Robotica (2018) volume 36, pp. 275–297. © Cambridge University Press 2017
doi:10.1017/S026357471700039X

A strategy for safe 3D navigation of non-holonomic robots among moving obstacles

Chao Wang†*, Andrey V. Savkin† and Matthew Garratt‡

†*School of Electrical Engineering and Telecommunications,*
The University of New South Wales, Sydney 2052, Australia. E-mail: a.savkin@unsw.edu.au
‡*School of Engineering and Information Technology,*
The University of New South Wales, Canberra 2600, Australia. E-mail: m.garratt@adfa.edu.au

(Accepted September 30, 2017. First published online: November 10, 2017)

SUMMARY
A non-holonomic robot with a bounded control input travels in a dynamic unknown 3D environment with moving obstacles. We propose a 3D navigation strategy to reach a given final destination point while avoiding collisions with obstacles. A formal analysis of the proposed 3D robot navigation algorithm is given. Computer simulation results and experiments with a real flying autonomous vehicle confirm the applicability and performance of the proposed guidance approach.

KEYWORDS: Guidance; 3D navigation; Moving obstacles; Sense-and-avoid; Collision avoidance; Flying robots; Underwater robots; Obstacle avoidance; Non-holonomic robots; Navigation in dynamic unknown environments.

1. Introduction

It is expected that unmanned aircraft vehicles and underwater robots will displace manned aircraft and submarines in diverse activities such as remote sensing, border patrol, aerial and underwater photography, environmental monitoring, package delivery, imaging and inspecting different kinds of underwater structures and military operations. In these applications, the problem of avoiding collisions with moving obstacles is extremely important since the autonomous vehicle operates in unknown and potentially dynamic environments cluttered with numerous moving obstacles, and therefore is more exposed to collisions. This problem is especially difficult in non-cooperative scenarios where the robot has no prior information on the future trajectory of a moving obstacle and cooperative communication between them is unavailable. The problem of non-cooperative collision avoidance (sense-and-avoid) has been identified as one of the most significant challenges facing the integration of unmanned aircraft vehicles in national airspaces.[1]

Collision free autonomous navigation among moving obstacles is a classic research area in robotics, which gives rise to a whole variety of approaches; see e.g., refs. [2–6] and references therein. However, a vast majority of the existing research in this area addresses the case of planar vehicles. Few papers studied the much more difficult case of collision free 3D navigation; however, various methods of collision free navigation were recently proposed for various types of underwater vehicles[7,8] and unmanned aircraft in refs. [9–15] . Furthermore, effective algorithms of navigation of underwater and surface vehicles with uncertain dynamics were developed in refs. [16, 17].

This paper addresses the problem of driving a single mobile under-actuated non-holonomic robot in a three-dimensional workspace to a given destination point while avoiding collisions with moving obstacles. We propose a sense-and-avoid type navigation algorithm. The performance of the proposed 3D robot navigation algorithm is verified by computer simulations and experiments with a real flying robot. Furthermore, we compare our three-dimensional guidance strategy with other 3D autonomous navigation methods such as the Yang–Alvarez–Bruggemann algorithm[14] and the Shim–Sastry algorithm.[15] Moreover, unlike many other papers on this topic, we present a formal proof of the

* Corresponding author. E-mail: z3184703@zmail.unsw.edu.au

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

performance of the proposed navigation algorithm. Rigorously proved collision avoidance is a highly desirable trait in autonomous navigation systems since it shows that the autonomous navigator safely operates under a broad range of conditions, rather than just those considered during testing. The safe navigation algorithm of this paper is applicable to both unmanned aircraft vehicles and underwater robots.

The remainder of the paper is organized as follows. Section 2 describes the mathematical model of an under-actuated non-holonomic 3D robot and its 3D environment and states the problem under consideration. In Section 3, the proposed safe robot navigation strategy is introduced. Computer simulations of the proposed 3D robot navigation algorithm are given in Section 4, whereas Section 5 reports experiments with a real flying robot (a commercial Parrot AR Drone 2.0 quadrotor). A formal analysis of the robot navigation algorithm is given in Section 4. Finally, Section 7 presents a brief conclusion and directions of future research.

2. Problem Statement

In this paper, we study a three-dimensional non-holonomic autonomous vehicle or mobile robot. The mathematical model of the 3D robot is as follows. Let

$$c(t) := [x(t), y(t), z(t)] \tag{1}$$

be the three-dimensional vector of the robot's Cartesian coordinates. Then, the motion of the robot is described by the equations:

$$\dot{c}(t) = V(t)\tilde{v}(t), \tag{2}$$

$$\dot{\tilde{v}}(t) = a(t) \tag{3}$$

where $\tilde{v}(t) \in \mathbf{R}^3$, $\|\tilde{v}(t)\| = 1$ for all t, $a(t) \in \mathbf{R}^3$, $V(t) \in \mathbf{R}$, and the following constraints hold:

$$\|a(t)\| \le A_{\max}, \quad V(t) \in [V_{\min}, V_{\max}], \tag{4}$$

$$(\tilde{v}(t), a(t)) = 0 \tag{5}$$

for all t. Here, $\|\cdot\|$ denotes the standard Euclidean vector norm, $(\cdot,\cdot)$ denotes the scalar product of two vectors, the constants $A_{\max}$, $V_{\min}$ and $V_{\max}$ are given. The scalar variable $V(t)$ and the vector variable $a(t)$ are control inputs, $V(t)$ is the speed or linear velocity of the 3D vehicle, $a(t)$ is applied to change the direction of the robot's motion. The condition (5) guarantees that the vectors $\tilde{v}(t)$ and $a(t)$ are always orthogonal. Furthermore, let $v(t) := \dot{c}(t)$ denotes the velocity vector of the robot. Then, it is obvious that

$$\tilde{v}(t) = \frac{1}{\|v(t)\|} v(t) \tag{6}$$

The kinematics of many unmanned aerial and underwater vehicles can be described by the non-holonomic model (2), (3), see, e.g., ref. [18] and references therein.

In this paper, we study a quite general three-dimensional problem of autonomous vehicle navigation with collision avoidance. The robot operates in an unknown uncertain dynamic three-dimensional environment with a number of stationary or moving obstacles. In particular, we assume that there are several disjoint moving obstacles $1, 2, \ldots, k$ and a stationary final destination point $\mathbf{F}$ in the three-dimensional space. The objective is to drive the robot to this final destination point while avoiding collisions with the moving obstacles $1, 2, \ldots, k$. We suppose that any obstacle i is moving in the three-dimensional space. We consider the most general case of arbitrary 3D motions of obstacles including any rotations. Obstacles may have any irregular and non-convex shapes. Moreover, the obstacles can be deforming. However, we assume that the obstacle is always inside of some moving sphere of a known radius R_i constructed by the robot's navigation system. This sphere is said to be the *covering sphere of the obstacle i*. Moreover, the robot knows the direction $D_i(t) \in \mathbf{R}^3$ from the

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

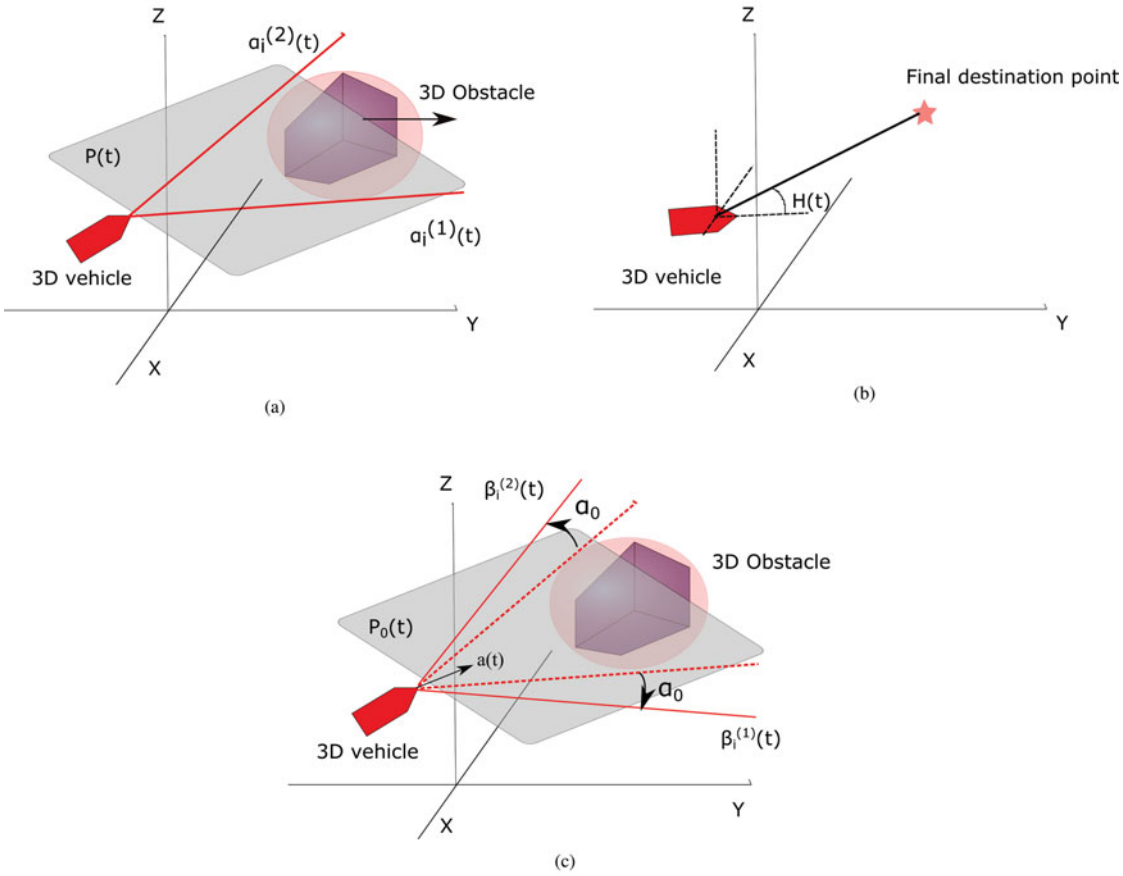


Fig. 1. (a) Covering sphere of the obstacle i; (b) Heading $H(t)$ to the final destination point; (c) Enlarged vision cone.

robot's coordinates to the center of this sphere and the current minimum distance $d_i(t)$ to the covering sphere. Also, the robot has access to $\dot{d}_i(t)$. Furthermore, the center of the sphere is moving with the velocity $v_i(t)$, which is available to the robot. We also assume that velocities $v_i(t)$ satisfy the following constraint:

$$\|v_i(t)\| \leq V_o < V_{\max} \tag{7}$$

for some $V_o > 0$ and all i, t. It is obvious that if (7) does not hold, the collision avoidance may be impossible. Also, for any plane $\mathcal{P}$ going through the robot's coordinates and intersecting the obstacle i, the robot measures the angles $\alpha_i^{(1)}(t)$ and $\alpha_i^{(2)}(t)$, which are made by the tangent straight lines in the plane $\mathcal{P}$ from the robot's current position to the covering sphere of the obstacle i; see Fig. 1(a). These lines define a vision cone of the covering sphere from the robot's current location. Finally, the robot knows the heading $H(t) \in \mathbf{R}^3$ to the final destination point $\mathbf{F}$; see Fig. 1(b).

Definition 2.1. Let $d_{\text{safe}} > 0$ be a given constant. A robot navigation strategy is said to be *destination reaching with collision avoidance* if there exists a time $t_f > 0$ such that $c(t_f) = \boldsymbol{F}$ and $d_i(t) \geq d_{safe}\ \forall i = 1, 2, \ldots, k, \quad \forall t \in [0, t_f]$.

The objective of this paper is to develop a destination reaching with collision avoidance navigation strategy for 3D non-holonomic robots.

3. Robot Navigation Strategy

For avoiding a collision of the navigated 3D robot with the moving obstacle i, we will use the following collision avoidance strategy. Let $\mathcal{P}_0(t)$ be the plane going through the robot's coordinates and defined

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

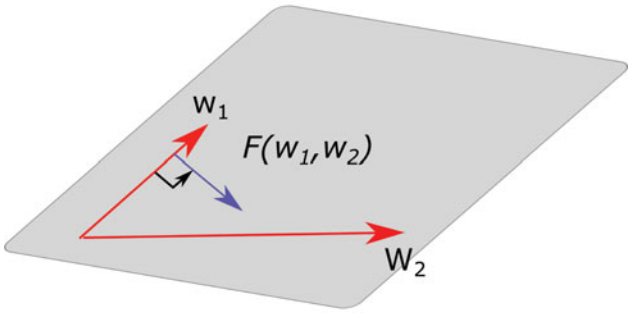


Fig. 2. The vector $F(w_1, w_2)$.

by the vectors $\tilde{v}(t)$ and $D_i(t)$, where $\tilde{v}(t)$ is the robot's motion direction and $D_i(t)$ is the direction from the robot's coordinates to the center of the sphere containing the obstacle i, see Fig. 1(c). Furthermore, let α_0 be a given angle, $0 < \alpha_0 < \frac{\pi}{2}$. Let

$$\beta_i^{(1)}(t) := \alpha_i^{(1)}(t) + \alpha_0, \quad \beta_i^{(2)}(t) := \alpha_i^{(2)}(t) - \alpha_0; \tag{8}$$

see Fig. 1(c). Furthermore, introduce the vectors $l_i^{(1)}$ and $l_i^{(2)}$ as

$$l_i^{(j)}(t) := (V_{\max} - V_o)[\cos(\beta_i^{(j)}(t)), \sin(\beta_i^{(j)}(t))] \quad \forall j = 1, 2 \tag{9}$$

where $V_{\max}$ and V_o are the constants from (4) and (7), respectively. A geometric interpretation of (9) is straightforward. Equation (9) defines the two lines that are the boundaries of an enlarged vision cone of the covering sphere as it is seen from the robot's current position.

Let w_1 and w_2 be non-zero three-dimensional vectors, and $\|w_1\| = 1$. Now introduce the following function $F(\cdot, \cdot)$ mapping from $\mathbf{R}^3 \times \mathbf{R}^3$ to $\mathbf{R}^3$ as

$$F(w_1, w_2) := \begin{cases} 0 & f(w_1, w_2) = 0 \\ \|f(w_1, w_2)\|^{-1} f(w_1, w_2) & f(w_1, w_2) \neq 0 \end{cases}$$
$$f(w_1, w_2) := w_2 - (w_1, w_2)w_1. \tag{10}$$

In other words, the rule (10) defined in the plane of vectors w_1 and w_2, a vector $F(w_1, w_2)$ that is orthogonal to w_1 and directed "towards" w_2, see Fig. 2. It is also clear that $F(w_1, w_2) = 0$ if w_1 and w_2 are co-linear, and $\|F(w_1, w_2)\| = 1$ otherwise.

We start our maneuver to avoid the moving obstacle i at the time t_0. First, among the vectors $v_i(t_0) + l_i^{(j)}(t_0)$, $j = 1, 2$, we choose the one that makes the smallest angle between itself and the robot's motion direction vector $\tilde{v}(t_0)$. Let $h \in \{1, 2\}$ be the index of such a vector. Now, we propose the following navigation law:

$$a(t) = A_{\max} F(a(t), v_i(t) + l_i^{(h)}(t));$$
$$V(t) = \|v_i(t) + l_i^{(h)}(t)\|. \tag{11}$$

Remark 3.1. It is clear that the control inputs defined by (11) satisfy the constraints (4). Indeed, (10) implies that $|a(t)| \leq A_{max}$. Furthermore, (11), (7) and (9) imply that

$$V(t) = \|v_i(t) + l_i^{(h)}(t)\| \leq \|v_i(t)\| + \|l_i^{(h)}(t)\|$$
$$\leq V_o + (V_{\max} - V_o) = V_{\max}.$$

Therefore, the second constraint (4) holds too.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

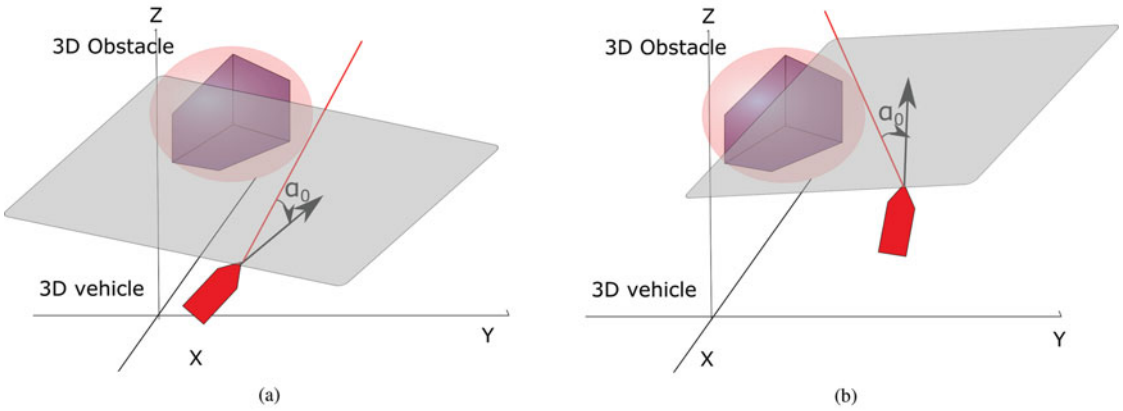


Fig. 3. Constant avoiding angle α_0 between the instantaneous moving direction of the robot and one of the two boundary rays.

Remark 3.2. The intuition behind the obstacle avoidance strategy (11) can be explained as follows. The navigation strategy (11) belongs to the class of sliding mode control algorithms; see e.g., ref. [19]. The algorithm (11) steers the motion direction vector $\tilde{v}(t)$ of the robot to a switching surface on which the robot's velocity $v(t)$ becomes equal to the vector $v_i(t) + l_i^{(h)}(t)$. Now, consider a coordinates system moving with the velocity $v_i(t)$. In these coordinates, the obstacle i becomes stationary, and the robot is moving with the velocity $l_i^{(h)}(t)$. Furthermore, the motion with the velocity $l_i^{(h)}(t)$ is an obstacle avoidance strategy, which keeps a constant avoiding angle α_0 between the instantaneous moving direction of the robot and one of the two boundary rays of the vision cone of the covering sphere of the obstacle i from the robot; see Fig. 3. In other words, the control law consists of steering towards the closer of the two edges of the enlarged covering sphere vision cone defined by (9) in the plane defined by the robot's velocity vector and the direction towards the center of the covering sphere of the obstacle.

Remark 3.3. In many other research papers, the authors replace the robot motion model (2), (3) by the following first-order model:

$$\dot{c}(t) = v(t) \tag{12}$$

where $c(t)$ is 3D coordinates of the robot, and the velocity vector $v(t)$ is the control input. In this case, the navigation law (11) can be replaced by the following navigation law:

$$v(t) = v_i(t) + l_i^{(h)}(t)). \tag{13}$$

Our proposed strategy consists of switching between the obstacle avoidance law (11) and the straight motion to the final destination point at the maximum speed:

$$a(t) = 0; \quad V(t) = V_{\max}. \tag{14}$$

Let $C > 0$ be a given number. Furthermore, introduce the constants $a_i > 0$ as

$$a_i := \frac{R_i}{\cos\alpha_0} - R_i \quad \forall i = 1, 2, \ldots, k \tag{15}$$

where R_i is the radius of the moving sphere covering the obstacle i. Specifically, the rule for switching between (14) and (11) is as follows:

R1: Switching from (14) to (11) occurs at any time t_0 when the distance from the robot to the obstacle i reduces to the value C, i.e., $d_i(t_0) = C$ and $\dot{d}_i(t_0) < 0$;

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

R2: Switching from (11) to (14) occurs at any time t_* when $d_i(t_*) \leq 1.1a_i$ and the vehicle is oriented towards the final destination point, i.e., the vehicle velocity vector $v(t)$ is co-linear to the heading to the final destination point $H(t^*)$.

Remark 3.4. The constant 1.1 in the control rule **R2** can be replaced by any quantity $(1 + \epsilon)$ where $\epsilon > 0$. With a larger ϵ, the algorithm becomes too conservative, i.e., the robot will keep a larger distance from the obstacle. When ϵ tends to 0, the robot can follow the boundary of the obstacle for a long time before switching towards the destination point.

It should be pointed out that the proposed navigation strategy is essentially three-dimensional, i.e., the robot does not avoid a moving obstacle in a horizontal plane but chooses the best in some sense avoiding trajectory in 3D. The proposed strategy can be viewed as an extension of the planar algorithm of ref. [3] to the much more difficult problem of 3D obstacle avoidance.

4. Computer Simulations

We illustrate the performance of the proposed navigation algorithm by computer simulations with MATLAB. In our computer simulations, we take the sampling period $\delta = 0.1$ s.

In the first simulation, we consider a simple scenario where the vehicle tries to avoid a single obstacle as shown in Fig. 4(a). In Fig. 4(b), we depict the plane $\mathcal{P}_0(t)$ and the angles $\alpha_i^{(1)}(t)$, $\alpha_i^{(2)}(t)$ that are the boundaries of the vision cone of the covering sphere. The proposed navigation algorithm enlarges these two angles using a constant angle using (8), finds the vectors $l_i^{(1)}$, $l_i^{(2)}$ using (9) and finally computes the proper control inputs with (11). Figure 4(c) shows the moment when the robot is oriented towards the final destination point and starts to move towards it safely. The vehicle safely and efficiently avoids the obstacles by a proper switching between (11) and (14). This simulation shows that the proposed navigation algorithm is able to maintain a safe distance between the vehicle and the obstacle.

In the next simulation, we apply the proposed navigation algorithm to a more complex scenario where the robot tries to reach the final destination point while avoiding multiple moving obstacles. Figure 5(a)–(c) shows the crucial moments when the vehicle avoids the obstacles. The vehicle avoids each of the obstacles in a similar fashion as shown in Fig. 4. Figure 5(d) shows the complete path taken by the vehicle in this simulation and Fig. 5(e) shows the projection of the path on X–Y plane. These simulation results again confirm that the proposed navigation algorithm is capable of keeping the vehicle at a safe distance to the obstacle even in complicated dynamic environments with multiple moving obstacles. Figure 6 reports the result of another simulation with a similar multiple moving obstacles scenario. This simulation resembles a race scenario where the vehicle starts behind all the moving obstacles and moves towards the final destination point while avoiding collisions with the moving obstacles.

In the real world scenarios, the shapes of the obstacles are not always spherical as we have seen in the previous simulation results. In these cases, we need to approximate the shapes of the obstacles by a sphere. A polyhedron is an example of an obstacle of a non-spherical shape as shown in Fig. 7(a). We cover the polyhedron by a sphere in Fig. 7(b). In cases where the dimensions of the obstacle are not known prior to the execution of the proposed navigation algorithm, we can approximate the size of the obstacle by maximum width of the obstacle on the plane $\mathcal{P}_0(t)$ as shown in Fig. 7(c). Figure 7(d) shows the complete path taken by the vehicle in this simulation. We extend the simulation to a multiple moving obstacles scenario as shown in Fig. 8.

Finally, we compare the performance of the proposed navigation algorithm with that of some other existing 3D navigation algorithms. In ref. [14], the authors propose a navigation algorithm that avoids a moving obstacle by maintaining a relative angles in both a horizontal plane and a vertical plane. This avoidance strategy steers the air vehicle far away from the obstacle in order to avoid colliding with the obstacle, on the other hand, the proposed navigation algorithm efficiently uses the space to steer the heading of the air vehicle to a safe direction and an efficient distance to the obstacle. The performance comparison of the two navigation algorithms is shown in Fig. 9. We also compare our proposed navigation algorithm to the algorithm proposed in ref. [15]. The navigation algorithm of ref. [15] utilizes model predictive control (MPC) to compute the optimal control sequence over the finite receding horizon according to a specific cost function. Figure 10 shows a scenario in which the

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

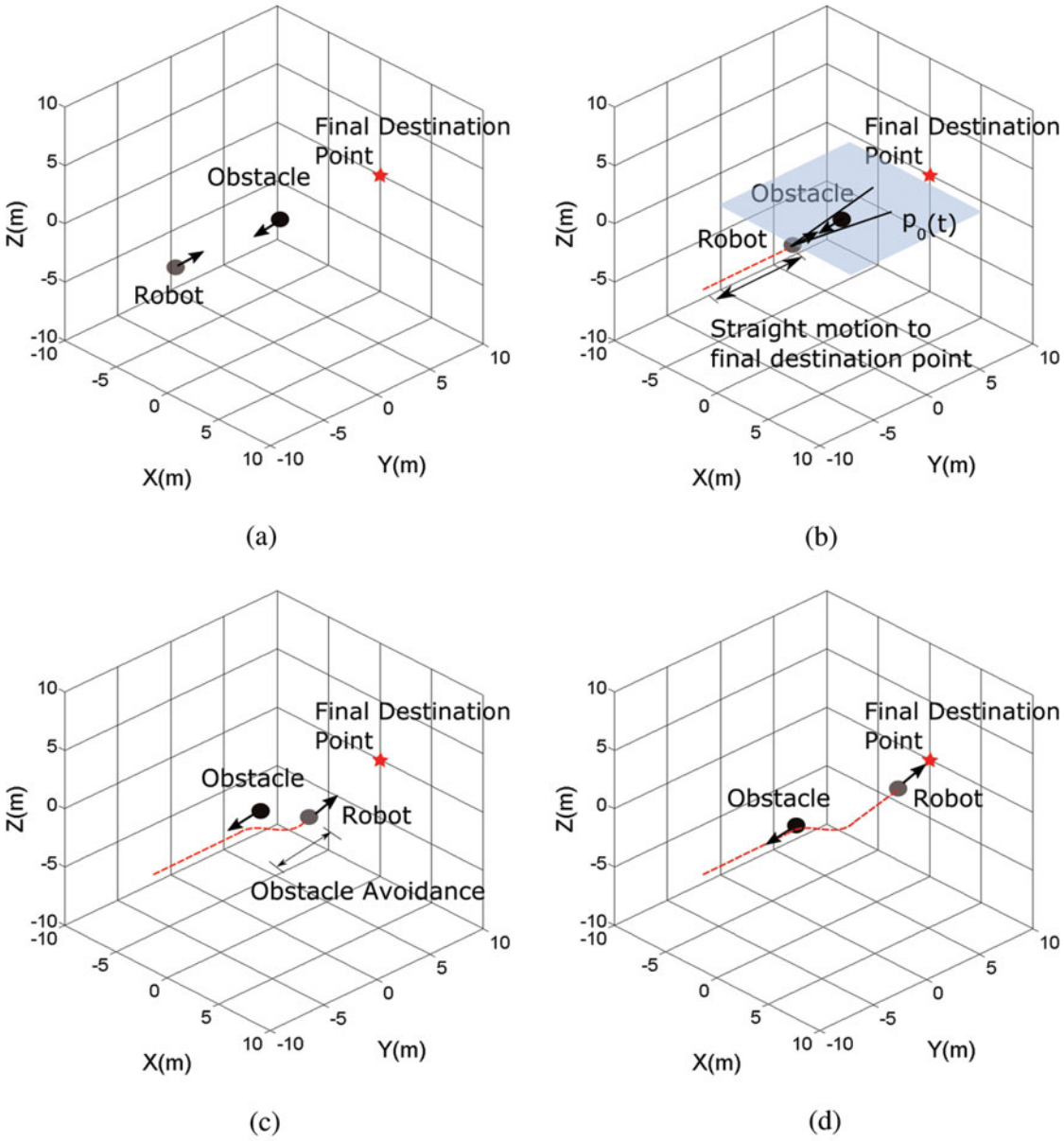


Fig. 4. Vehicle avoids single obstacle. (a) Initial setup of the simulation. (b) Switching from (14) to (11). (c) Switching from (11) to (14).

proposed navigation algorithm efficiently avoids the obstacle whereas the algorithm of ref. [15] waits for the obstacle to pass and then presumes moving towards the final destination point which is a less efficient avoidance maneuver. As a result, for avoidance of a single obstacle, the proposed navigation algorithm outperforms the algorithms from both papers. The ratio of the maneuver times is as follows: for the Yang–Alvarez–Bruggemann algorithm of ref. [14] $\frac{T_{\text{Proposed}}}{T_1} = \frac{12.1}{14.3} \approx 84.6\%$. Furthermore, for the Shim–Sastry algorithm of ref. [15], the ratio is $\frac{T_{\text{Proposed}}}{T_1} = \frac{11.4}{13.9} \approx 82.0\%$.

5. Experiments with a Real Flying Robot

We implement the proposed navigation algorithm with the commercial quadrator Parrot AR Drone 2.0, see Fig. 11 (see ref. [20] for detailed specification). The results of real world experiments are presented in this section.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

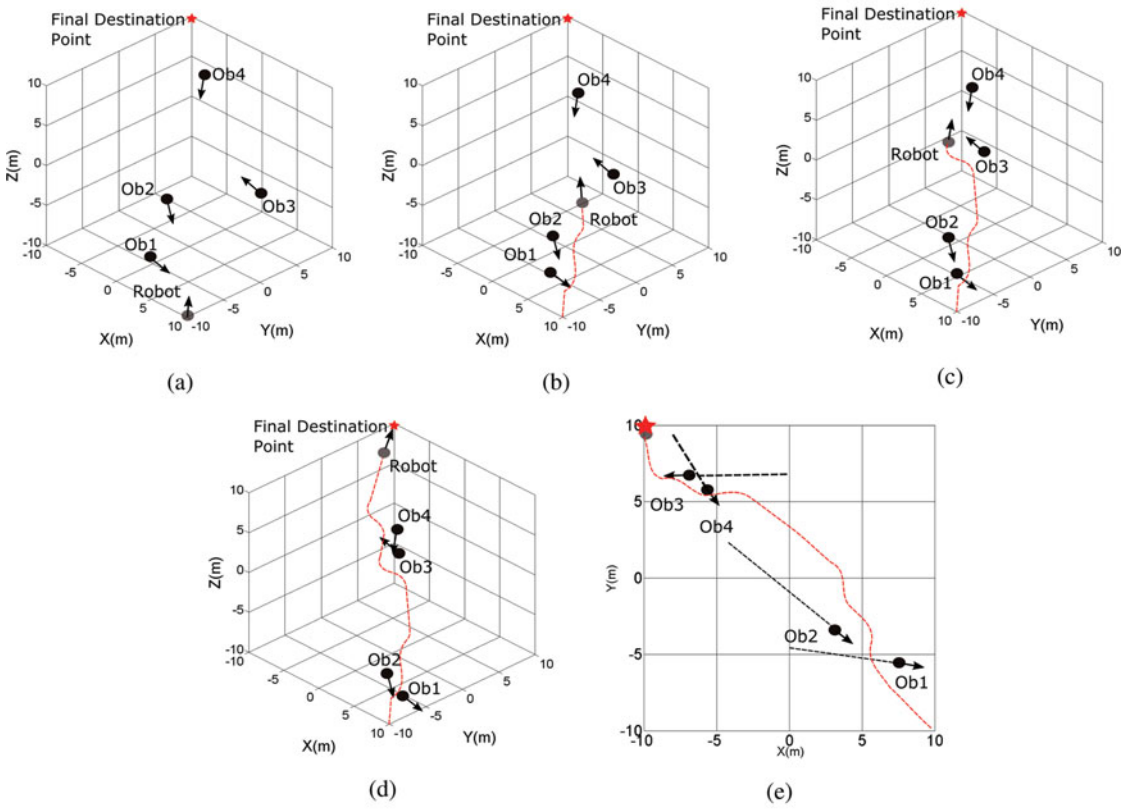


Fig. 5. Vehicle avoids multiple obstacles.

The AR Drone 2.0 is equipped with an on-board processor (32 bit ARM Cortex A8 processor running Linux) and various on-board sensors. It features two cameras, one forward facing camera that is used to detect the en-route obstacles and one downward-facing camera. The AR Drone is also equipped with an IMU which allows it to measure the velocity, acceleration and the orientation of the drone. The sensory data are sent to a laptop (running Ubuntu with dual core 2.66GHz CPU) via WiFi. The laptop computes the proper control commands by using the sensory data and then send the control commands back to the AR Drone. The AR Drone is equipped with two downward facing ultrasound sensors for ground altitude measurement (i.e., z values in $[x,y,z]$ coordinates). The downward facing camera allows to estimate the relative movement of the AR Drone with respect to its local origin using optical flow (i.e., x and y values in $[x,y,z]$ coordinates). The AR Drone can reasonably well estimate its position during the experiments by using the combination of these two types of sensors.

The ROS (Robot Operating System) driver (ardrone_autonomy see ref. [21]) for AR Drone 2.0 is designed so that the AR Drone is able to detect certain tags using its forward-facing camera. The roundel tag is one of the detectable tags, see Fig. 12. The ROS driver allows the user to obtain various distance and relative position information regarding the tags, which allows the users to estimate the position of the tag in 3D environment and perform various control over the AR Drone, see refs. [22, 23], etc. In the following experiments, we put five tags on one obstacle: one on the center to estimate the position of the center of mass, four on the corners of the obstacle estimate the size of the obstacle, see Fig. 13. Furthermore, the ROS driver allows the user to send velocity control commands to the AR Drone. In the experiments, we integrate the control commands (accelerations in three axes) computed by (11) to obtain the desired velocities.

We start with a simple scenario where the obstacles are stationary. In this experiment, stationary obstacles with various dimensions are placed at random positions in the environment, as shown in Fig. 14(a). Figure 14(b)–(d) shows the crucial moments when the AR Drone avoids each of the obstacles under the guidance of the proposed navigation algorithm. Figure 14(e)–(g) shows the snapshots from the front camera of the AR Drone when it detects and starts to avoid the closest

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

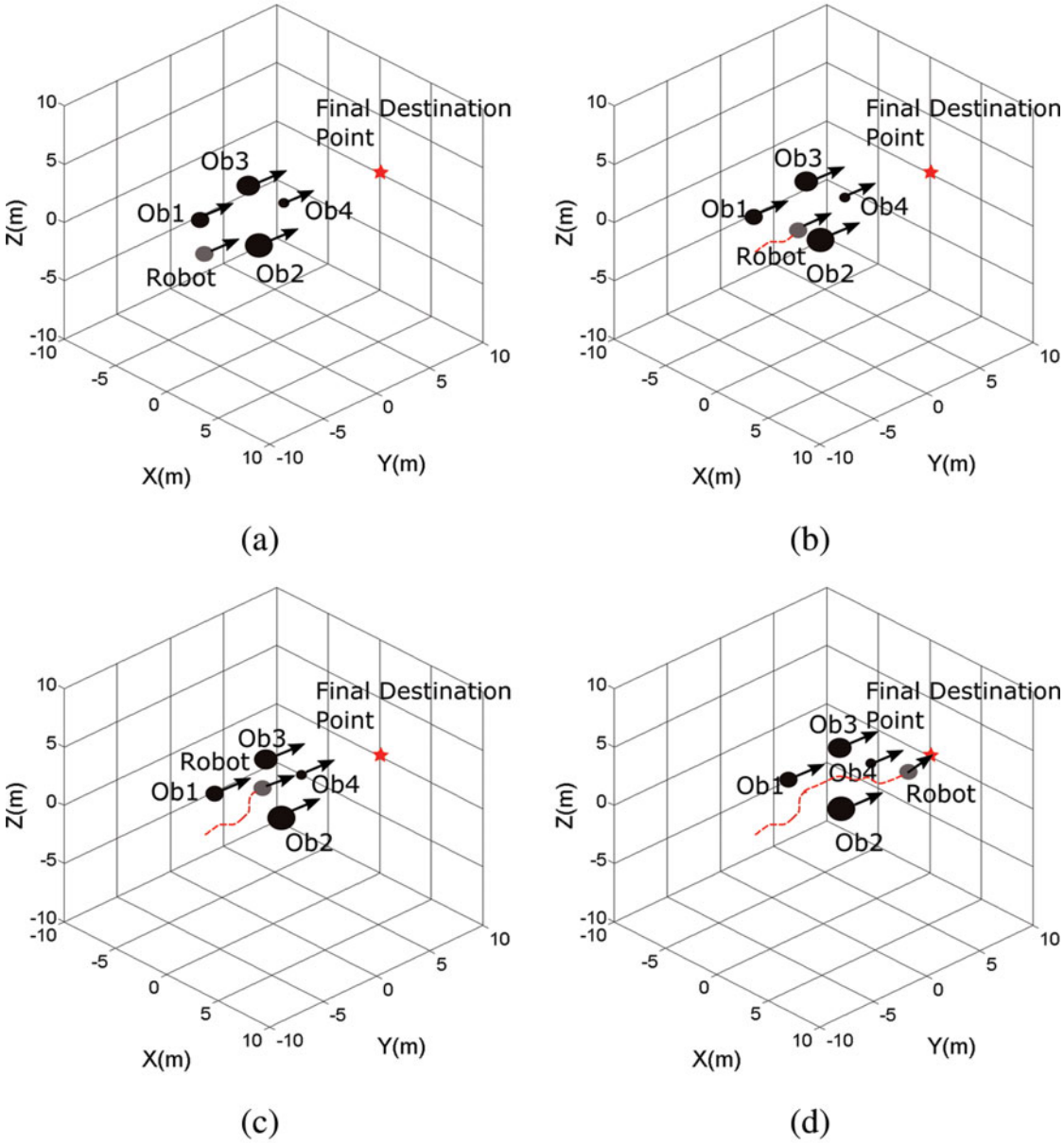


Fig. 6. Vehicle avoids multiple obstacles, case 2.

obstacle. Figure 14(h) shows the complete path taken by the AR Drone to avoid all the obstacles and reach the final destination point. The black dots shown in the figure depicts the estimated locations of the tags. Figure 15 shows the graphs of recorded speed data of the AR Drone in x, y and z directions throughout the experiment.

In the next experiment, we examine the performance of the proposed navigation algorithm in the real world environment with moving obstacles. When the AR Drone detects an obstacle within the distance C, it starts to avoid the obstacle under the guidance of the proposed navigation algorithm. It estimates the position of the center of mass of the closest obstacle, the plane $\mathcal{P}_0(t)$ and the boundaries of the vision cone of the covering sphere of the obstacle i from the AR Drone. The proposed navigation algorithm uses this information to compute the control inputs according to (8), (9) and (11). Figure 16(b) and (c) shows the moments when the AR Drone avoids each of the obstacles, whereas Fig. 16(d) and (e) shows the snapshots from the front camera of the AR Drone when it detects and starts to avoid the closest obstacle. Figure 16(f) shows the complete path taken by the AR Drone throughout the experiment. The circles indicate the initial positions of tags. The solid black dots depict the state of

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

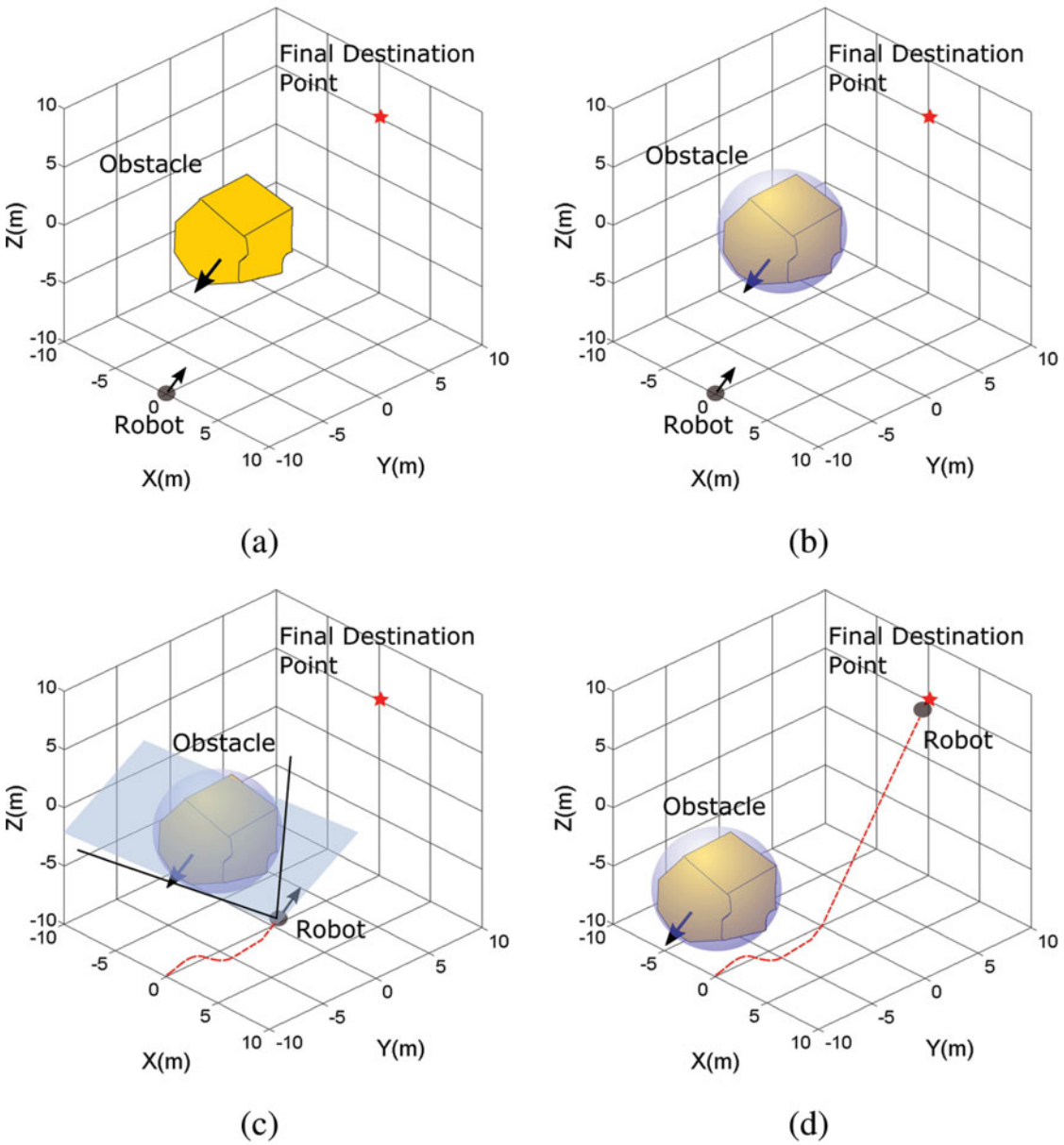


Fig. 7. Vehicle avoids single irregular polyhedron.

the obstacles when the AR Drone detects them. We also depict the estimated plane $\mathcal{P}_0(t)$ and the estimated size of the obstacle (shown as circles on the plane $\mathcal{P}_0(t)$). We repeat our experiment with obstacles moving with different velocities, the results are shown in Figs. 17–19. It can be concluded, from these real world experiment results, that the proposed navigation algorithm is able to ensure the safety of the AR Drone among multiple en-route moving obstacles and safely drive the AR Drone to the final destination point.

In real world scenarios, the obstacles may move with more complicated trajectories. Thanks to the fact that the proposed navigation algorithm takes into account the instantaneous moving direction $v_i(t)$ of the obstacle when computing the control signal, it always keeps a constant avoiding angle between the current moving direction of the AR Drone and the boundary of the enlarged vision cone of the covering sphere, meaning that the proposed navigation algorithm steers the moving direction of the AR Drone so that it will never collide with the obstacle in the future. Therefore, the proposed navigation algorithm is able to ensure the safety of the AR Drone even when the obstacle is moving with a quite complicated trajectory. The result of the experiment is shown in Fig. 20.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

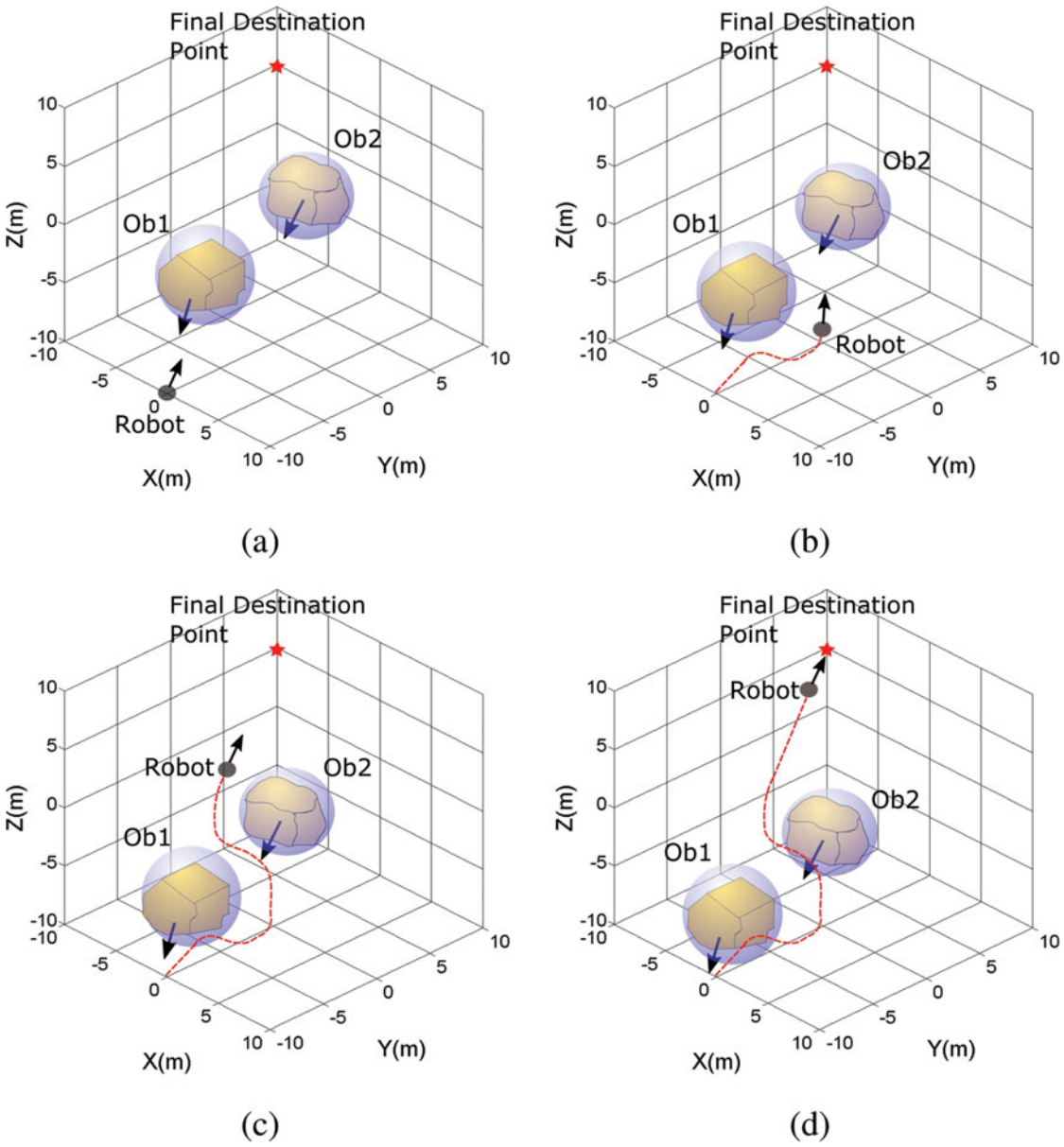


Fig. 8. Vehicle avoids multiple irregular polyhedron.

Remark 5.1. In our experiments, we placed tags on obstacles. This approach is not applicable to real-life situations. In real-life applications, we need to calculate the vision cone of an obstacle. This vision cone can be obtained by using laser range scanners. The laser range scanner will provide distance to the obstacles in a wide range of directions, the vision cone is simply the two boundary rays of the directions that intersect the obstacle in a plane (see illustration in Fig. 1(a)) assuming that the laser range scanner is aligned with the heading of the robot. Obstacle avoidance algorithms were implemented with laser range scanners in real life situations in refs. [3, 5, 6], where SICK laser range scanners were used for navigation of Pioneer 3-DX mobile wheeled robots, and Hokuyo URG laser range scanners were used for navigation of autonomous wheelchairs and autonomous mobile hospital beds. To implement the proposed obstacle avoidance algorithm, the problem of distinguishing the obstacle from the background should be solved using a laser range scanner. Furthermore, in some three-dimensional real-life applications, the covering sphere can be estimated using a 3D laser range scanner. Of course, 3D laser range scanners are more expensive than 2D laser range scanners.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

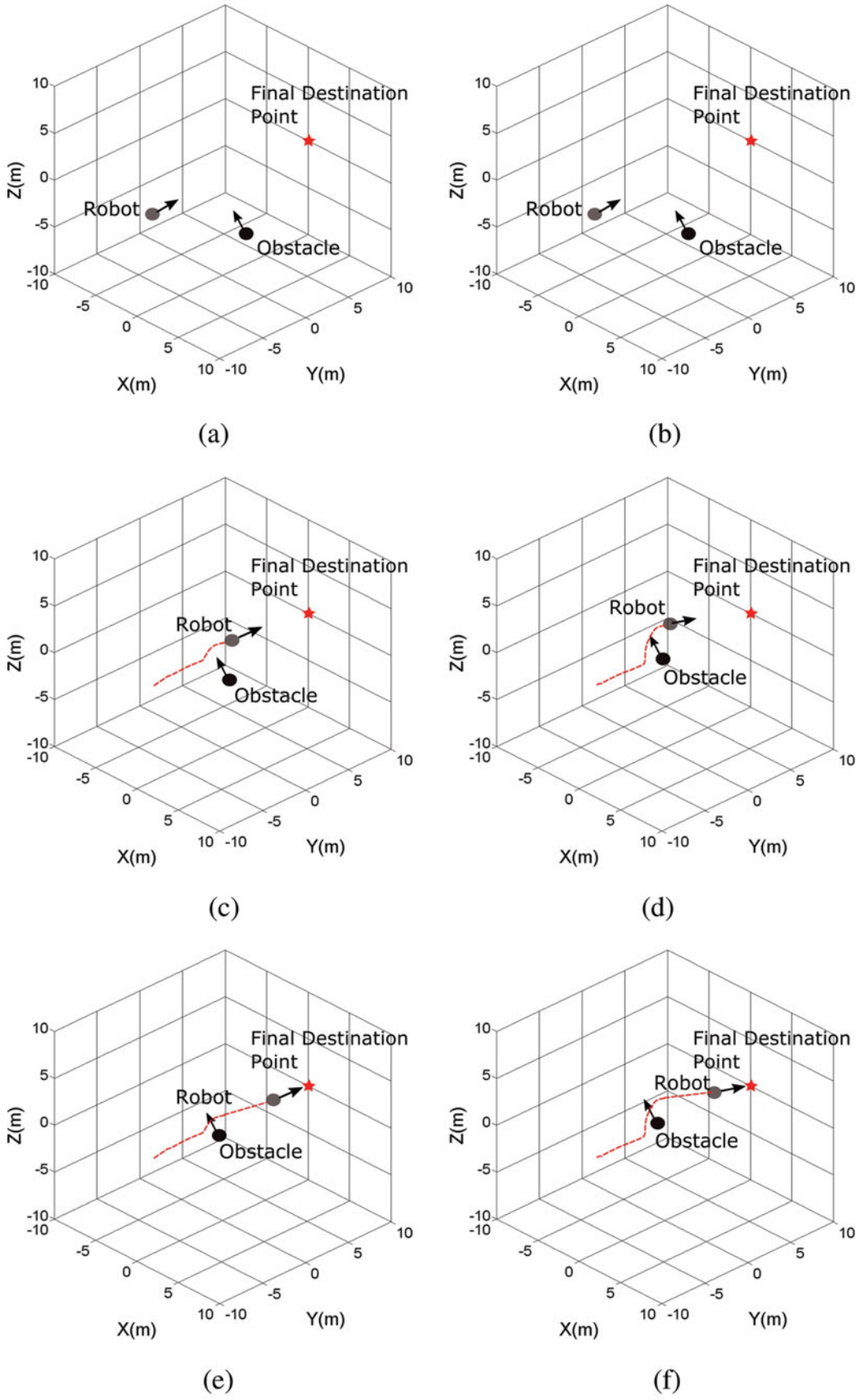


Fig. 9. Performance comparison between the proposed navigation algorithm and the algorithm in ref. [14]. (a) The proposed algorithm. (b) The Yang–Alvarez–Bruggemann algorithm. (c) The proposed algorithm. (d) The Yang–Alvarez–Bruggemann algorithm. (e) The proposed algorithm. (f) The Yang–Alvarez–Bruggemann algorithm.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

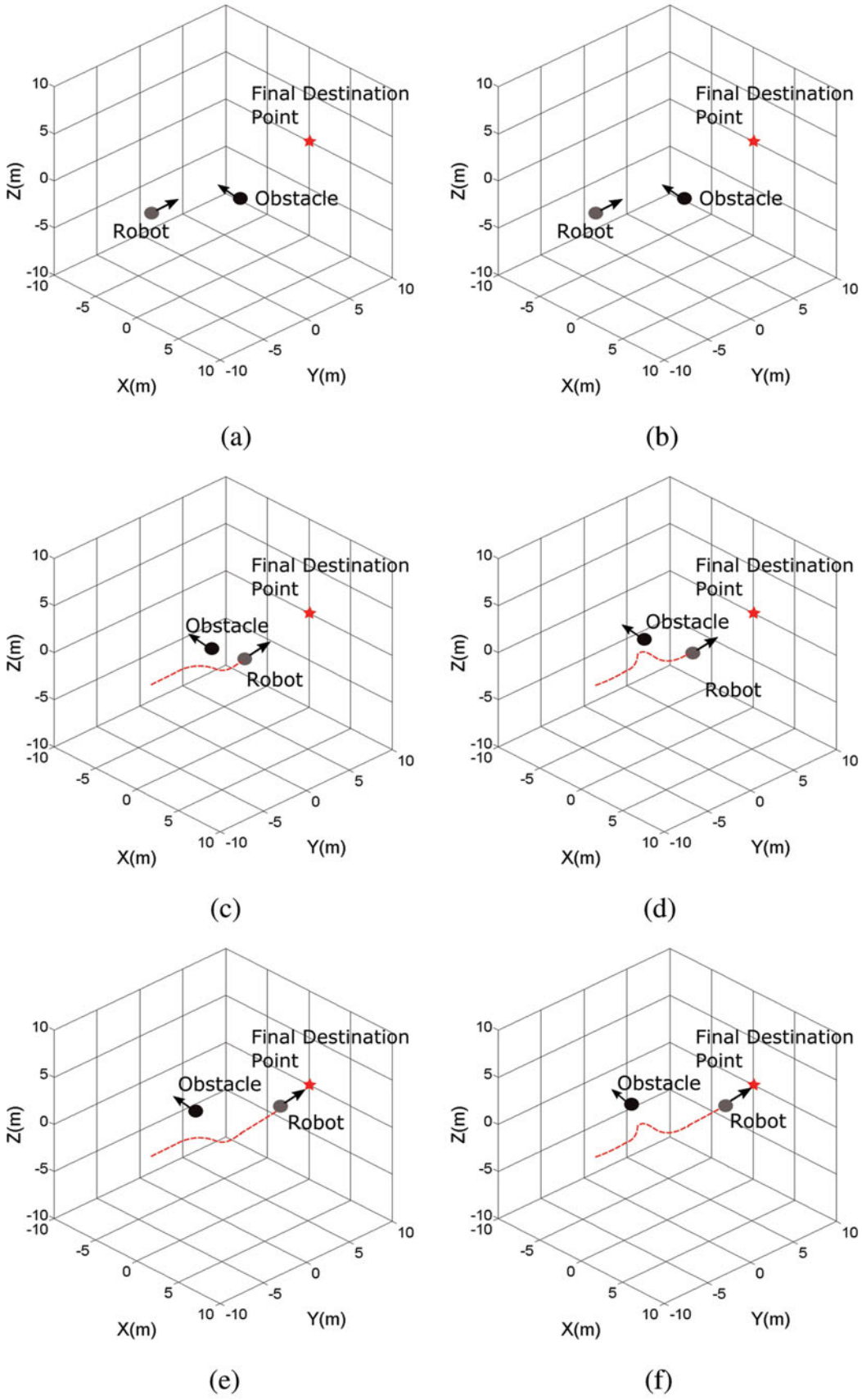


Fig. 10. Performance comparison between the proposed navigation algorithm and the algorithm in ref. [15]. (a) The proposed algorithm. (b) The Shim–Sastry algorithm. (c) The proposed algorithm. (d) The Shim–Sastry algorithm. (e) The proposed algorithm. (f) The Shim–Sastry algorithm.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press



(a)



(b)

Fig. 11. Top and bottom picture of AR Drone 2.0. (a) Top view. (b) Bottom view.



Fig. 12. Roundel tag.



Fig. 13. Tags on an obstacle.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

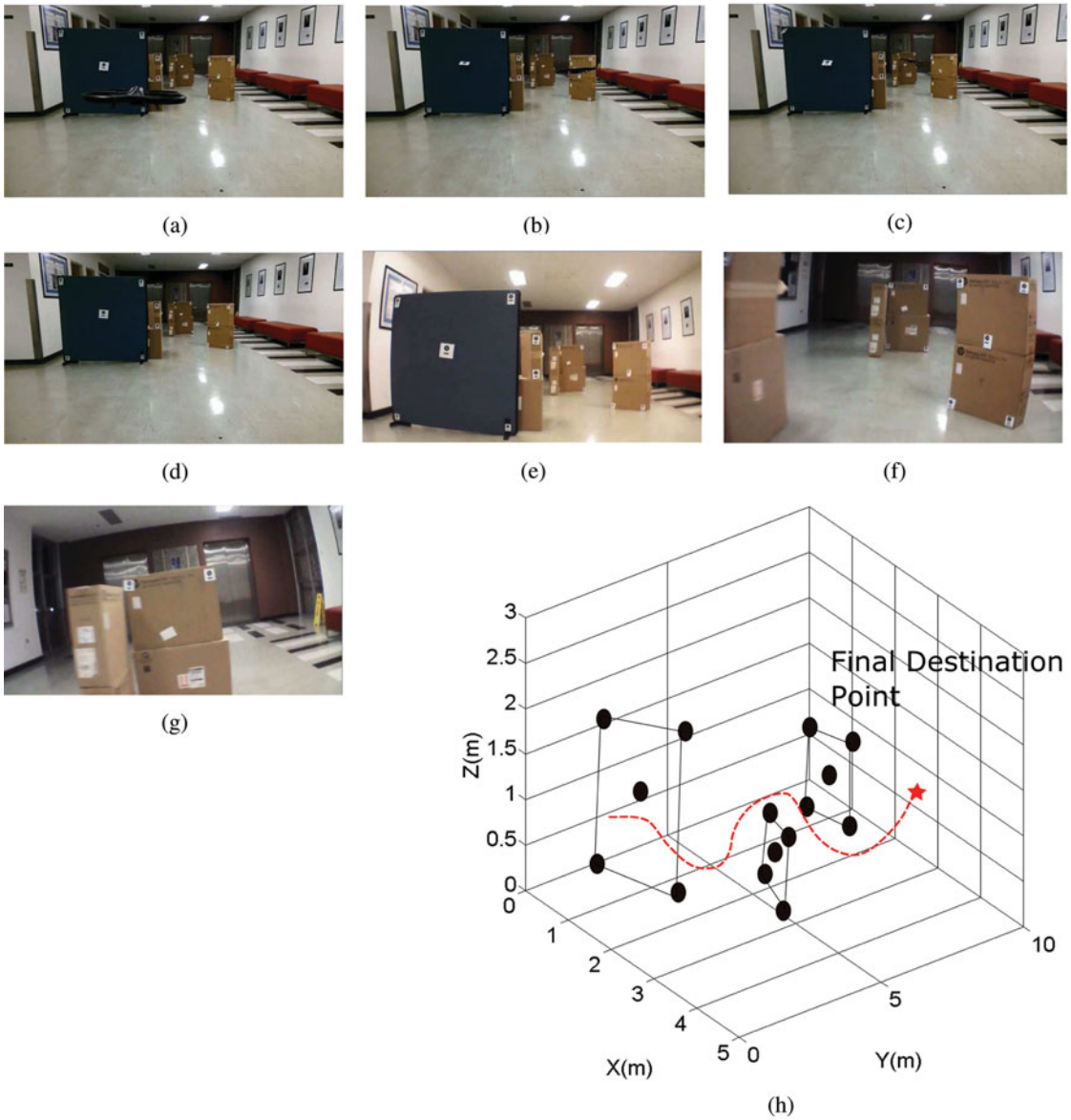


Fig. 14. AR Drone avoids multiple static obstacles.

6. Formal Analysis of the Robot Navigation Algorithm

In this section, we prove that under a number of quite detailed technical assumptions, that the proposed navigation strategy **R1, R2** is destination reaching with collision avoidance (see Definition 2.1).

Our main theoretical result requires the following assumptions.

Assumption 6.1. The center of the covering sphere of every obstacle i is moving with the constant velocity $v_i \neq 0$. Furthermore, $V_i < V_{max} \quad \forall i = 1, 2, \ldots, k$, where $V_i := \|v_i\|$, and the vectors v_i and v_j are not collinear for all $i \neq j$.

Remark 6.1. Assumption 6.1 does not mean that every obstacle is moving with a constant velocity. Any obstacle can perform irregular rotational motions inside of its covering sphere. Moreover, the obstacles can be deforming, i.e., change their shapes and sizes. Also, this assumption includes any stationary bounded obstacle.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

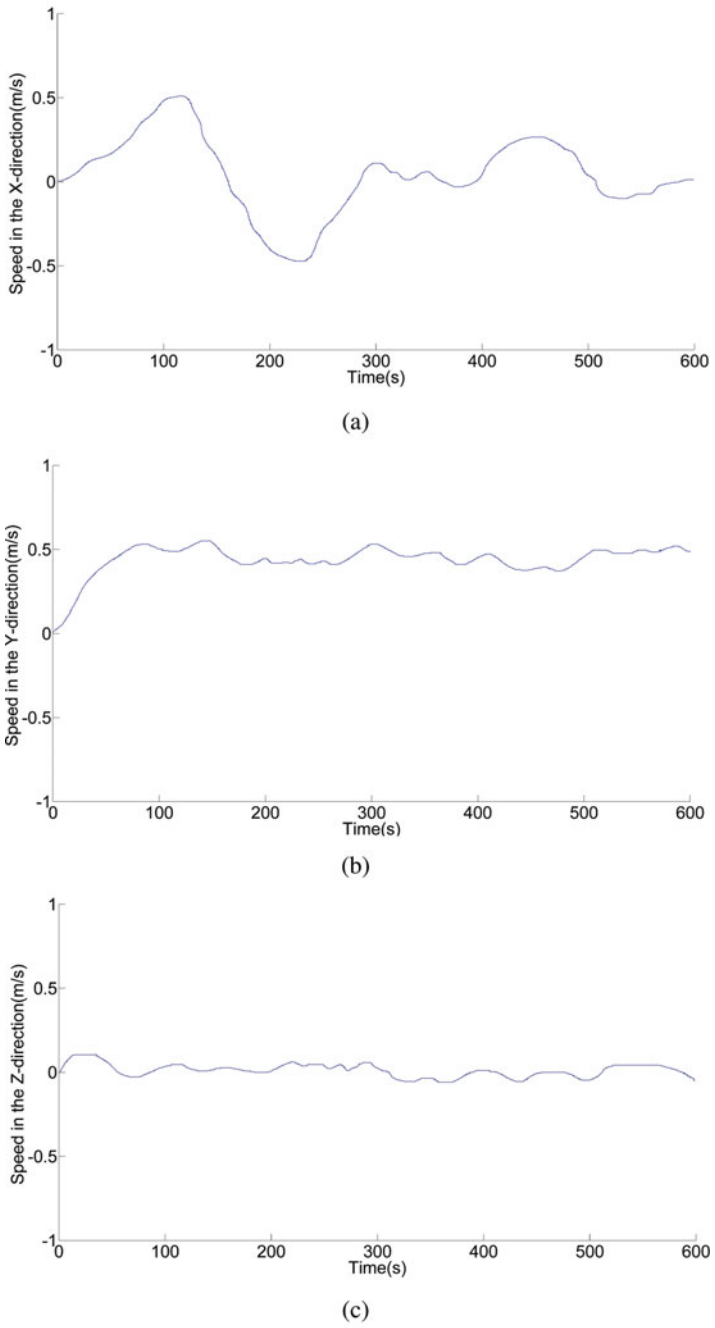


Fig. 15. Speed data in x, y and z directions.

Let $d_{\text{safe}} > 0$ be a given desired safety margin between the robot and obstacles. Introduce the constants F_i as

$$F_i := \frac{V_i + V_{\max}}{R_i + d_{\text{safe}}} \left(1 + \frac{R_i}{(R_i + d_{\text{safe}})\sqrt{1 - \frac{R_i^2}{(R_i + d_{\text{safe}})^2}}} \right) \tag{16}$$

for all $i = 1, 2, \ldots, k$.

Assumption 6.2. The inequality $F_i < A_{\max}$ holds for $i = 1, 2, \ldots, k$, where F_i are defined by (16).

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

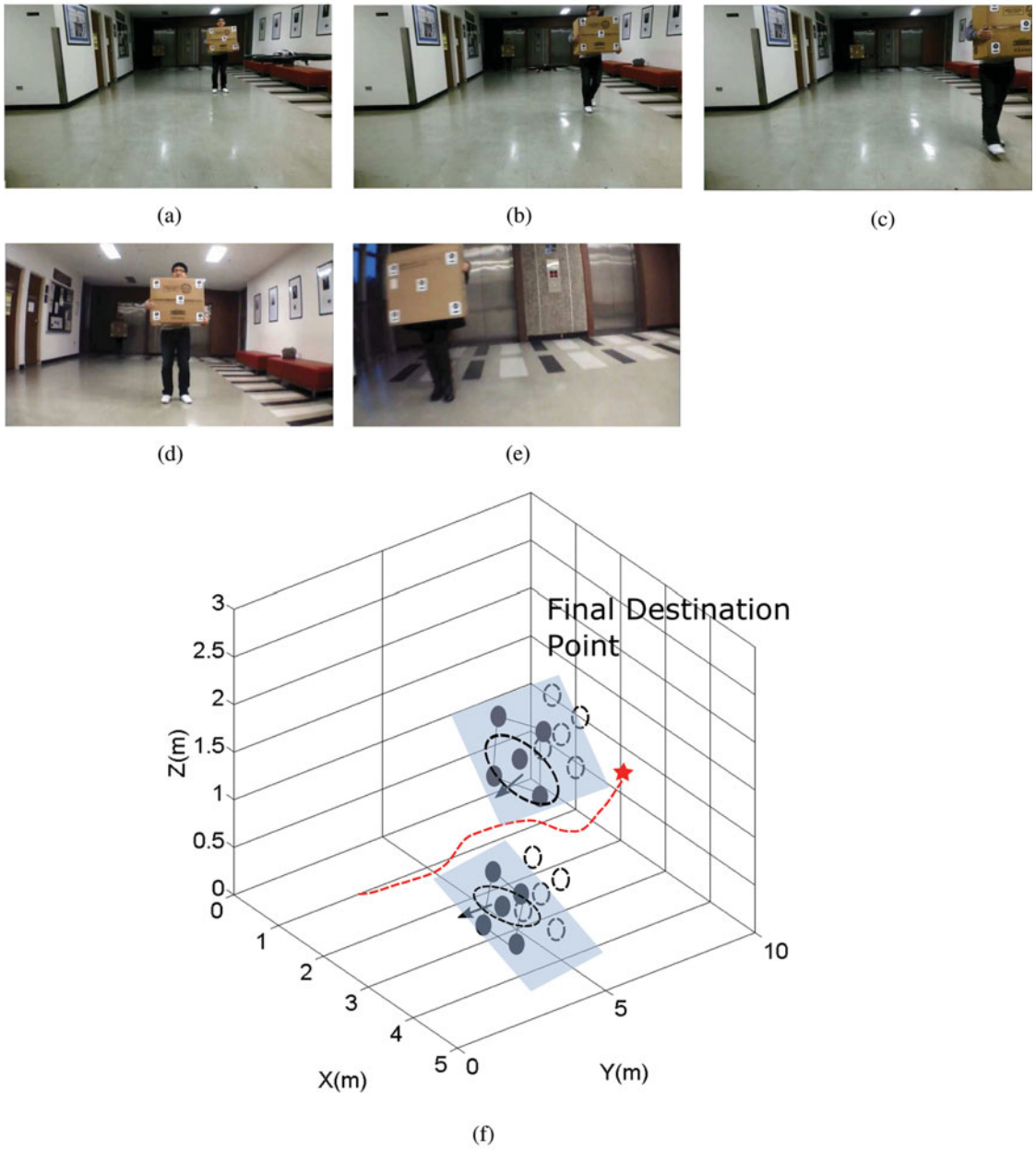


Fig. 16. AR Drone avoids multiple moving obstacles, case 1.

Assumption 6.3. The parameter C in the switching rule **R1** satisfies:

$$C \geq \frac{\pi V_{\max}}{A_{\max} - F_i} + 1.1a_i \quad \forall i = 1, 2, \ldots, k.$$

Let $S_{ij}(t)$ denote the distance between the obstacles i and j at time t, and let $S_i(t)$ be the distance between the obstacle i and the final destination point $\mathbf{F}$ at time t:

$$S_{ij}(t) := \min_{r \in D_i(t), q \in D_j(t)} \|r - q\|, \quad S_i(t) := \min_{r \in D_i(t)} \|r - \mathbf{F}\|.$$

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

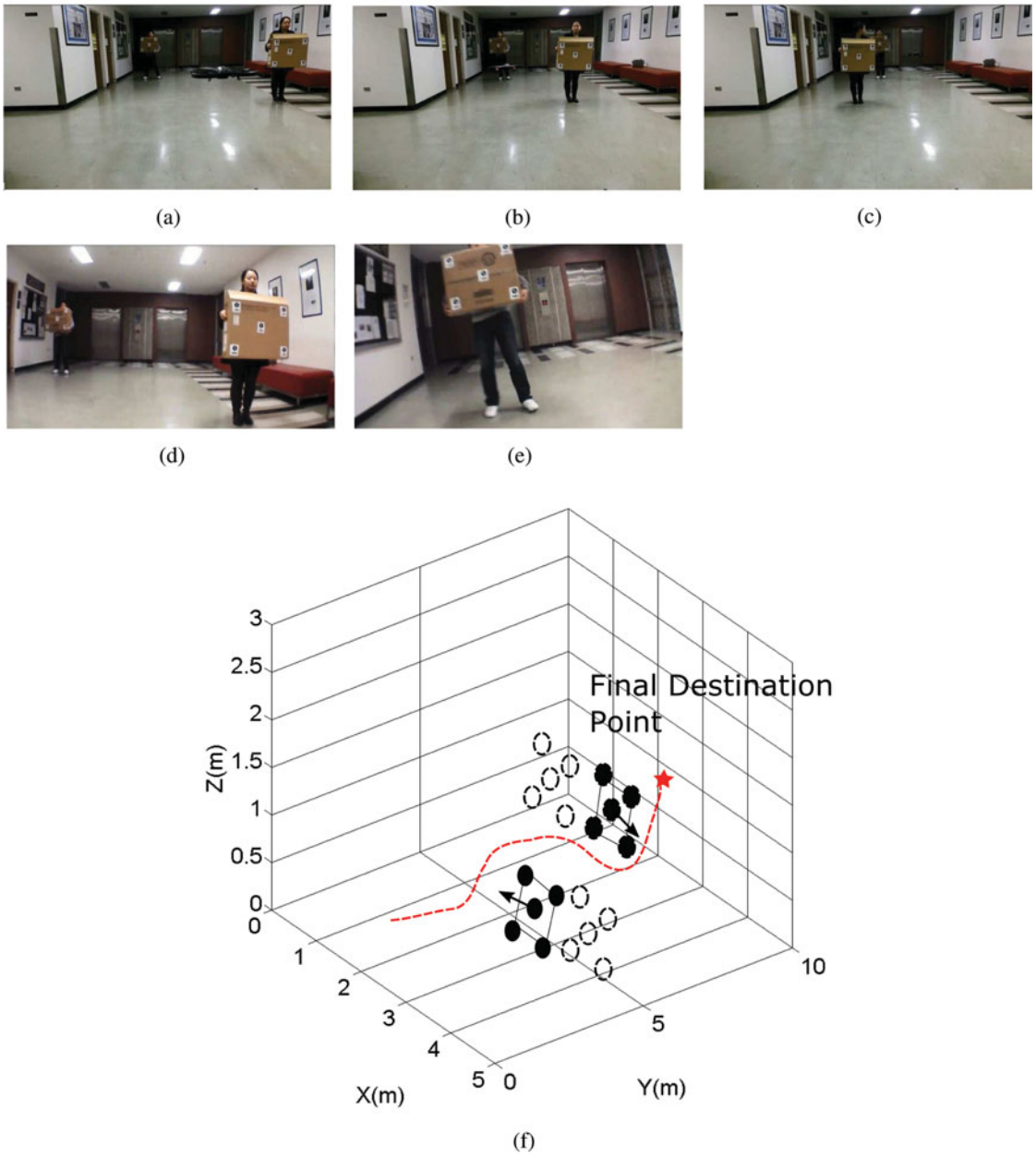


Fig. 17. AR Drone avoids multiple moving obstacles, case 2.

Assumption 6.4. We assume that

$$S_{ij}(t) \geq 2C + \frac{\pi V_{\max}}{A_{\max} - F_i};$$
$$S_i(t) > 1.1a_i$$

for all $i \neq j, t$.

Remark 6.2. The constant 1.1 in Assumptions 6.3, 6.4 is the constant from the control rule **R2** and can be replaced by any quantity $(1 + \epsilon)$, where $\epsilon > 0$, see Remark 3.4.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

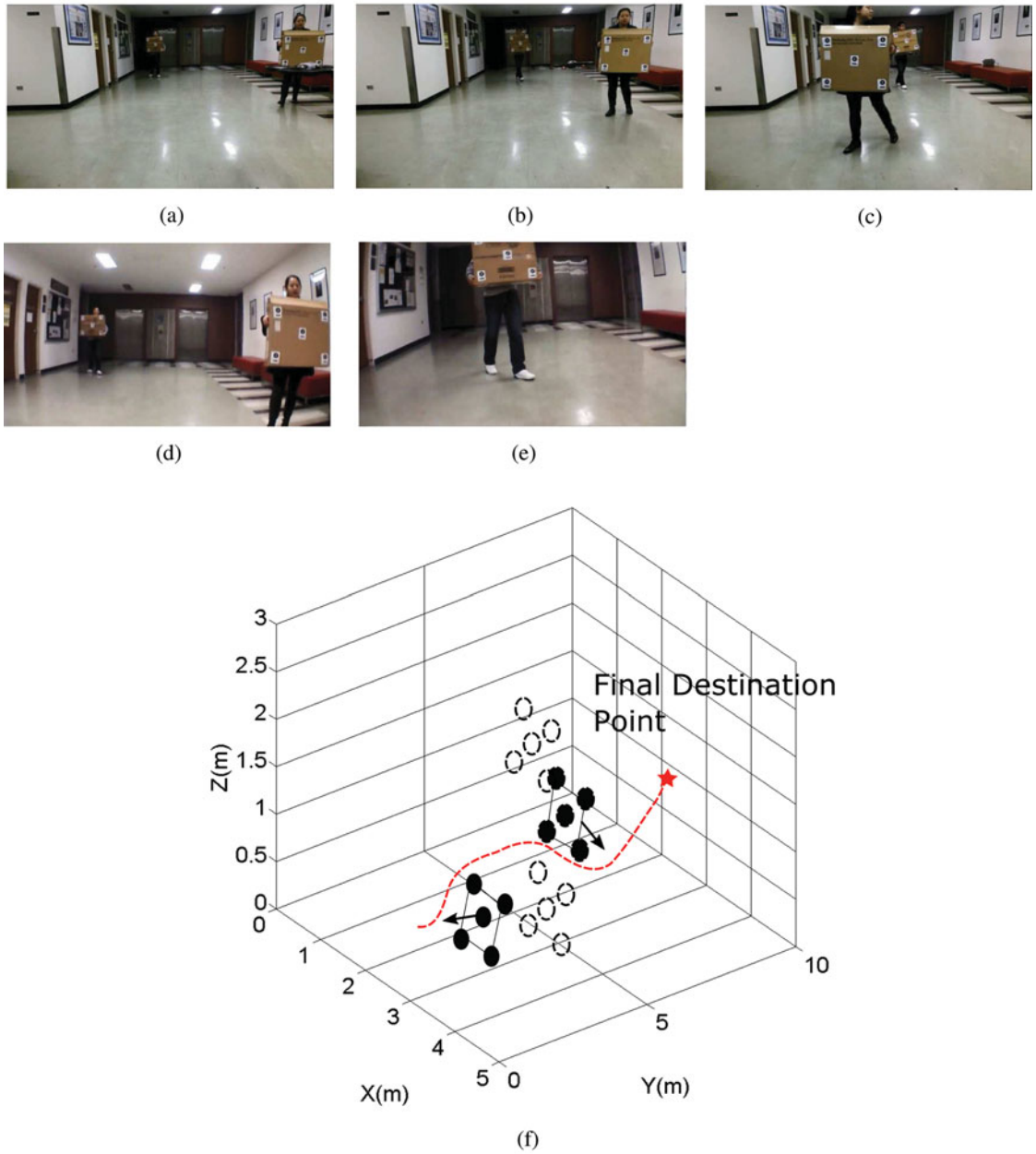


Fig. 18. AR Drone avoids multiple moving obstacles, case 3.

Assumption 6.5. The parameter α_0 in (8) and the safety margin d_{safe} satisfy:

$$\alpha_0 \geq \arccos\left(\frac{R_i}{R_i + d_{\text{safe}}}\right) \quad \forall i = 1, 2, \ldots, k.$$

Remark 6.3. It obviously follows from Assumption 6.5 and (15) that $a_i \geq d_{\text{safe}}$ for all i.

Assumption 6.6. We also assume that at initial time $t = 0$ the robot is moving towards the final destination point according to Eq. (14), and $d_i(0) > C$ for all i.

Theorem 6.1. Let $d_{safe} > 0$, $0 < \alpha_0 < \frac{\pi}{2}$ and $C > 0$ be given and suppose that Assumptions 6.1–6.6 hold. Then the control rule **R1, R2** provides a destination reaching strategy with collision avoidance.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

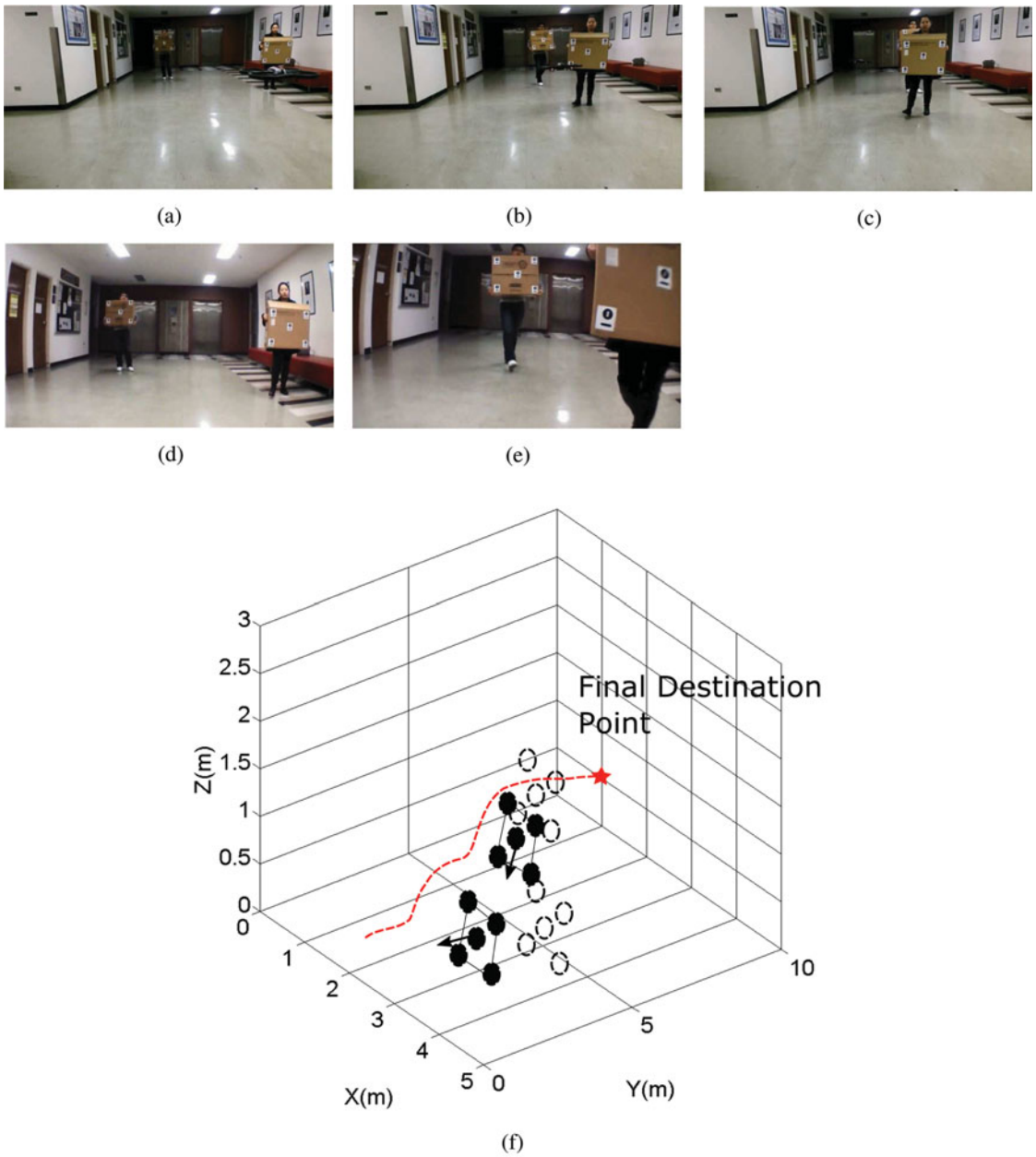


Fig. 19. AR Drone avoids multiple moving obstacles, case 4.

Remark 6.4. It should be pointed out that Assumptions 6.1–6.6 are quite technical assumptions which are necessary for a formal proof of the performance of the proposed reactive navigation algorithm. However, our computer simulations and experiments with a real flying robot presented in the previous two sections show that the proposed navigation algorithm often performs well even in situations when Assumptions 6.1–6.6 do not hold.

Proof of Theorem 6.1: First, we prove the following claim.

Claim 1: If the event **R1:** switching from (14) to (11) occurs at some time t_0, then there exists a time $t_* > t_0$ such that the event **R2:** switching from (11) to (14) occurs.

Indeed, the robot navigation strategy (11) steers the vector $v_i(t) + l_i^{(h)}(t)$ towards the vehicle velocity vector $v(t)$. It follows from a standard geometry that if $d_i(t) \geq d_{safe}$ then the absolute value of the angular velocity of the vector $v_i(t) + l_i^{(h)}(t)$ is less than F_i, where F_i is defined by (16). Therefore, Assumption 6.2 implies that the robot navigation strategy (11) guarantees that $v_i(t) + l_i^{(h)}(t) = v(t)$

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

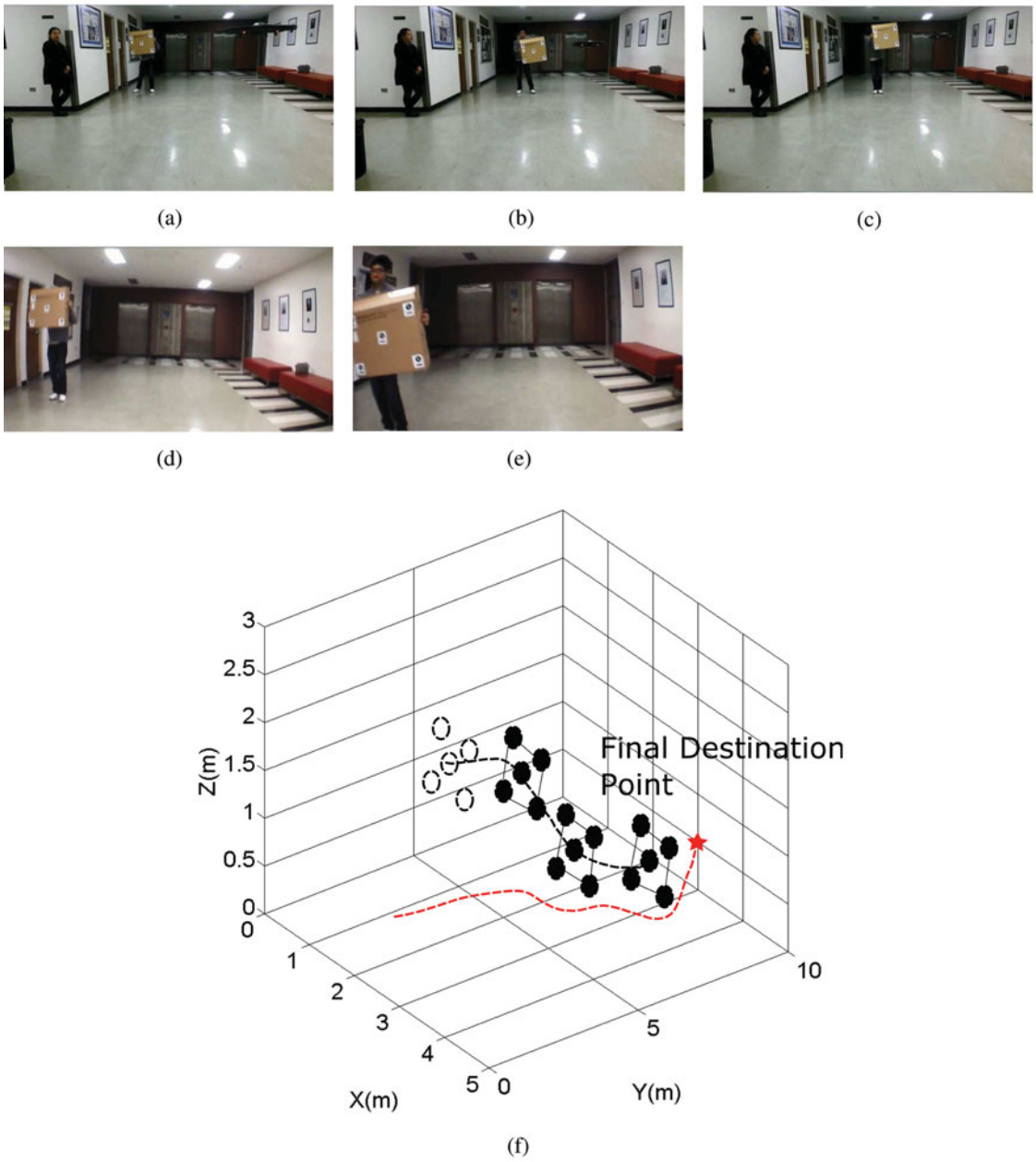


Fig. 20. AR Drone avoids obstacles moving with non-linear velocity.

for all $t > t_0 + \frac{\pi}{A_{\max} - F_i}$ until the event **R2:** switching from (11) to (14) occurs. Notice that Assumption 6.4 implies that during the time interval when the vehicle is governed by the control law (11), $d_j(t) > C$ for all $j \neq i$. Hence, when the unmanned vehicle is avoiding the obstacle i, switching to avoiding any other obstacle cannot occur. Furthermore, if $v_i(t) + l_i^{(h)}(t) = v_R(t)$, we can consider a system of coordinates moving with the velocity vector $v_i(t)$. In these coordinates, the obstacle i becomes steady, and the vehicle has the velocity vector $l_i^{(h)}(t)$. Furthermore, the motion with the velocity vector $l_i^{(h)}(t)$ keeps a constant avoiding angle α_0 between the instantaneous moving direction of the vehicle and one of the two boundaries of the vision cone from the robot of the covering sphere of the obstacle i see Fig. 3. For the motion with the velocity vector $l_i^{(h)}(t)$, we have that

$$\dot{d}_i(t) = -(V_{\max} - V_o)\cos(\gamma(t) + \alpha_0) \tag{17}$$

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

where $\gamma(t) + \alpha_0$ is the angle between the line connecting the robot with the center of the covering sphere of the obstacle i and the vehicle's velocity vector. It follows from (17) that the distance $d_i(t)$ is decreasing if $\gamma(t) + \alpha_0 < \frac{\pi}{2}$. Therefore, the distance $d_i(t)$ is decreasing when the unmanned vehicle is outside of the sphere of radius $\frac{R_i}{\cos\alpha_0}$ centered at the center of the obstacle's covering sphere, and $\dot{d}_i(t) = 0$ when the vehicle's coordinates belong to this sphere. Moreover, the vehicle is always in the plane defined by the direction towards the center of the covering sphere of the obstacle and the robot's motion direction. Hence, the vehicle's trajectory converging to the circle of radius $\frac{R_i}{\cos\alpha_0}$ in this plane centered at the center of the covering sphere. On this circle, $v(t)$ becomes orthogonal to the line connecting the robot with the center of the covering sphere of the obstacle i. It follows from Assumption 6.5 that the safety constraint $d_i(t) \geq d_{safe}$ will be satisfied. Moreover, it follows from (15) that the circle's radius $\frac{R_i}{\cos\alpha_0}$ equals $R_i + a_i$, hence, $d_i(t) \to R_i + a_i$. This implies that $d_i(t) \leq 1.1a_i$ for large enough t. Furthermore, it is obvious that if the vehicle's path is converging to a circle centered at the center of the covering sphere and the final destination point is outside of this circle (see Assumption 6.4, then there exists a time t_* such that the vehicle is oriented towards the final destination point. Therefore, switching **R2:** from (11) to (14) occurs at some t_*. We have proved that every time the robot moves according to the obstacle avoiding strategy (11), it will switch to moving towards the final destination point (14). Furthermore, Assumption 6.1 guarantees that all the obstacles move in different directions. Therefore, there exists a time $t_{**} > 0$ such that at time t_{**} the distance between any point on the straight line interval between the final destination point and any point on the circle of radius $R_i + 1.1a_i$ lying in the robot's current motion plane and centered at the center of the covering sphere of the obstacle i is greater than C. Hence, after the time t_{**}, switching **R1:** to the collision avoiding navigation strategy (11) cannot occur. This and **Claim 1** immediately imply that the final destination point will be reached. This completes the proof of Theorem 6.1.

Remark 6.5. In our approach, we have a simple method to make the algorithm robust to any measurement noise. If we make the covering sphere of an obstacle bigger than the obstacle's minimum covering sphere, then with a bounded measurement noise with the magnitude less than a certain positive constant, the obstacle will still be inside this bigger sphere. Since our algorithm guarantees that the robot is always outside the covering sphere, it guarantees that the robot always avoids collision with the obstacle.

7. Conclusion and Future Research

A method for collision free three-dimensional navigation of a non-holonomic unmanned vehicle among moving obstacles is proposed and rigorously justified. The efficiency of the proposed algorithm is illustrated by computer simulations and experiments with a real flying robot. The proposed method can also be implemented with unmanned underwater vehicles. An important direction of future research will be an extension of the proposed navigation law to more complex robot models incorporating dynamics and model uncertainty; see e.g., refs. [16, 17] Another interesting direction of future research will be the use of covering ellipsoids instead of covering spheres. This would allow to make the navigation algorithm less conservative and more efficient.

Acknowledgment

This work was supported by the Australian Research Council.

References

1. A. James, J. Winnefeld and K. Frank, Unmanned Systems Integrated Roadmap. (Department of Defense, USA, 2013).
2. A. S. Matveev, C. Wang and A. V. Savkin, "Real-time navigation of mobile robots in problems of border patrolling and avoiding collisions with moving and deforming obstacles," *Robot. Auton. Syst.* **60**(6), 769–788 (2012).
3. A. V. Savkin and C. Wang, "A simple biologically inspired algorithm for collision-free navigation of a unicycle-like robot in dynamic environments with moving obstacles," *Robotica*, vol. 31, pp. 993–1001, 2013.

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press

4. M. C. Hoy, A. S. Matveev and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica* **33**(3), 463–497 (2015).
5. A. S. Matveev, A. V. Savkin, M. C. Hoy and C. Wang, *Safe Robot Navigation among Moving and Steady Obstacles* (Elsevier, 2015).
6. A. V. Savkin and C. Wang, "Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation," *Robot. Auton. Syst.* **62**(10), 1568–1580 (2014).
7. J. D. Hernandez, E. Vidal, G. Vallicrosa, E. Galceran and M. Carreras, "Online Path Planning for Autonomous Underwater Vehicles in Unknown Environments," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, USA (May 2015) pp. 1152–1157.
8. S. Subramanian, T. George and A. Thondiyath, "Real-time obstacle avoidance for an underactuated flat-fish type autonomous underwater vehicle in 3D space," *Int. J. Robot. Autom.* **29**(4), 424–431 (2014).
9. F. Munoz, E. Espinoza, I. Gonzalez, L. Garcia Carrillo, S. Salazar and R. Lozano, "A UAS Obstacle Avoidance Strategy Based on Spiral Trajectory Tracking," *Proceedings of the International Conference on Unmanned Aircraft Systems*, Arlington, USA (Jun. 2015) pp. 593–600.
10. T. J. Stastny, G. A. Garcia and S. S. Keshmiri, "Collision and obstacle avoidance in unmanned aerial systems using morphing potential field navigation and nonlinear model predictive control," *J. Dyn. Syst. Meas. Control* **137**(1), 014503 (2015).
11. R. Brockers, Y. Kuwata, S. Weiss and L. Matthies, "Micro Air Vehicle Autonomous Obstacle Avoidance From Stereo-Vision," *Proceedings of SPIE Vol. 9084, Unmanned Systems Technology XVI* (Jun. 2014) pp. 90 840O–1–90 840O–12.
12. S. Hrabar, "Reactive Obstacle Avoidance for Rotorcraft UAVs," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, USA (Sep. 2011) pp. 4967–4974.
13. S. Hrabar, "An evaluation of stereo and laser-based range sensing for rotorcraft unmanned aerial vehicle obstacle avoidance," *J. Field Robot.* **29**(2), 215–239 (2012).
14. X. Yang, L. M. Alvarez and T. Bruggemann, "A 3D collision avoidance strategy for UAVs in a non-cooperative environment," *J. Intell. Robot. Syst.* **70**(1–4), 315–327 (2013).
15. D. H. Shim and S. Sastry, "An Evasive Maneuvering Algorithm for UAVs in See-and-Avoid Situations," *American Control Conference*, IEEE (2007) pp. 3886–3891.
16. Z. Peng, D. Wang, Y. Shi, H. Wang and W. Wang, "Containment control of networked autonomous underwater vehicles with model uncertainty and ocean disturbances guided by multiple leaders," *Inform. Sci.* **316**, 163–179 (2015).
17. Z. Peng, D. Wang, Z. Chen, X. Hu and W. Lan, "Adaptive dynamic surface control for formations of autonomous surface vehicles with uncertain dynamics," *IEEE Trans. Control Syst. Technol.* **21**(2), 513–520 (2013).
18. A. S. Matveev, M. C. Hoy and A. V. Savkin, "3D environmental extremum seeking navigation of a nonholonomic mobile robot," *Automatica* **50**(7), 1802–1815 (2014).
19. V. I. Utkin, *Sliding Modes in Control Optimization*. (Springer-Verlag, Berlin, 1992).
20. Offical Website for Parrot ARDrone. [Online]. Available: `http://ardrone2.parrot.com/(Jun., 2015)`
21. ROS Driver for Parrot AR-Drone 1.0 and 2.0 Quadrocopters. [Online]. Available: `https://github.com/AutonomyLab/ardrone_autonomy(Jun., 2015)`
22. ARDrone Aag Follow. [Online]. Available: `https://github.com/krishnavamshi/ardrone_tagfollow(Jun., 2015)`
23. Control for the Parrot AR Drone. [Online]. Available: `https://github.com/gigasquid/clj-drone(Jun., 2015)`

https://doi.org/10.1017/S026357471700039X Published online by Cambridge University Press