \sin\psi_w$. The lateral acceleration u is constrained using the function max_limit as given in Algorithm 1.

The kinematic model of the UAV represents a Dubins model [13]. To analyze the functioning of the algorithms and the performance under different parameter conditions, the kinematic model of the UAV without wind is considered for simplicity, which revises the kinematic equations of motion to

$$
\begin{aligned}
\dot{x} &= v_a \cos\psi, \\
\dot{y} &= v_a \sin\psi, \\
\dot{\psi} &= k(\psi_d - \psi), \\
u &= \text{max\_limit}(\dot{\psi} v_a).
\end{aligned} \tag{2}
$$

For all the simulations, the UAV speed is $v_a = 15$ m/s and the minimum turning radius is 45 m. The considered loiter circle radius is $r = 100$ m. It is assumed that the UAV autopilot has an air-speed controller that holds the air speed constant and has an altitude hold controller for constant UAV altitude. The max_limit$(\cdot)$ condition in the kinematic equations is used only in simulations to limit the maximum lateral acceleration. This condition is not used by any of the path-following guidance laws while proving the stability of the algorithms.

CARROT-CHASING ALGORITHM

One approach to follow a desired path is to introduce a VTP on the path and make the UAV chase the VTP. The UAV updates its heading direction toward the VTP. As time progresses, the UAV will move toward the path and asymptotically follow the path. In this approach, the VTP is called the carrot and the UAV chases the carrot, which explains the name "carrot-chasing" algorithm. In the missile guidance and aerospace literature, this algorithm is also called the "rabbit-chasing" algorithm.

Straight-Line Following

A straight-line path is denoted by two waypoints W_i and W_{i+1}, as shown in Figure 2(a). Following a straight line involves three steps:

1) determine the cross-track error (d)
2) update the location of the VTP
3) update ψ_d and u.

Assume the UAV is at location p with heading ψ. The projection of p onto the LOS is the point q at a distance R from W_i. This distance is the cross-track error d. The VTP $(s = (x_t, y_t))$ is located at a distance δ from q as shown in Figure 2(a). The UAV desired heading ψ_d is updated based on the location of s. This process continues until the UAV reaches W_{i+1}. The procedure to update the location of VTP and ψ_d is given in Algorithm 2.

The control input u in Algorithm 2 uses a proportional controller with gain $k > 0$. Figure 3(a) shows the UAV following the reference path for different initial heading angles of $\psi = 0, \pi/2, \pi$, and $-\pi/2$. The simulation parameters are $k = 0.5$ and $\delta = 50$.

The path parameter δ can modify the performance of the carrot-chasing algorithm as shown in Figure 3(b). For low δ (zero or ten), the UAV is forced to move directly toward the LOS. A trajectory to the path that is normal to the path will result in overshoot. A curvature constraint will then result in another trajectory normal to the path, resulting in overshoot again, etc. Hence, it takes more time to settle on the path, resulting in higher cross-track error. With an increase in δ, the UAV is able to settle on the path quickly, reducing the cross-track error. However, for even larger values of δ, the cross-track error again increases because the UAV is slow in its asymptotic approach to the path.

Loiter

In the loiter maneuver, the UAV circles around a point (O) with a standoff distance of r, as shown in Figure 2(b). The UAV performs the loiter either clockwise or counterclockwise.

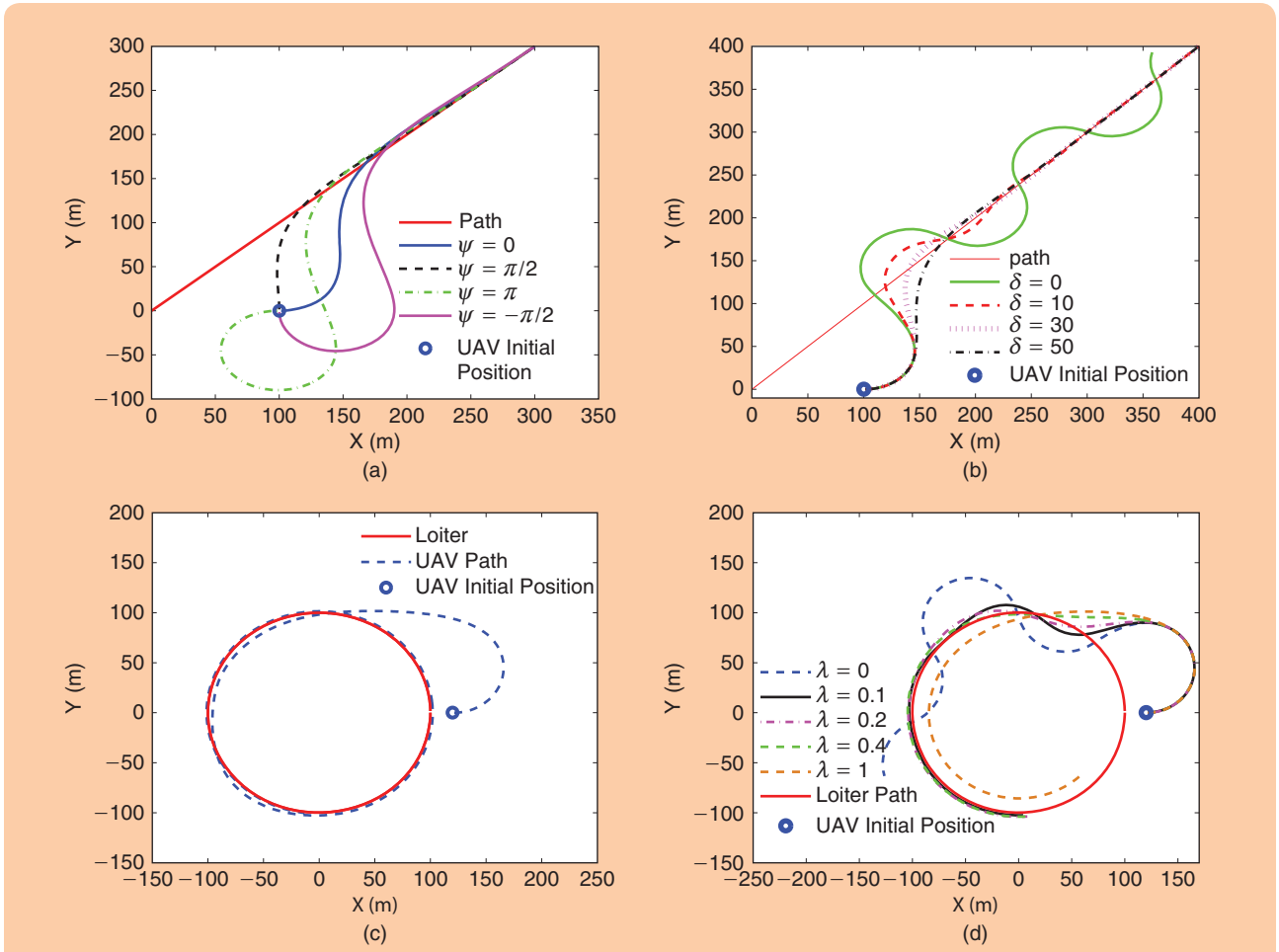


FIGURE 3 The performance of the carrot-chasing algorithm with different heading angles and δ. (a) The unmanned aerial vehicle (UAV) trajectories for the straight-line following with different initial heading angles $\psi = 0$, $\pi/2$, π, and $-\pi/2$. (b) The vehicle trajectories for straight-line following when δ is varied. (c) The UAV trajectory for loiter following with initial heading angle $\psi = 0$. (d) The vehicle trajectories for loiter following when λ is varied.

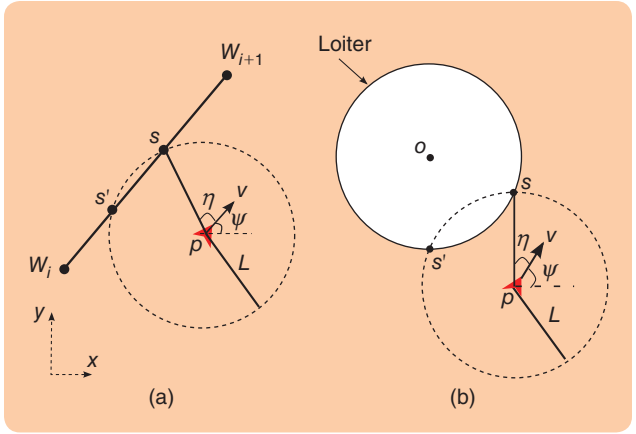


FIGURE 4 Determining the virtual target point (VTP) (s and s') for the nonlinear guidance law. The VTP for (a) the straight-line path and (b) the loiter path.

To loiter, the UAV must be located on the circle and its heading direction must be orthogonal to r. Assume the UAV is located at p. The cross-track error is $d = \|O - p\| - r$. Using the cross-track error information and the direction of the loiter, the VTP location can be generated [s in Figure 2(b)]. The procedure to update the UAV and the VTP is given in Algorithm 3.

Figure 3(c) shows the UAV following a loiter. The UAV is able to follow the loiter for different heading angles, but including them in the figure made the picture unclear and hence they were omitted.

The performance of the carrot-following algorithm depends on the value for λ. When $\lambda = 0$, the UAV is unable to track the path as the desired VTP always points orthogonal to the UAV position causing it to perform a sinusoidal path along the loiter. When λ is increased to $\lambda = 0.1, 0.2,$ and 0.4 rad, the VTP is in the pursuit range, and hence the vehicle is able to track it quickly. However, increasing λ to $\lambda = 1$ rad, the LOS angle between the UAV and VTP is significant and the desired heading angle for the vehicle points inside the loiter. Hence the UAV travels inside the

ALGORITHM 4 Algorithm for straight-line following using the NLGL algorithm.

1: Initialize: $W_i = (x_i, y_i), W_{i+1} = (x_{i+1}, y_{i+1}), p, L, v_a$
2: Determine $s = (x_t, y_t)$
3: $\theta = \text{atan2}(y_t - y, x_t - x)$
4: $\eta = \theta - \psi$
5: $u = \text{max\_limit}(2v_a^2 \sin(\eta)/L)$

loiter to pursue the VTP and the cross-track error increases significantly.

NONLINEAR GUIDANCE LAW

The NLGL also uses a VTP concept [11]. For simplicity, assume the UAV is following a path, as shown in Figure 4. At the current UAV position p, draw a circle of radius L. The circle will intercept the path at two points q and q'. Depending on the direction in which the UAV has to move, either q or q'. will be selected. The advantage of this algorithm is that the same algorithm can be applied to any type of trajectory. The stability of the guidance law is shown using Lyapunov stability arguments [11].

Straight-Line Following

Algorithm 4 gives the procedure for straight-line following using the NLGL algorithm.

The VTP location $s = (x_t, y_t)$ is directly determined from the intersection of a straight line and a circle with radius L; L is selected as a constant with the assumption that it always intersects the path. Due to this assumption, if the cross-track error is greater than L, then there will not be any intersection, resulting in no VTP. Figure 5(a) shows the ability of NLGL to track a straight-line path with different initial heading angles. For the simulations, the circle radius is $L = 100$. To know the effect of L, another simulation is done with initial heading angle of $\psi = 0$ as shown in Figure 5(b). When L does not intersect, the VTP s does not exist

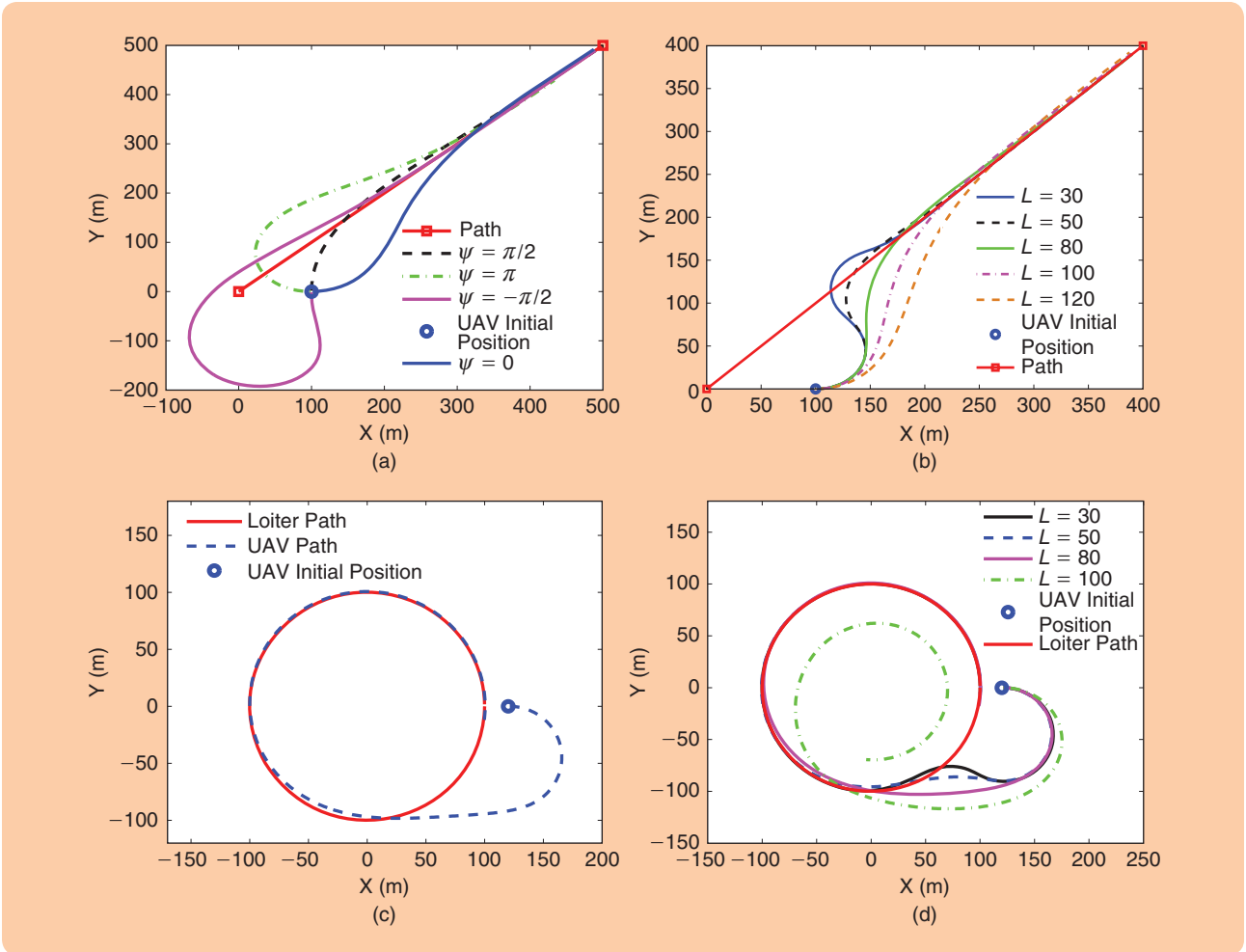


FIGURE 5 The performance of the nonlinear guidance law for different heading angles and L. (a) Unmanned aerial vehicle (UAV) trajectories for straight-line following with heading angles of $\psi = 0$, $\pi/2$, π, and $-\pi/2$. The simulations used $L = 100$. (b) The vehicle trajectories for the straight-line following when the gain L is varied. (c) UAV trajectories for loiter following with heading angle of $\psi = 0$. The simulation used $L = 80$. (d) UAV trajectories for loiter following when the gain L is varied with initial heading angle of $\psi = 0$.

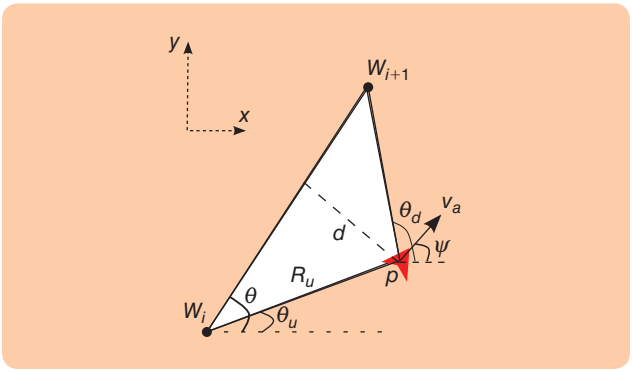


FIGURE 6 The geometry of the pure pursuit with line-of-sight path following.

ALGORITHM 5 Algorithm for straight-line following using PLOS.

1: Initialize: $W_i = (x_i, y_i), W_{i+1} = (x_{i+1}, y_{i+1}), p, \psi, k_1, k_2, v_a, \kappa$

2: $\theta_d = \text{atan2}(y_{i+1} - y, x_{i+1} - x)$

3: $d = \dfrac{(y_i - y_{i+1})x + (x_{i+1} - x_i)y + (x_i y_{i+1} - x_{i+1} y_i)}{\| W_i - W_{i+1} \|}$

4: $\psi_d = k_1(\theta_d - \psi) + k_2 d$

5: $u = \text{max\_limit}(\kappa(\psi_d - \psi)v_a)$

and $\theta = \pi/2$ in line 3 of Algorithm 4. Since $\theta = \pi/2$, the vehicle moves in the orthogonal direction toward the path, as shown in Figure 5(b). This lack of convergence to the path is an issue when implementing the NLGL algorithm with low L on embedded computers or microcontrollers that may not provide the desired heading command. A large enough L ensures that the VTP always exists but may result in a longer convergence time.

Loiter

The procedure for a loiter pattern is the same as that for a straight line, except that the circle of radius L intersects with the loiter circle centered at O. The algorithm for loiter is similar to Algorithm 4 where (x_t, y_t) are determined using two-circle intersection geometry.

Even for a loiter maneuver, the performance of the guidance law depends on the selection of L. Figure 5(d) shows the performance of NLGL for loiter maneuvers with different L. Similar to the straight-line case, with an increase in L, the performance of the algorithm improves. However, when $L > 100$, the VTP is too far ahead, and the vehicle tries to move inside the loiter circle to meet the VTP, resulting in higher cross-track error. Note that a smaller L leads to lower cross-track error while larger L leads to increased radius of convergence.

PURE PURSUIT AND LOS-BASED PATH FOLLOWING (PLOS)

PLOS path following is a combination of the pure pursuit and LOS-based guidance laws. The pure pursuit guidance law drives the UAV to the waypoint W_{i+1}, while the LOS guidance law steers the vehicle toward the LOS. The combination of these guidance laws with different gains will make the vehicle follow the path. The geometry of the vehicle and the path are shown in Figure 6.

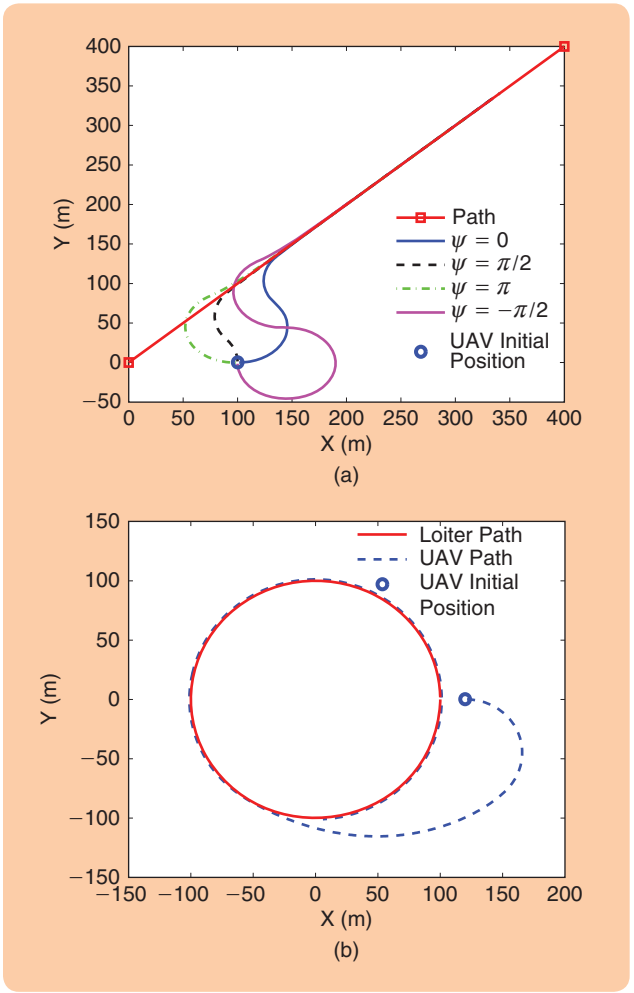


FIGURE 7 The trajectory of the unmanned aerial vehicle using pure pursuit with line-of-sight guidance for (a) straight-line following with different initial heading angles and (b) loiter following with initial heading angle $\psi = 0$.

The guidance law for pure pursuit is

$$\psi_p = k_1(\theta_d - \psi), \tag{3}$$

where k_1 is the gain and θ_d is the desired angle. The LOS guidance law ensures that the angle between the UAV and W_i is the same as that of the LOS angle. The LOS guidance law is given by

$$\psi_l = k_2 d, \tag{4}$$

where d is the cross-track error. The combined guidance law is

$$\psi_d = k_1(\theta_d - \psi) + k_2 d, \tag{5}$$

where θ is the LOS angle and θ_u is the angle between the UAV and W_i.

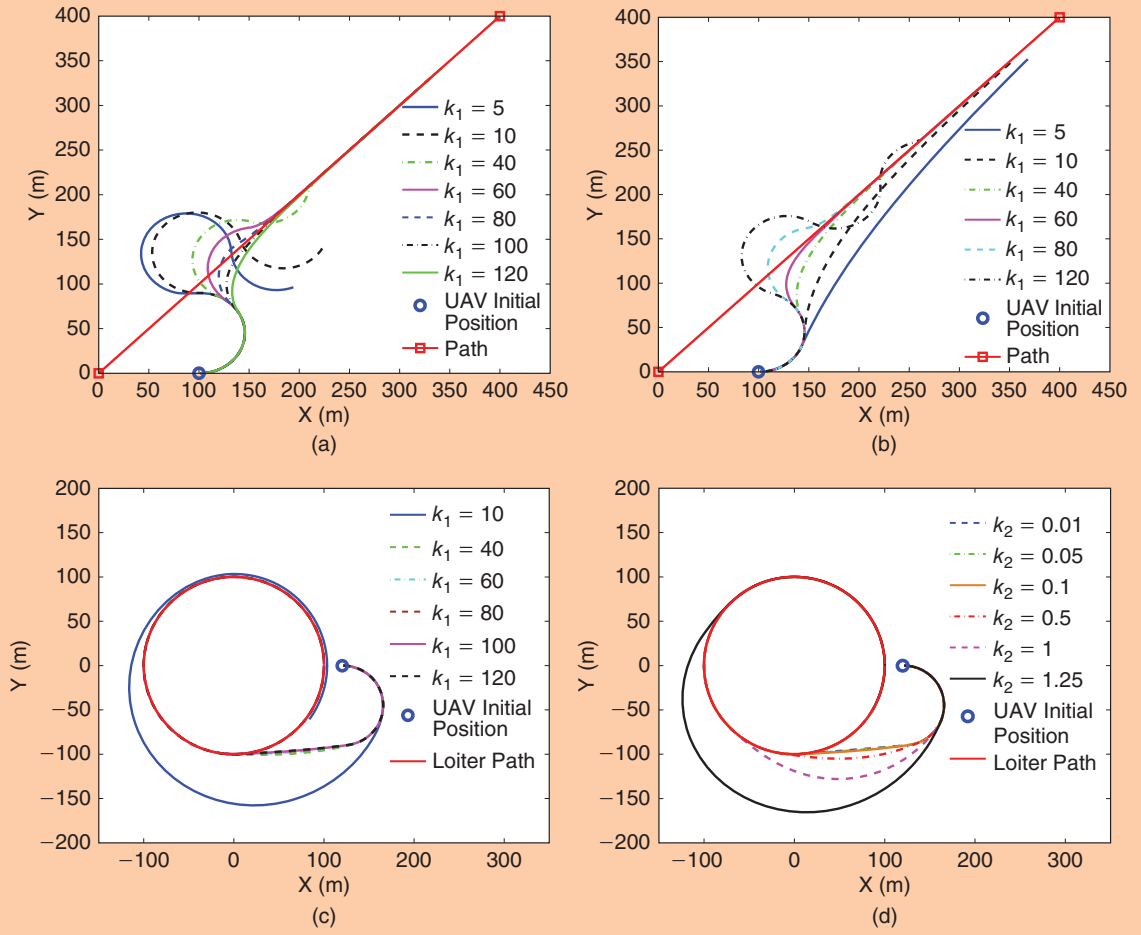


FIGURE 8 Unmanned aerial vehicle (UAV) paths using the PLOS algorithm for straight-line following [(a) and (b)] and loiter following [(c) and (d)]. (a) The effect on the UAV path when gain k_1 is varied and $k_2 = 3$. (b) The effect on the UAV path when gain k_2 is varied and $k_1 = 100$. (c) The effect on the UAV path for a loiter when gain k_1 is varied and $k_2 = 3$. (d) The effect of the UAV path for a loiter when gain k_2 is varied and $k_1 = 100$.

Straight-Line Following

The PLOS guidance law for straight-line path following is outlined in Algorithm 5.

Figure 7(a) shows that the vehicle can track the straight-line path for different initial heading angles. For $k_1 > 0$ and $k_2 > 0$, the PLOS guidance law is stable [45]. For a constant k_2, when k_1 is small, the LOS guidance dominates over the pursuit guidance, and hence the vehicle path oscillates along the desired straight-line path, as shown in Figure 8(a) for $k_1 = 5$. With an increase in k_1, the oscillation of the vehicle path decreases as the pursuit-guidance weight increases and draws the vehicle toward W_{i+1}. This effect can be seen in Figure 8(a) for $k_1 = 10$ and $k_1 = 40$. With further increase in k_1, the vehicle settles onto the straight-line path.

On the other hand, for a fixed k_1, with very low $k_2 = 0.5$, the pursuit guidance is dominant, and the vehicle is directed toward the waypoint W_{i+1}. With $k_2 = 1$, the vehicle tries to move toward the desired path; however, it takes a longer time to converge onto the path. The vehicle tracks the path quickly for gains $k_2 = 2$ and $k_2 = 3$. However, when $k_2 > 3$, the LOS guidance dominates, resulting in oscillation of the path as shown in Figure 8(b). Hence, the gains must be selected appropriately through a trial-and-error process for the best path performance.

ALGORITHM 6 Algorithm for loiter using PLOS.

1: Initialize: $O = (x_l, y_l), p, r, k_1, k_2, v_a, \kappa$

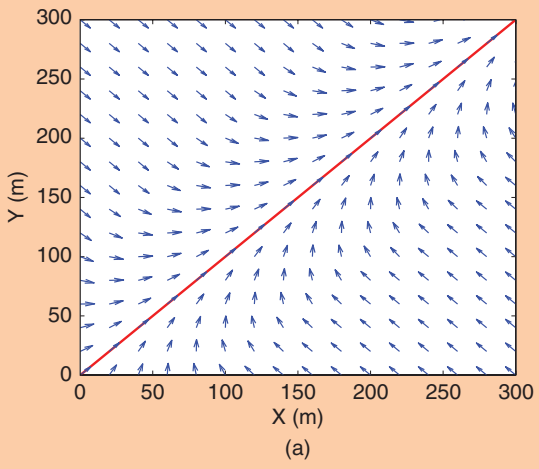
2: $d = \|p - O\| - r$

3: $\theta = \text{atan2}(y - y_l, x - x_l)$

4: $\theta_p = \theta \pm \pi/2$

5: $\psi_d = k_1(\theta_p - \psi) + k_2 d$

6: $u = \text{max\_limit}(\kappa(\psi_d - \psi)v_a)$



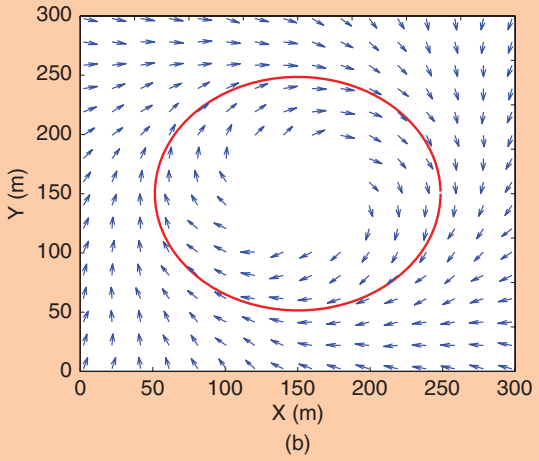


FIGURE 9 The vector fields for (a) the straight-line path and (b) the loiter path.

Loiter

Determining cross-track error d for the loiter is similar to doing so for the carrot-chasing algorithm. However, the reference heading angle θ_p is generated as shown in Figure 1(b). The ± sign in line 4 of Algorithm 6 represents the direction of the loiter.

Figure 7(b) shows that the vehicle is capable of tracking a circle with an initial heading angle $\psi = 0$. To evaluate the sensitivity of the algorithm to gains k_1 and k_2, experiments keeping one gain constant while varying the other gain were performed. Figure 8(c) shows the performance of the PLOS algorithm with constant $k_2 = 0.1$ and varying k_1. With low k_1, the vehicle takes a longer time to converge onto the loiter path. However, with $k_1 > 10$, the vehicle tracks the circle with a smaller error.

Figure 8(d) shows the results for constant $k_1 = 100$ and varying k_2. The vehicle tracks the path for low values of $0 \le k_2 < 1$, while the performance degrades with an increase in k_2. This shows that for loiters the LOS guidance is sensitive and the gain k_2 must be very low.

ALGORITHM 7 Algorithm for straight-line following using vector field [16].

1: Initialize: $W_i = (x_i, y_i), W_{i+1} = (x_{i+1}, y_{i+1}), p, \psi, \chi^e, \tau, v_a, k, \alpha, \kappa$

2: $\theta \leftarrow \text{atan2}(y_{i+1} - y_i, x_{i+1} - x_i)$

3: $s^* \leftarrow \dfrac{(p - W_i)^\top (W_{i+1} - W_i)}{\| W_{i+1} - W_i \|^2}$

4: $\epsilon \leftarrow \| p - (s^*(W_{i+1} - W_i) + W_i) \|$

5: $\rho \leftarrow \text{sign}[(W_{i+1} - W_i) \times (p - W_i)]$

6: $\epsilon \leftarrow \rho\epsilon$

7: **if** $|\epsilon| > \tau$ **then**

8: $\quad \psi_d \leftarrow \theta - \rho\chi^e$

9: $\quad \psi_c \leftarrow \psi_d$

10: **else**

11: $\quad \psi_d \leftarrow \theta - (\chi^e)\left(\dfrac{\epsilon}{\tau}\right)^k$

12: $\quad \psi_c \leftarrow \psi_d - \left(\dfrac{k\chi^e v_a}{\alpha\tau^k}\right)\epsilon^{k-1}\sin\psi$

13: **end if**

14: $u = \text{max\_limit}(\kappa(\psi_c - \psi)v_a)$

VECTOR-FIELD-BASED PATH FOLLOWING

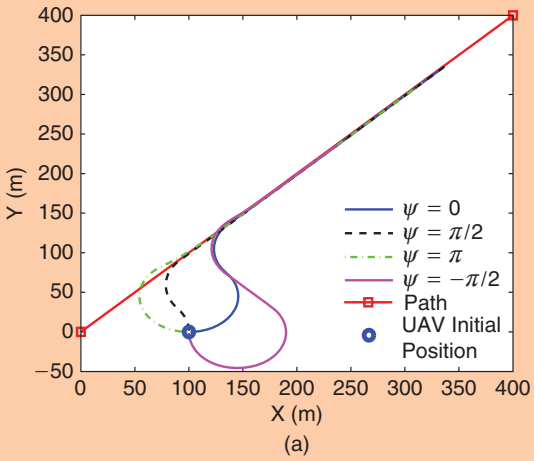
The main idea of this approach is that VFs indicate the direction of flow. If the vehicle follows the VF direction, then it will follow the path. The desired VFs for straight-line and loiter paths are shown in Figure 9(a) and (b), respectively. The stability of the algorithm is shown by Lyapunov stability arguments in [16].

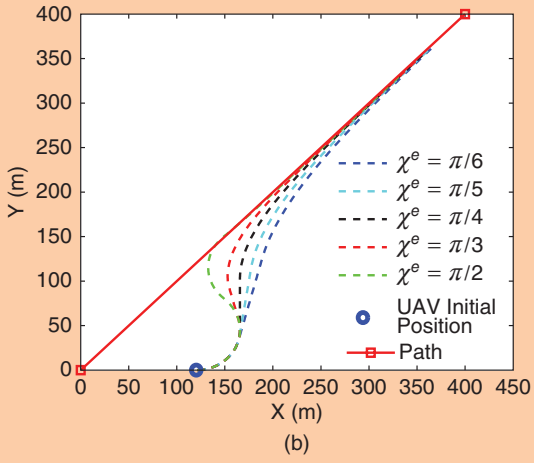
Straight-Line Following

VFs operate in two modes: 1) when the vehicle is away from the desired path and 2) when the vehicle is close to the path within a transition boundary τ. When the vehicle is far away, the natural strategy is to move toward the path, which is done by a parameter χ^e known as the *entry heading*. When the vehicle enters the transition region, χ^e is updated. The procedure to determine the desired heading angle ψ_d is given in Algorithm 7.

The performance of the VF varies with different values of the parameters χ^e, τ, k and initial heading angles. Figure 10(a) shows the ability of VFs to track the straight-line path with different initial heading angles. Further, to study the sensitivity of the model to χ^e and τ, two more experiments are performed. In the first experiment, the parameters $\tau = 45, k = 1, \psi = 0$ are held constant and while χ^e is varied between zero and $\pi/2$. The angle χ^e determines the angle of approach to the path. High χ^e will force the vehicle to travel toward LOS quickly, resulting in quick convergence. Lower χ^e values increase the convergence time and the cross-track error. The simulation results are shown in Figure 10(b).

In the second experiment, τ is varied while holding $\chi^e = \pi/2$, $k = 1$, and $\psi = 0$ constant. The parameter τ is the





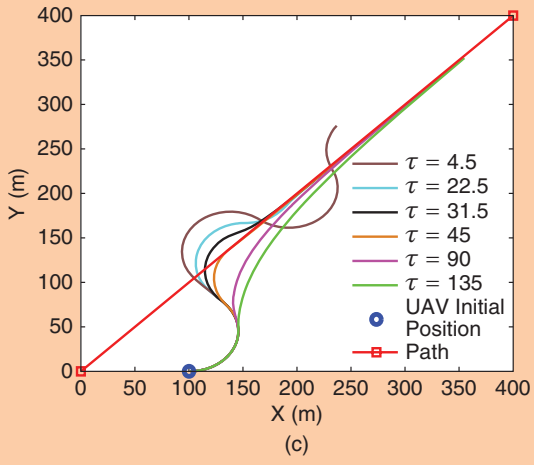


FIGURE 10 Unmanned aerial vehicle (UAV) trajectories for straight-line following using the VF algorithm. (a) UAV trajectories for different initial heading angles. (b) UAV trajectories for different χ^e, with $k = 1$ and $\tau = 45$. (c) UAV trajectories for different τ, with $k = 1$ and $\chi^e = \pi/2$.

width of the transition region near the path, as shown in Figure 9(a). When τ is small, the vehicle comes close to the path and then starts tracking the path. The time taken to converge onto the path is high. However, with an increase in τ the vehicle converges quickly onto the path. With further increase in τ, the time taken to converge onto the path increases. This effect is similar to the VTP where having a VTP too far ahead results in lower performance. The results of this analysis can be confirmed through simulations, as shown in Figure 10(c).

ALGORITHM 8 Algorithm for loiter following using the vector field algorithm [16].

1: Initialize: $O = (x_l, y_l), p, r, v_a, \alpha, k, \kappa$

2: $d = \| p - O \|$

3: $\theta = \text{atan2}(y - y_l, x - x_l)$

4: **if** $d > 2r$ **then**

5: $\quad \psi_d \leftarrow \theta - \pi + \sin^{-1}\left(\frac{r}{d}\right)$

6: $\quad \psi_c \leftarrow \psi_d + \frac{v_a}{\alpha d}\sin(\psi - \theta)$

7: **else**

8: $\quad \psi_d \leftarrow \theta - \frac{\pi}{2} - \frac{\pi}{3}\left(\frac{d - r}{r}\right)^k$

9: $\quad \psi_c \leftarrow \psi_d - \frac{v_a}{\alpha d}\sin(\psi - \theta) - \frac{k v_a \pi}{3 r^k \alpha}\tilde{d}^{k-1}\cos(\psi - \theta)$

10: **end if**

11: $u = \text{max\_limit}(\kappa(\psi_c - \psi) v_a)$

Loiter

For loitering, the VF is created such that it converges on the loiter circle circumference, as shown in Figure 9(b). When the vehicle is outside or inside the loiter circle, the VF directs onto the circle circumference. The desired heading angle for the vehicle is determined as shown in Algorithm 8.

Figure 11(a) shows the vehicle tracking the loiter circle for $\psi = 0$ with parameters $k = 1$ and $\alpha = 10$. Unlike the straight-line following case, loiter following has only two parameters, k and α. The sensitivity with respect to k is significant, hence $k = 1$ is kept constant. Figure 11(b) shows that the convergence time to the loiter circle decreases with increasing α. From the simulations, increasing α beyond a certain value does not decrease the convergence time noticeably. However, increasing α significantly increases the control effort exerted by the guidance law, which will affect the performance metrics.

LQR PATH-FOLLOWING ALGORITHM

Most of the above guidance laws use geometric methods to compute the commanded heading angle. In the LQR-based path-following algorithm, the control effort is computed using optimal control theory. This framework allows the vehicle to determine the minimal control effort required to minimize the cross-track error and cross-track error rate. Details of the LQR formulation for the path-following problem are given in [20]. The LQR control action is

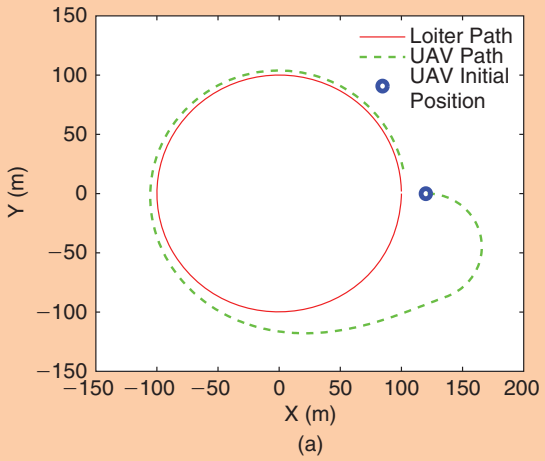


FIGURE 11 Unmanned aerial vehicle (UAV) trajectories for loiter following using the VF algorithm. (a) The UAV trajectory with initial heading $\psi = 0$. (b) The UAV trajectory for different α, constant $k = 1$, and initial heading angle $\psi = 0$.

$$u = -\left[d\sqrt{\left|\frac{\tau}{\tau - d}\right|} + v_d\sqrt{2\sqrt{\left|\frac{\tau}{\tau - d}\right|} + q_{22}}\right], \tag{6}$$

where τ is the bounded region along the path (similar to τ in VF), q_{22} is the gain parameter of the $\boldsymbol{Q}$ matrix, and $v_d = v_a \sin(\psi - \theta)$ is the cross-track error velocity. The $\boldsymbol{Q}$ matrix is

$$\boldsymbol{Q} = \begin{bmatrix} q_{11} & 0 \\ 0 & q_{22} \end{bmatrix},$$

where $q_{11} = |\tau/\tau - d|$. The LQR path-following algorithm has two tuning parameters, τ and q_{22}.

The LQR path-following algorithm computes the optimal control effort to minimize the cross-track error and also ensures that the vehicle is close to the path through the use of a boundary defined by τ. Three parameters control the trajectory of the vehicle: $\boldsymbol{R}$, q_{11} and q_{22}. For a stable LQR controller, $\boldsymbol{R}$ and $\boldsymbol{Q}$ must be positive definite; typically, $\boldsymbol{R}$ is the identity, $q_{22} > 0$ can be selected, and q_{11} is always positive as given above.

ALGORITHM 9 Algorithm for straight-line following using the LQR.

1: Initialize: $W_i = (x_i, y_i), W_{i+1} = (x_{i+1}, y_{i+1}), \tau, q_{22}, v_a, \psi$

2: $\theta_u = \text{atan2}(x - x_i, y - y_i)$

3: $\theta = \text{atan2}(x_{i+1} - x_i, y_{i+1} - y_i)$

4: $R_u = \| p - W_i \|$

5: $d = R\sin(\theta - \theta_u)$

6: $v_d = v_a \sin(\psi - \theta)$

7: Determine u as defined in (6) with maximum u as given in Algorithm 1

Straight-Line Following

Consider a straight-line segment defined by two waypoints, as shown in Figure 2(a). Let R_u be the displacement of the UAV with respect to point W_1 making an angle θ_u with respect to the coordinate system. The UAV position and velocity errors are

$$d = R_u \sin(\theta - \theta_v) \tag{7}$$

and

$$v_d = \dot{d} = v_a \sin(\psi - \theta), \tag{8}$$

respectively. By selecting τ and q_{22}, the required control effort u can be computed from Algorithm 9.

Figure 12(a) shows that the vehicle can track a straight-line path for different initial heading angles with $q_{22} = 1$. For a given heading angle of $\psi = 0$, the trajectories of the vehicle for different q_{22} are shown in Figure 12(b). With very low values of q_{22}, the vehicle tries to track the path very quickly, as the penalty on the state is very low. The vehicle applies maximum acceleration and hence the control effort is high. On the other hand, increasing q_{22}, increases the time to converge to the path, as shown in Figure 12(b).

Loiter

Consider a circular path of radius r as shown in Figure 1(b). Let $R = d + r$ be the displacement of the UAV with respect to the center of the circular path. The UAV position and velocity errors are

$$d = R - r \tag{9}$$

and

$$v_d = v_a \sin(\psi - \theta), \tag{10}$$

respectively. The values of d and v_d in (10) are substituted in (6) for a given q_{22} and τ. The procedure for determining u for loiter is described in Algorithm 10.

The loiter trajectory with $r = 100$ and $q_{22} = 0.1$ is shown in Figure 12(c). Trajectories for different values of q_{22} are shown in Figure 12(d). UAV performance is similar to that

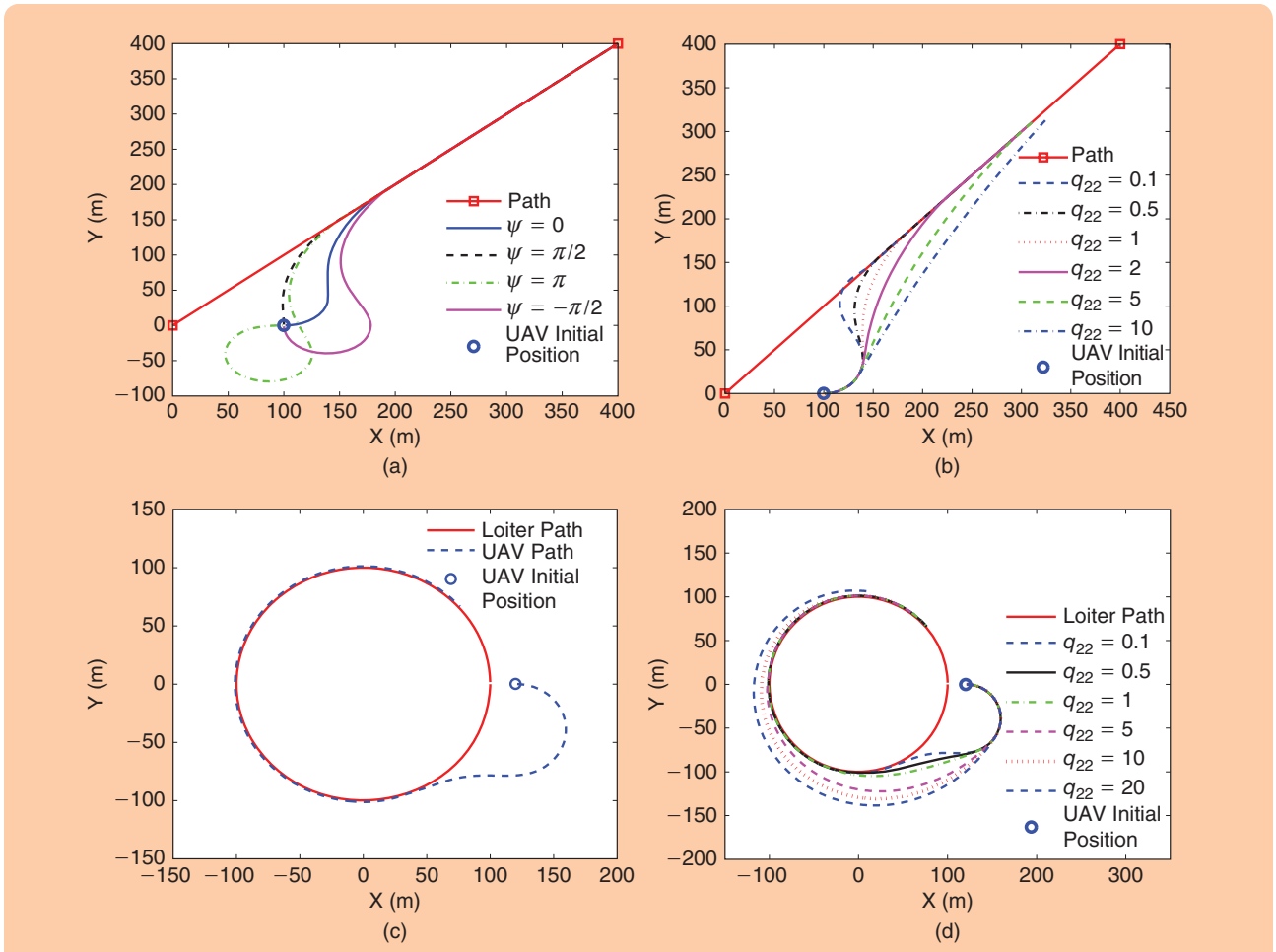


FIGURE 12 The performance of the adaptive linear quadratic regulator (LQR) path-following algorithm for the straight-line path: (a) with different initial heading angles and (b) with different q_{22} values for an initial heading angle of 0 rad. The performance of the LQR path-following algorithm for loiter: (c) with an initial heading angle of 0 rad and (d) with different q_{22} values for an initial heading angle of 0 rad.

for the straight-line case, where the path converges onto the loiter quickly for low values of q_{22}, while the convergence time increases for high values of q_{22}. Selecting q_{22} around one to two results in a performance where the control effort is not high and the convergence to the path is quick.

ALGORITHM 10 Algorithm for loiter following using the LQR.

1: Initialize: $O = (x_l, y_l), p, r, v_a, q_{22}$

2: $\theta = \text{atan2}\,(x - x_l, y - y_l)$

3: $d = \|p - O\| - r$

4: $v_d = v_a \sin(\psi - \theta)$

5: Determine u as defined in (6) with maximum u as given in Algorithm 1

COMPARISON OF ALGORITHMS

The different path-following algorithms are compared using a 6-DOF aircraft simulation model without system delays [46]. The 6-DOF simulation model is based on a delta-wing shaped Zagi aircraft [46]. For comparison purposes, two metrics are defined—total control effort (U) and total cross-track error (D). The total control effort quantifies the control demands of the algorithm. If the solution takes too many turns, then the control effort will be high, which is not desirable. The total cross-track error is the offset of the vehicle from the desired path. For performance comparison, several types of norms can be considered, such as the one-norm ($|\cdot|$), two-norm ($\|\cdot\|$), and infinity norm. The infinity norm does not provide sufficient information as all the guidance laws apply maximum control effort during loiter and the maximum cross-track error for all the guidance laws is similar. The two-norm provides better comparison information than the one-norm. As an example, see the Pareto front of VF path-following in Figure 13(a) and (b). Figure 13(b) shows the (1, 3, 3) gain parameter value to be the best of several potential gain parameters for the two-norm case. For a given guidance law, let $u(t)$ and $d(t)$ be the control effort and

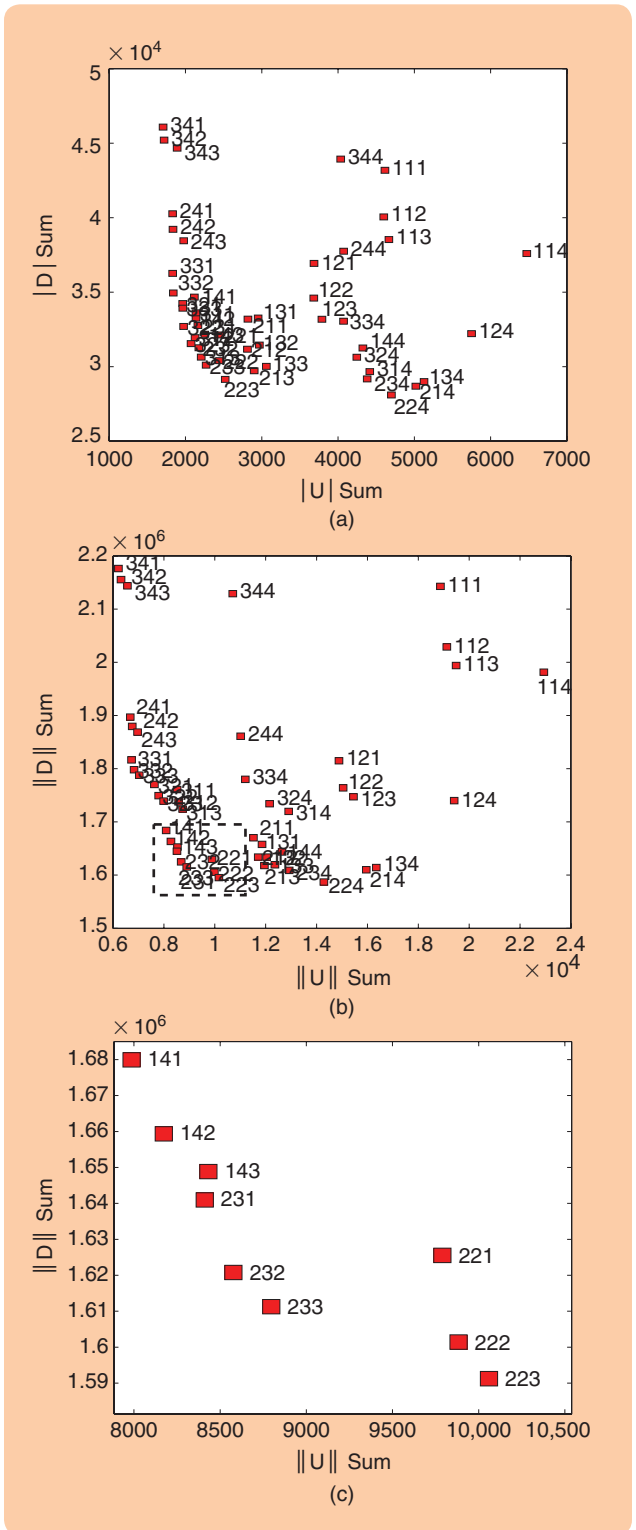


FIGURE 13 The Pareto front of the VF algorithm for different χ^e, τ, and α gain parameters. The gain parameters χ^e, τ, α are selected for the sets $\chi^e = [\pi/2, \pi/3, \pi/4]$, $\tau = [22.5, 31.5, 45, 90]$, and $\alpha = [2, 3, 5, 10]$. The text label for each point refers to the index of different gain parameters χ^e, τ, α. For example, 133 refers to $\chi^e(1) = \pi/2$, $\tau(3) = 45$, $\alpha(3) = 5$. (a) Performance with one-norm and (b) performance with two-norm. (c) Zoomed rectangle area of the two-norm figure. The optimal gain parameter is [233], representing $\chi^e = \pi/2$, $\tau = 45$, $\alpha = 5$.

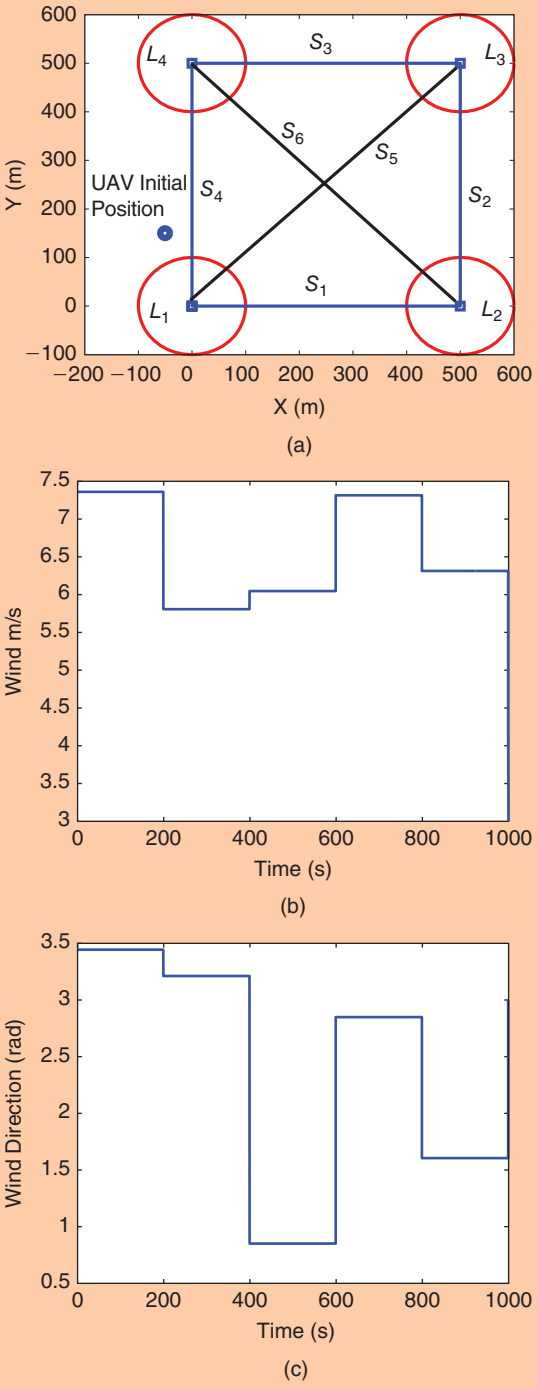


FIGURE 14 (a) The mission for the unmanned aerial vehicle is to follow straight-line paths and loiters in a sequence. The visiting sequence is $S_4, L_4, S_3, L_3, S_2, L_2, S_1, L_1, S_5, L_3, S_3, L_4, S_6$. (b) Change in wind magnitude with respect to time. (c) Change in wind direction with respect to time.

cross-track error at time t respectively, then the total control effort using the two-norm is defined as

$$U = \sum_{t=0}^{t=T} u(t)^2, \tag{11}$$

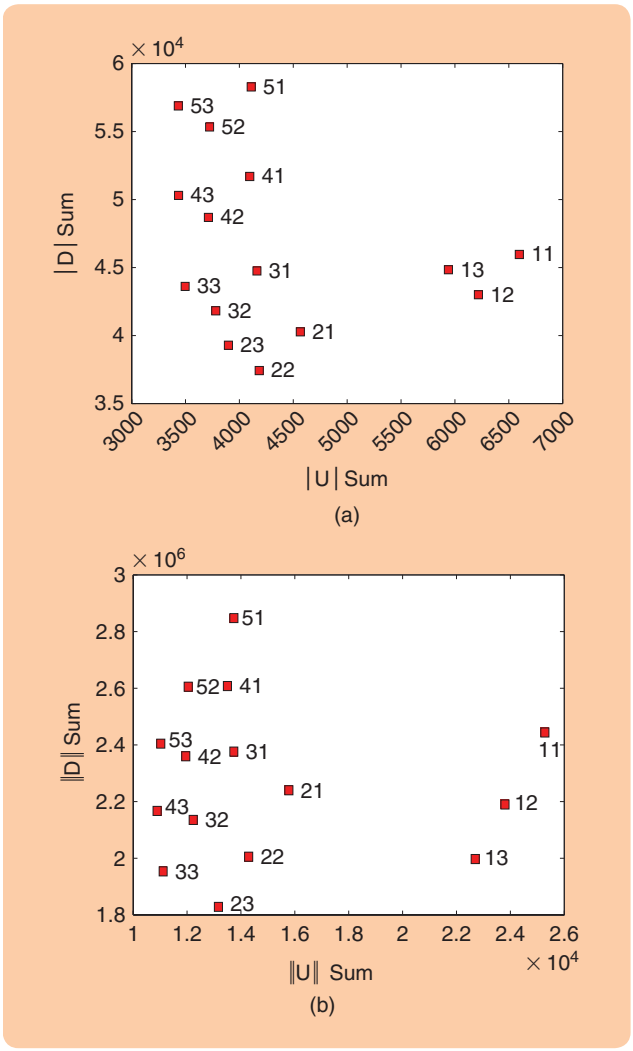


FIGURE 15 The Pareto front of the carrot-chasing path-following algorithm for different gain parameters. The gain parameters δ and λ are selected from the sets $\delta \in \{10, 30, 50, 75, 100\}$, $\lambda \in \{0.1, 0.2, 0.4\}$. The text label for each point refers to the index of the gain parameters $\delta\lambda$. For example, 43 refers to $\delta(4) = 75$, $\lambda(3) = 0.4$. (a) Performance with one-norm and (b) performance with two-norm. The selected gain parameter is [2, 3], representing $\delta = 30$, $\lambda = 0.4$.

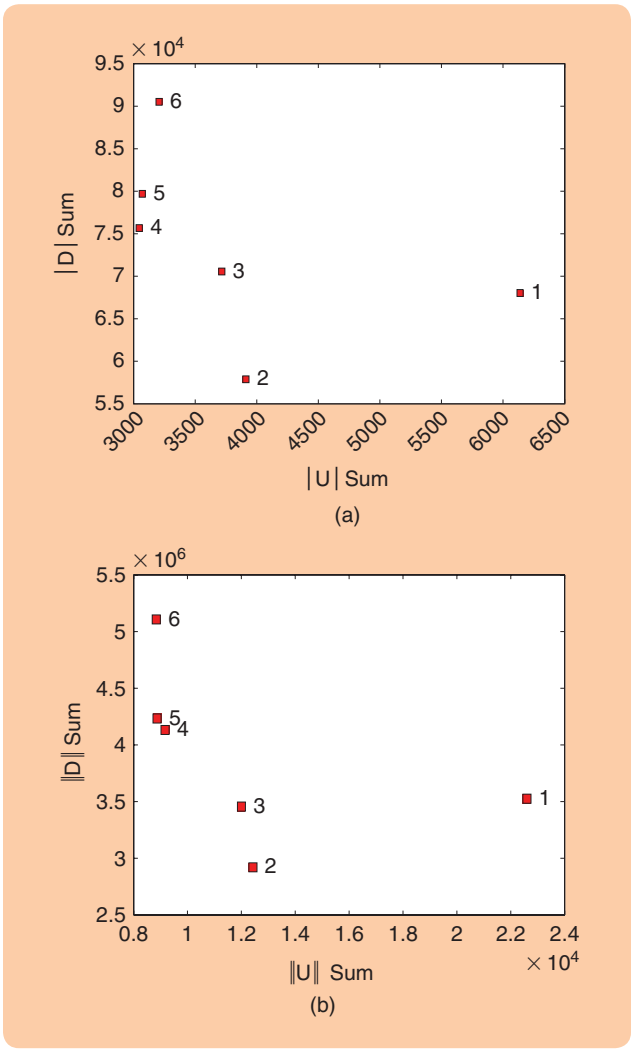


FIGURE 16 The Pareto front of the NLGL algorithm for different L gain parameters. The gain parameter L is selected from the set $L \in \{25, 50, 75, 100, 125, 150\}$. The text label for each point refers to the index of the gain parameter L. For example, "4" refers to $L(4) = 100$. (a) Performance with one-norm and (b) performance with two-norm. The selected gain parameter is [2], representing $L = 50$.

while the total cross track error is given as

$$D = \sum_{t=0}^{t=T} d(t)^2. \tag{12}$$

In the real world, UAVs have to face wind disturbances, so path-following algorithms must be robust to wind. To analyze the performance of the algorithms in the presence of varying wind conditions, a mission is considered where the UAV has to alternately follow straight-line paths and perform loiters. Figure 14(a) shows the sample mission plan and the sequence of waypoints for the UAV to navigate, where S_i denotes the ith straight-line path and L_j denotes the jth loiter. The vehicle follows a straight line, and when it reaches the loiter radius of the ending waypoint, it switches to loiter. When the UAV is close to the next straight-line path, it switches to a straight-line path. Each simulation used a constant northeast wind of 3 m/s with up to 5 m/s wind gusts in a random direction for 20 s, as shown in Figure 14(b) and (c), respectively. The time taken by the vehicle to perform one loiter (about 42 s) is almost twice the time taken for a straight-line path (about 20 s).

Monte Carlo simulations are performed and a comprehensive comparison made of performance metrics for different path-following algorithms, as shown in Figures 15(a)–19. There were $N = 1000$ simulations with the same initial position and heading angle. For each simulation, the constant wind and gust profile are different. The gust direction changes randomly using a uniform normal distribution

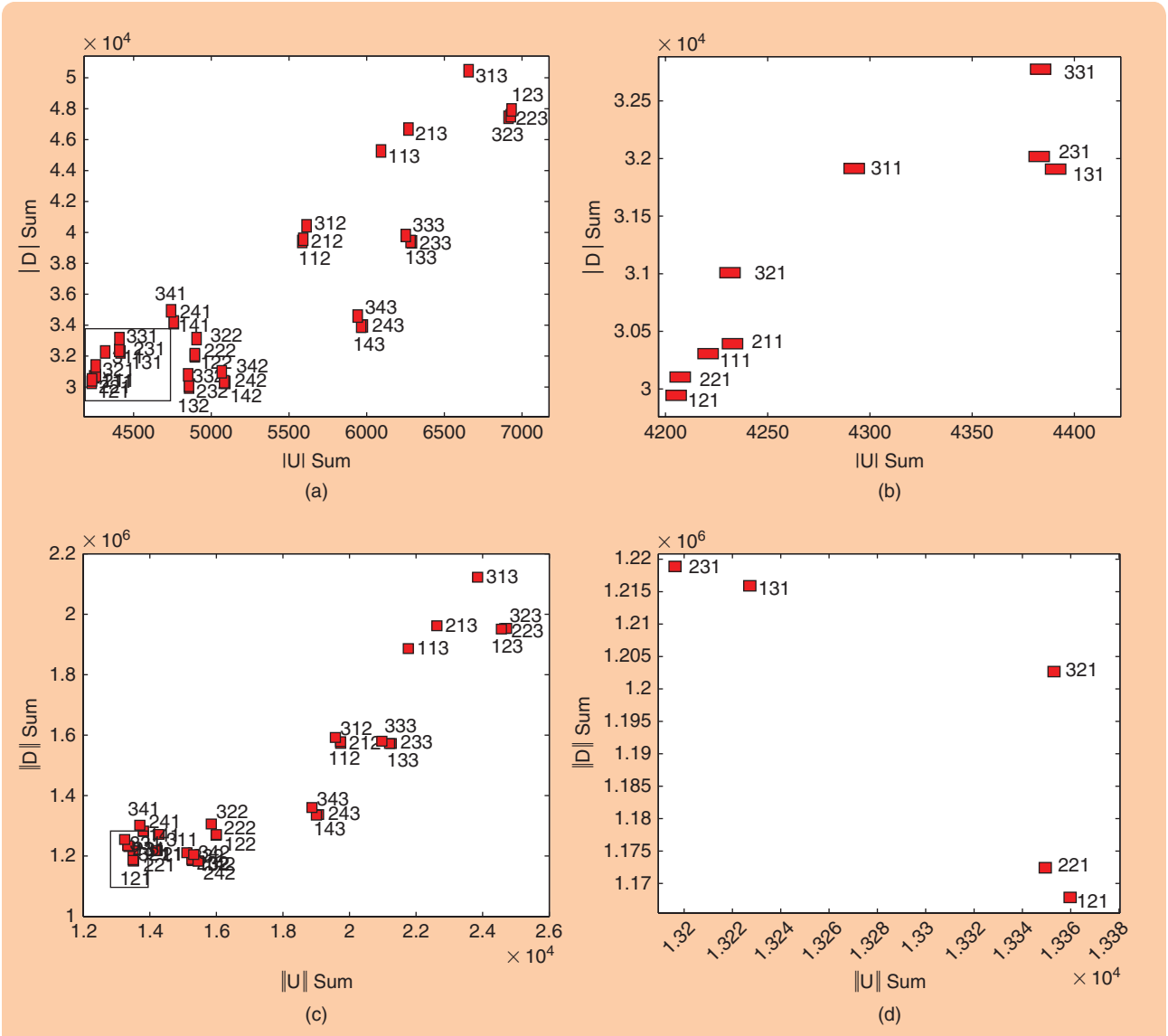


FIGURE 17 The Pareto front of the PLOS algorithm for different k_1, k_2^s, k_2^c gain parameters, where k_2^s refers to k_2 of a straight line while k_2^c refers to k_2 of a loiter path. The gain parameters are $k_1 \in \{60, 80, 100, 120\}$, $k_2^s \in \{2, 3, 5\}$, and $k_2^c \in \{0.05, 0.1, 0.5\}$. The text label for each point refers to the index of different gain parameters as $k_1 k_2^s k_2^c$. For example, 133 refers to $k_1(1) = 60$, $k_2^s(3) = 5$, $k_2^c(3) = 0.5$. (a) Performance with one-norm. (b) Zoomed performance of one-norm. (c) Performance with two-norm. (d) Zoomed performance of two-norm. The selected gain parameter is [121], representing $k_1 = 60$, $k_2^s = 3$, $k_2^c = 0.05$.

$\mathcal{N}(0,1)$ every 20 s. The wind and gust parameters of the 1000 simulations remain the same while testing the different guidance laws. The performance of guidance laws with respect to both cross-track error and control effort are compared in terms of the average percentage weight ζ metric,

$$\zeta = \Gamma \bar{U} + (1 - \Gamma)\bar{D}, \tag{13}$$

where the tradeoff weight Γ varied from zero to one, $\bar{U} = (1/N)\sum_{i=1}^{N} U_i$ is the mean control effort, and $\bar{D} = (1/N)\sum_{i=1}^{N} D_i$ is the mean cross-track error.

The performance of the algorithms depends on the selected gain parameters. To ensure that the parameters are the best for each algorithm, a set of 30 simulations are performed for different parameter settings of an algorithm. The same set of 30 simulations are used for evaluating the performance all the algorithms. The mean Pareto fronts are plotted for each algorithm, as shown in Figures 15(a)–18(b). Each point in the plot shows the performance for a given set of gain parameters. Using these plots as reference, the gain parameter that has the lowest control effort and cross-track error was selected. The set parameters that performed best for each path-following algorithm is given in Table 1.

Figure 19 shows the performance of different path-following algorithms with change in tradeoff weight Γ. The VF path-following algorithm performs the best for any given

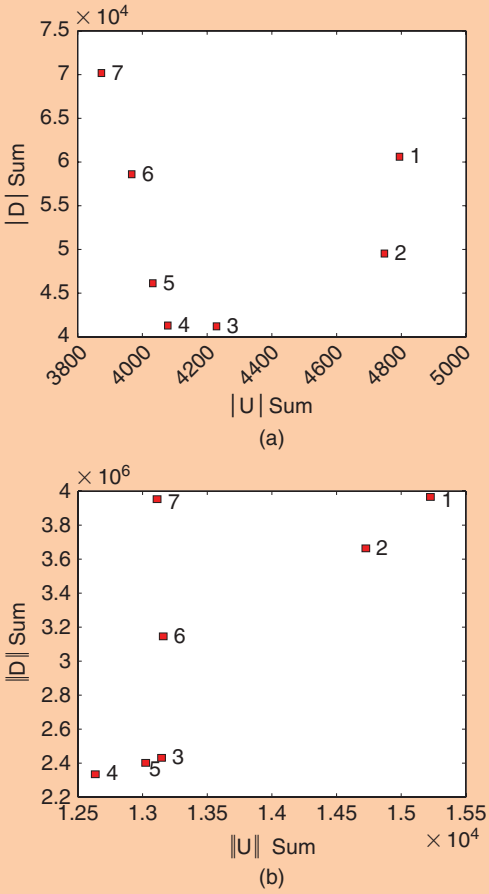


FIGURE 18 The Pareto front of the LQR algorithm for different q_{22} gain parameters. The gain parameters are $q_{22} \in \{0.25, 0.5, 0.75, 1, 2, 5, 10\}$. The text label for each point refers to the index of the q_{22} gain parameter. For example, "4" refers to $q_{22}(4) = 1$. (a) Performance for one-norm and (b) two-norm. The selected gain parameter is $q_{22} = 1$.

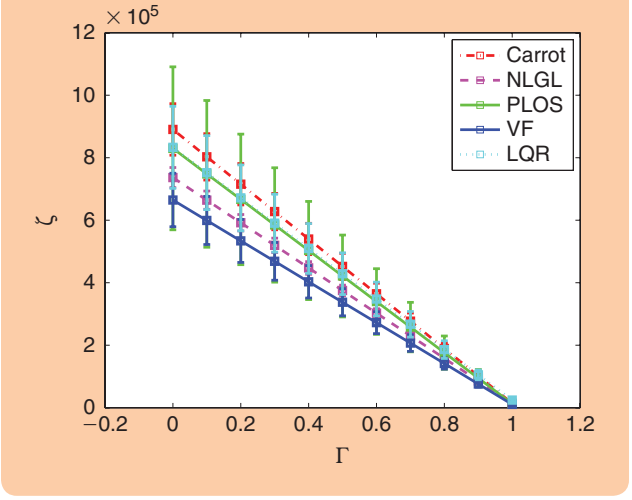


FIGURE 19 The average performance of the mission with different guidance laws for varying weight on cross-track error and control effort.

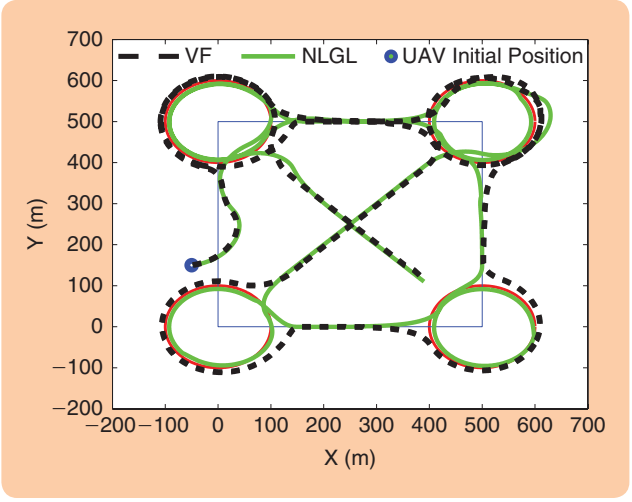


FIGURE 20 The vehicle route using nonlinear (NLGL) and vector field (VF) guidance laws. The dashed black path is the unmanned aerial vehicle path using the VF algorithm and the solid green path is the vehicle path with NLGL algorithm.

weight on the cross-track error and control effort. The NLGL closely follows VF. The carrot-chasing algorithm has the worst performance, with high variance and high cross-track error, while requiring a control effort comparable to that of the other algorithms. The LQR and PLOS algorithms exhibit similar performance but not as good as the VF and NLGL algorithms. The paths taken by the NLGL and VF algorithms are shown in Figure 20. The route of the vehicle using the VF algorithm has a lower error than the NLGL algorithm.

CONCLUSIONS

A detailed analysis of five path-following algorithms was performed with different parameter settings and wind disturbances. All the algorithms control a vehicle to follow a path under different wind conditions. Monte Carlo simulations show that the VF path-following technique more accurately follows the path than the other techniques and also requires the least control effort. The NLGL algorithm follows closely the VF algorithm. The carrot-chasing algorithm performed the worst. The advantages and limitations of the algorithms are summarized in Table 1.

Monte Carlo simulations also showed the robustness of the algorithms to track paths under different wind conditions—both in direction and magnitude. Although the wind estimation is assumed to be accurate, most low-cost autopilots do not have accurate wind estimation because of inferior sensors and insufficient computational power to use advanced estimation algorithms. Having accurate wind estimators would improve the performance of path-following algorithms in the field.

Commercial autopilots are installed on airplanes whose aerodynamic coefficients are often unknown. While the coefficients can be estimated using system identification techniques [47], the procedure requires a wind tunnel and extensive flight tests. These are time consuming and expensive.

TABLE 1 The parameters used for different guidance laws: their advantages and limitations.

Algorithm	Parameters for the Algorithm	Advantages	Limitations
Carrot	Straight line: $\delta = 30$ m Loiter: $\lambda = 0.4$ rad	Simple implementation	Not robust to large wind disturbances
NLGL	Straight line: $L = 50$ Loiter: $L = 50$	Same algorithm can be used for any type of path	Cross-track error is more than VF
PLOS	Straight line: $k_1 = 60, k_2 = 3$ Loiter: $k_1 = 60, k_2 = 0.05$	Simple implementation; intuitive	Sensitive to gains; cannot track the path when $v_w > 0.5\, v_a$
LQR	Straight line: $q_{22} = 1$ Loiter: $q_{22} = 1$	Optimizes control effort	Has high cross-track error; cannot track path when $v_w > 0.5\, v_a$
VF	Straight line: $\tau = 45,\ \chi^e = \pi/3,\ \alpha = 5$ Loiter: $\alpha = 5$	Low cross-track error	Three parameters to tune; chattering effect

Furthermore, additional sensors are needed for estimating these coefficients. Nonlinear control techniques in the literature can be useful under these situations. However, some of these techniques are computationally intensive [48], [49]. Designing path-following algorithms that require low computational cost is an area for further exploration.

For UAVs to be accepted in the national aerospace, 3D path planning is essential. While there has been some research on 3D and four-dimensional planning, most of the research is theoretical and most commercial autopilots do not have 3D path-planning algorithms.

The carrot-following and PLOS algorithms can be easily extended to 3D paths. Computing u for LQR becomes computational expensive, as solving the algebraic Ricatti equation for LQR control in higher dimensions is computationally expensive. Designing VFs in 3D is not simple and requires significant work. Finding the VTP for the NLGL is simple but requires additional inputs to compute u, taking flight path angle into account. Therefore, LQR, VF, and NLGL require additional development for 3D path following. Although there are several articles on 3D path following, the path-following problem is still challenging in 3D environments.

ACKNOWLEDGMENTS

P.B. Sujit would like to thank Rajnikanth Sharma at Utah State University, Mangal Kothari at Northumbria University, and Ashwini Ratnoo at Indian Institute of Science, Bangalore, India, for their insightful comments. This work is partly supported by the grant C2007-ISRP-2007 from ISR-Porto and FCT-Portugal, and FCT project PERSIST-PTDC/EEA-CRO/104901/2008.

AUTHOR INFORMATION

P.B. Sujit (sujit@iiitd.ac.in) is an assistant professor at the Indraprastha Institute of Information Technology, New Delhi, India. He received the bachelor's degree from Bangalore University, India, in 1998, the master's degree from Visveswaraya Technological University, India, in 2002, and the Ph.D. from the Indian Institute of Science, Bangalore, India, in 2006. Previously, he was a research scientist at the Universidade do Porto, Portugal, and a post-doctoral fellow at Brigham Young University, Provo, Utah. His research interests include unmanned aerial and underwater vehicles, multirobot systems, and human-robot interaction. He can be contacted at Indraprastha Institute of Information Technology-Delhi, Okhla Phase 3, New Delhi 110020, India.

Srikanth Saripalli is an assistant professor in the School of Earth and Space Exploration at Arizona State University. He received the Ph.D. and M.S. degrees from the University of Southern California in 2007 and 2002, respectively, and the B.E. (Hons) degree from Birla Institute of Technology and Sciences, Pilani, India, in 1999. His research focuses on robotic exploration, particularly in air and space, and the necessary foundations in perception, planning, and control for this domain. His work spans algorithmic design and implementation to field experimentation of aerial robots that explore difficult, dangerous, and usually inaccessible places on earth and other planetary bodies to further scientific knowledge.

João Borges Sousa is an assistant professor with the Electrical and Computer Engineering Department at Porto University. He is the head of the Underwater Systems and Technologies Laboratory. His research interests include unmanned vehicles, control architectures, control, and coordination. He has been leading the design, implementation, and deployment of advanced unmanned air and ocean vehicle systems in projects funded by the Portuguese Science Foundation, the Portuguese Ministry of Defense, the Portuguese Innovation Agency, the European Union, NATO, and, in the United States, the Office of Naval Research and DARPA. In 2006 the laboratory received the national BES Innovation National Award for the design of the Light Autonomous Underwater Vehicle. In 2007 he received an

outstanding teaching award from Porto University. He has authored approximately 220 scientific papers.

REFERENCES

[1] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotović, "Performance limitations in reference tracking and path following for nonlinear systems," *Automatica*, vol. 44, no. 3, pp. 598–610, 2008.

[2] V. Cichella, E. Xargay, V. Dobrokhodov, I. Kaminer, A. M. Pascoal, and N. Hovakimyan, "Geometric 3D path-following control for a fixed-wing UAV on SO(3)," in *Proc. AIAA Conf. Guidance, Navigation Control*, Aug. 8–11, 2011, pp. 3578–3592.

[3] C. Samson, "Time-varying feedback stabilization of car-like wheeled mobile robots," *Int. J. Robot. Res.*, vol. 12, no. 1, pp. 55–64, 1993.

[4] G. Conte, S. Duranti, and T. Merz, "Dynamic 3D path following for an autonomous helicopter," in *Proc. 5th IFAC Symp. Intelligent Autonomous Vehicles*, Oxford, U.K., 2004, pp. 473–478.

[5] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. de Lellis, and A. Pironti, "Path generation and tracking in 3-D for UAVs," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 4, pp. 980–988, 2009.

[6] J. Osborne and R. Rysdyk, "Waypoint guidance for small UAVs in wind," in *Proc. AIAA Infotech Aerospace*, Arlington, VA, 2005, vol. 193, nos. 1–4, pp. 1–12.

[7] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," in *Proc. 6th IFAC Conf. Manoeuvring Control Marine Craft*, Sept. 17–19, 2003, pp. 244–249.

[8] F. A. Papoulias, "Stability considerations of guidance and control laws for autonomous underwater vehicles in the horizontal plane," in *Proc. 7th Int. Symp. Unmanned Untethered Vehicle Technology*, 1991, pp. 140–158.

[9] R. Rysdyk, "UAV path following for constant line-of-sight," presented at the 2nd AIAA Unmanned Unlimited Systems, Technologies Operations Aerospace, Land Sea Conference, 2003, Paper AIAA-2004-4900.

[10] R. Rysdyk, "Unmanned aerial vehicle path following for target observation in wind," *J. Guidance, Control, Dyn.*, vol. 29, no. 5, pp. 1092–1100, 2006.

[11] S. Park, J. Deystt, and J. P. How, "Performance and Lyapunov stability of a nonlinear path-following guidance method," *J. Guidance, Control, Dyn.*, vol. 30, no. 6, pp. 1718–1728, 2007.

[12] M. Kothari, I. Postlethwaite, and D. W. Gu, "A suboptimal path planning algorithm using rapidly-exploring random trees," *Int. J. Aerosp. Innov.*, vol. 2, no. 1, pp. 93–104, 2010.

[13] L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, no. 3, pp. 497–516, 1957.

[14] S. Hota and D. Ghose, "A modified Dubins method for optimal path planning of a miniature air vehicle converging to a straight line path," in *Proc. American Control Conf.*, 2009, pp. 2397–2402.

[15] S. Hota and D. Ghose, "Optimal geometrical path in 3D with curvature constraint," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, Piscataway, NJ, 2010, pp. 113–118.

[16] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," *IEEE Trans. Robot.*, vol. 23, no. 3, pp. 519–529, 2007.

[17] M. Sun, R. Zhu, and X. Yang, "UAV path generation, path following and gimbal control," in *Proc. IEEE Int. Conf. Networking, Sensing Control*, 2008, pp. 870–873.

[18] I. Rhee, S. Park, and C. K. Ryoo, "A tight path following algorithm of an UAS based on PID control," in *Proc. SICE Annu. Conf.*, 2010, pp. 1270–1273.

[19] B. Wang, X. Dong, and B. Chen, "Cascaded control of 3D path following for an unmanned helicopter," in *Proc. IEEE Conf. Cybernetics Intelligent Systems*, 2010, pp. 70–75.

[20] A. Ratnoo, P. B. Sujit, and M. Kothari, "Optimal path following for high wind flights," in *Proc. IFAC World Congr.*, Milan, Italy, Aug. 28–Sept. 2, 2011, pp. 12,985–12,990.

[21] S. Lee, A. Cho, and C. Kee, "Integrated waypoint path generation and following of an unmanned aerial vehicle," *Aircr. Eng. Aerosp. Technol.*, vol. 82, no. 5, pp. 296–304, 2010.

[22] A. J. Healey and D. Lienard, "Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles," *IEEE J. Oceanic Eng.*, vol. 18, no. 3, pp. 327–339, July 1993.

[23] Z. Li, J. Sun, and S. Oh, "Handling roll constraints for path following of marine surface vessels using coordinated rudder and propulsion control," in *Proc. American Control Conf.*, 2010, pp. 6010–6015.

[24] S. Jackson, J. Tisdale, M. Kamgarpour, B. Basso, and J. K. Hedrick, "Tracking controllers for small UAVs with wind disturbances: Theory and flight results," in *Proc. 47th IEEE Conf. Decision Control*, 2008, pp. 564–569.

[25] P. Encarnacao and A. Pascoal, "Combined trajectory tracking and path following: An application to the coordinated control of autonomous marine craft," in *Proc. 40th IEEE Conf. Decision Control*, 2001, pp. 964–969.

[26] V. K. Chitrakaran, D. M. Dawson, H. Kannan, and M. Feemster, "Vision assisted autonomous path following for unmanned aerial vehicles," in *Proc. 45th IEEE Conf. Decision Control*, 2006, pp. 63–68.

[27] M. Ahmed and K. Subbarao, "Nonlinear 3-D trajectory guidance for unmanned aerial vehicles," in *Proc. 11th Int. Conf. Control Automation Robotics Vision*, 2010, pp. 1923–1927.

[28] Z. Li, J. Sun, and S. Oh, "Design, analysis and experimental validation of a robust nonlinear path following controller for marine surface vessels," *Automatica*, vol. 45, no. 7, pp. 1649–1658, 2009.

[29] R. Cunha, C. Silvestre, and A. Pascoal, "A path following controller for model-scale helicopters," in *Proc. European Control Conf.*, Cambridge, U.K., 2003.

[30] C. Cao, N. Hovakimyan, I. Kaminer, V. V. Patel, and V. Dobrokhodov, "Stabilization of cascaded systems via L1 adaptive controller with application to a UAV path following problem and flight test results," in *Proc. American Control Conf.*, New York, July 2007, pp. 1787–1792.

[31] A. P. Aguiar, I. Kaminer, R. Ghabcheloo, A. M. Pascoal, N. Hovakimyan, C. Cao, and V. Dobrokhodov, "Coordinated path following of multiple UAVs for time-critical missions in the presence of time-varying communication topologies," in *Proc. 17th IFAC World Congr.*, Seoul, South Korea, 2008, pp. 16,015–16,020.

[32] I. Kaminer, O. Yakimenko, V. Dobrokhodov, A. Pascoal, N. Hovakimyan, C. Cao, A. Young, and V. Patel, "Coordinated path following for time-critical missions of multiple UAVs via L1 adaptive output feedback controllers," presented at the AIAA Guidance, Navigation Control Conf. Exhibit, 2007, Paper AIAA 2007-6409.

[33] J. E. da Silva and J. B. de Sousa, "A dynamic programming approach for the motion control of autonomous vehicles," in *Proc. 49th IEEE Conf. Decision Control*, Dec. 2010, pp. 6660–6665.

[34] S. Shehab and L. Rodrigues, "Preliminary results on UAV path following using piecewise-affine control," in *Proc. IEEE Conf. Control Applications*, Aug. 2005, pp. 358–363.

[35] D. A. Lawrence, E. W. Frew, and W. J. Pisano, "Lyapunov vector fields for autonomous UAV flight control," presented at the AIAA Guidance, Navigation Control Conf. Exhibit, Hilton Head, SC, Aug. 2007, Paper AIAA 2007-6317.

[36] H. Chen, K. C. Chang, and C. S. Agate, "Tracking with UAV using tangent-plus-Lyapunov vector field guidance," in *Proc. 12th Int. Conf. Information Fusion*, 2009, pp. 363–372.

[37] I. Kaminer, A. Pascoal, E. Hallberg, and C. Silvestre, "Trajectory tracking for autonomous vehicles: An integrated approach to guidance and control," *J. Guid. Control Dyn.*, vol. 21, no. 1, pp. 29–38, 1998.

[38] E. Xargay, V. Dobrokhodov, I. Kaminer, A. M. Pascoal, N. Hovakimyan, and C. Cao, "Time-critical cooperative control of multiple autonomous vehicles: Robust distributed strategies for path-following control and time-coordination over dynamic communications networks," *IEEE Control Syst.*, vol. 32, no. 5, pp. 49–73, 2012.

[39] Kestrel autopilot. (2013). [Online]. Available: http://www.lockheedmartin.com/us/products/procerus/kestrel.html

[40] Piccolo autopilot. (2013). [Online]. Available: http://www.cloudcaptech.com

[41] Ardupilot autopilot. (2013). [Online]. Available: http://www.diydrones.com

[42] Paparazzi autopilot. (2013). [Online]. Available: http://paparazzi.enac.fr

[43] Openpilot autopilot. (2013). [Online]. Available: http://openpilot.org

[44] P. B. Sujit, S. Saripalli, and J. B. Sousa, "An evaluation of UAV path-following algorithms," in *Proc. European Control Conf.*, Zurich, Switzerland, July 17–19 2013, pp. 3332–3337.

[45] M. Kothari and I. Postlethwaite. "A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees," *J. Intell. Robot. Syst.*, vol. 71, no. 2, pp. 231–253, Aug. 2013.

[46] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*, 1st ed. Princeton, NJ: Princeton Univ. Press, 2012.

[47] L. Ljung, *System Identification*. New York: Wiley, 1999.

[48] N. Slegers, J. Kyle, and M. Costello, "Nonlinear model predictive control technique for unmanned air vehicles," *J. Guid. Control Dyn.*, vol. 29, no. 5, pp. 1179–1188, 2006.

[49] Y. Kang and J. K. Hedrick, "Linear tracking for a fixed-wing UAV using nonlinear model predictive control," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1202–1210, 2009.