Obstacle Recognition and Avoidance for UAVs Under Resource-Constrained Environments

YIMING MAO, MING CHEN, XIANGLIN WEI, AND BING CHEN

1Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
2The 63rd Research Institute, National University of Defense Technology, Nanjing 210007, China

This work was supported in part by the National Natural Science Foundation of China under Grant 61772271 and Grant 61379149.

ABSTRACT Existing resource-intensive obstacle avoidance techniques are hard to be applied on the small-size Unmanned Aerial Vehicles (UAVs) that have limited sensing and computation capacity. Therefore, it is necessary to develop an obstacle recognition and avoidance scheme which works under resource-constrained environments. To this backdrop, this paper first presents an obstacle recognition model based on monocular vision feature points. Afterwards, an obstacle recognition algorithm is put forward, whose computational complexity is low. Then, an obstacle avoidance method is proposed to regulate the obstacle-avoidance path of a UAV until it arrives its destination. To evaluate the effectiveness of the presented algorithms, we design and implement a simulation platform on Objective Modular Network Testbed in C++ (OMNeT++), and conduct a series of experiments. Experimental results show that the proposed model and algorithms can effectively guide a micro UAV to its destination using only an embedded processor and 0.5 kg extra load. Even in poor communication conditions, the UAV can independently avoid obstacles and reach the destination only by acquiring the destination coordinates from the ground station.

INDEX TERMS UAV, obstacle avoidance, monocular visual feature point, obstacle recognition.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have been widely used in many mission-critical applications, such as reconnaissance [1], search and rescue [2], land and resources monitoring [3], bridge crack detection [4], and computation enhancement in Internet of Things scenarios [5], due to their inherent advantages including flexibility, low-cost, and mobility. To effectively conduct tasks in complex or even hostile environments, UAVs are required to be able to recognize and avoid obstacles in an autonomous manner. Therefore, obstacle avoidance flight control and path planning has become one of the key challenges for the practitioners [6].

Advances in battery, computation, and mechanical technologies have enabled the development miniaturized flying vehicles [7], which could fly flexibly in complex scenarios. In order to avoid collision between itself and various objects (known as obstacles), a UAV needs to have the ability of environmental awareness, target tracking, path planning, and flight control. Environmental awareness requires the UAV to recognize the obstacles on its moving direction using its equipped sensors, and obtain the relative distance and angular position information between itself and the obstacles. Target tracking updates the dynamic information of obstacles in real time and provides reference for planning obstacle-avoidance path by analyzing and processing data retrieved by various sensors. Path planning can dynamically calculates an obstacle-avoidance path based on the obstacle information and the UAV’s flight status. Flight control is used to control and adjust a UAV’s flight speed and direction based on its planned obstacle avoidance path. Moreover, all these functions should be implemented under the limited load-carrying capability and power supply severely of the UAVs.

Different types of sensors, including optical, infrared, ultrasonic, millimeter wave, laser, and etc., can be used to detect obstacles. However, many sensors are not suitable for UAV obstacle avoidance because of weight, volume or energy limitation. Therefore, the UAV obstacle avoidance methods based on small sensors, especially cameras, have attracted much attention [8]. Existing proposals could be classified into three categories. In the methods of the first category, visual obstacle avoidance is conducted based on identifying...
The main contributions of this paper are that satisfy the flight curvature constraint and the tangential connecting two two-dimensional planes under conditions.

In this paper, an obstacle recognition algorithm and an obstacle avoidance method for UAVs are put forward under resource-constrained environments. Firstly, an obstacle recognition model based on monocular visual feature points is proposed, and an obstacle recognition algorithm (DOMD) based on the model is presented. The algorithm only uses the information collected from one monocular camera and one laser range finder, and has no requirements on communication capability. The required sensors are also small and light, which meets the requirements of UAV. By comparing feature points of the images captured in different time intervals and adding one-dimensional distance information, our algorithm can revert the actual size of the obstacle in the direction of the UAV. Compared with traditional morphological methods, multi-image feature analysis in our proposal has better stability. In comparison with vSLAM-based methods, our algorithm is built on much simpler sensors, and the time-consuming of obstacle avoidance is saved because of the neglect of mapping. Our algorithm’s primary advantage over those machine learning methods is that we do not need any prior data and no training process is needed. Secondly, a method called PLDOMD is put forward to control the UAV to avoid obstacles. The aim of PLDOMD is not to solve the problems in route planning, but to complete the navigation of UAV obstacle avoidance after the obstacle is recognized process completed by DOMD algorithm, so as to realize real-time and effective obstacle avoidance. PLDOMD relies on DOMD algorithm and Dubins path to plan the obstacle avoidance path. The Dubins path is the shortest path connecting two two-dimensional planes under conditions that satisfy the flight curvature constraint and the tangential direction of the specified start and end [12]. Combining the actual size of the obstacles obtained by DOMD algorithm and the constraints of the UAV’s own flight conditions, our method can quickly calculate the shortest obstacle avoidance path of the UAV to bypass the obstacle and finally reach the destination. The main contributions of this paper are as follows:

1) An obstacle recognition method that is easy to be conducted under resource-constrained environments by UAVs is proposed. Compared with existing methods, it has unique advantages, including low cost and light weight, and can better adapt to resource-constrained UAVs. Moreover, its computational complexity is low and can be easily supported by the UAV’s on-board processor.

2) After recognizing the obstacle, a method called PLDOMD that considers the UAV’s own flight constraints is developed to establish a new route to quickly bypass the obstacle in real-time. With the help of PLDOMD method, a UAV with limited resources could avoid collision with obstacles in real time.

3) Based on OMNeT++ [13], which is an open source multi-protocol network simulation platform, a simulation platform is designed, and a prototype is implemented to verify the feasibility of the proposed model and algorithms.

The remain parts of this paper are organized as follows. Section II outlines related work. Section III establishes the model for identifying fault points based on monocular visual feature points. Section IV designs the UAV obstacle avoidance algorithm based on the model and the proposed DOMD. Section V describes the implementation of our experimental system and shows experimental results. Finally, Section VI briefly summarizes this paper.

II. RELATED WORK

Helping UAVs understand their environments and work intelligently is one of the hottest topics in UAV area. An important building block of current research is developing obstacle-avoidance algorithms for UAVs. Kendoul et al. summarized a series of drone navigation, positioning, and control algorithms [14]. Next, we will introduce the traditional sensor methods, and the common vSLAM methods especially based on monocular vision.

Currently, practitioners adopted different sensors to detect and locate potential obstacles, including sound waves, infrared, radar, laser, and photoelectric. The laser-based method has been proven to be effective in reliably obstacles detection in different environments. But the lidar is expensive and heavy to carry [15]–[17]. In contrast to the lidar, the lightweight laser rangefinder can only provide very simple distance information for one-way objects.

Compared with traditional laser range finder, high-precision inertial measurement unit and GPS sensor, the vision sensor had a large field of view and rich image information. The cost performance of vision sensor was relatively high, and at the same time, the identification features can be tracked, especially the monocular camera. As the most important sensing device was widely assembled on mobile robots, the research of monocular vision SLAM technology had gradually become an important research direction [18].

In the visual SLAM technology, the widely applied depth detection methods include motion parallax, monocular cues, and stereo vision in the biological vision system. The motion parallax is used in traditional optical flow methods. For example, Beyeler et al. developed an ultra-light indoor vehicle that used a small one-dimensional camera to avoid collisions with obstacles and away from obstacles [19]. Oh and Green
developed a barrier avoidance scheme based on autonomous optical flow with similar design concepts [20], [21]. Merrel et al. used larger fixed-wing aircraft to quickly avoid obstacles and travelled through the canyon using radar and optical flow tracking [22]. A fundamental limitation of the optical flow method is that the optical flow from one frame to another is proportional to the forward angle. This makes these methods very effective for applications to detect and fly along the wall, but difficult to use for frontal obstacles. Hrabar et al. also conducted a large number of experiments using optical flow and stereoscopic differences [23]. Since there should be no optical flow in front of the machine, stereoscopic differences were used in their work to detect obstacles in front. Other researchers had also used the optical flow method to study the obstacle avoidance of drones [24], [25].

Monocular cues are also used to avoid collisions. For example, Çelik et al. conducted a collision avoidance test at the center of the corridor using a perspective cues method [18]. Lee et al. used the MOPS algorithm to detect the outline of obstacle and at the same time used Scale-invariant feature transform (SIFT) algorithm to calculate the feature points internal outline of the obstacle to get the three-dimensional information of the obstacle [26]. In dealing with poor illumination of indoor environment, Bai et al. proposed a method by improving Parallel Tracking And Mapping (PTAM) algorithm, reduce the dependency of the algorithm on number of feature points and illumination conditions [27]. These methods were primarily designed to work well in environments such as corridors, but in the natural environment there were often no perspective cues that the method requires.

Recently, with the popularity of machine learning, scholars began to use it to solve the problem of UAV obstacle avoidance. Zhang et al. collected data under full state observation based on Model predictive control (MPC) method to train a neural network to guide UAV to avoid obstacle [28]. This method collected the prior MPC obstacle avoidance data and trained the UAV obstacle avoidance neural network. When the UAV encountered similar problems, it can only rely on the neural network to get the obstacle avoidance control policy. Kelchtermans et al. successfully trained a Long Short-Term Memory (LSTM) network to control the UAV to fly in the room with obstacles [29]. They used a behavior arbitration algorithm in the training data collection phase instead of manual. This saved time during experimentation and ensures reproducibility of the results. Shah et al. used Deep Neural Network (DNN) to find the biggest box through the image captured by monocular camera [30]. The center of the box was selected as next waypoint of UAV. They calibrated 12000 sets of images as training data and tested their method in different real environment. Machine learning algorithm has high computational complexity, high resource overhead and dependence on a large number of prior data, which makes it difficult to deal with dynamic and unknown scenarios.

In this paper, we rely on simple sensors to complete complex obstacle avoidance tasks. Existing sensor-based obstacle-avoidance schemes are usually expensive and heavy, and are not suitable for UAVs with high mobility. Current visual obstacle avoidance schemes require extremely high computational performance and have certain requirements for the scenes. Our methods can effectively work under resource-constrained environments without any prior data.

III. PROBLEM STATEMENT AND MODELING

A. PROBLEM STATEMENT

The scenario considered in this paper is shown in Fig. 1, and could be described as follows:

1) A UAV \( u \) equipped with a monocular camera and a laser range finder, aims to move from \( S \) (start position) to \( D \) (target position). An initial path, i.e. \( l \) in Fig. 1, shows the predefined path for \( u \).
2) An obstacle exists on \( l \). During the flight, \( u \) needs to identify the existence of an obstacle.
3) The equipped monocular camera and laser range finder of \( u \) are in the same direction as the head of \( u \).
4) If the object in front is close enough, \( u \) can obtain the distance information of this object by the laser range finder and the three-dimensional information by monocular camera.
5) If \( u \) observes an obstacle on its way, it needs to adjust its flight trajectory, e.g. \( l' \) shown in Fig. 1, to avoid a collision with the obstacle.

B. MODELING OF THE PROBLEM

Based on the above statement, the method should be able to solve the following two major problems. First, the UAV needs to judge whether there is an obstacle on the planned path based on the images and the distance information acquired by its sensors. Second, the UAV has to plan an obstacle-avoidance detour to bypass the obstacle as soon as possible.

In addition to these two problems, our method also needs to control the entire flight process of the UAV to finally reach the ending point. Fig. 2 shows a finite state machine depicting the obstacle avoidance process. There are three states in this figure: Accessible Flight, Obstacle Avoidance Flight, and Termination Flight. When \( u \) does not observe an obstacle, \( u \) is
in a state of “Accessible Flight”; if an obstacle is observed, \( u \) is in a state of “Obstacle Avoidance Flight”; once \( u \) reaches the ending point, it will be in a state of “Termination Flight”. After task release, \( u \) sets the ending point, plans a route to the destination, and enters the “Accessible Flight” state. When \( u \) is in the “Accessible Flight” state, if it finds itself reaches the destination, it will stop and enter “Termination Flight” state; if it finds an obstacle, it obtains obstacle information to calculate a detour and enters the “Obstacle Avoidance Flight” state. When \( u \) is in “Obstacle Avoidance Flight” state, if it reaches the obstacle avoidance point, it reschedules its flight path to the destination and reenters the “Accessible Flight” state.

IV. DOMD AND PLDOMD ALGORITHM

A. OBSTACLE RECOGNITION

After the UAV \( u \) captures an image in front by its monocular camera, Oriented FAST and Rotated BRIEF (ORB) method is utilized to extract feature points in the image [31]. Compared with SIFT and Speeded Up Robust Features (SURF) methods, ORB method is less robust to rotation and scale transformation, but it has the fastest calculation speed [32]. Moreover, when \( u \) is flying forward to obtain the obstacle’s image, there is usually no rotation and scale conversion. So ORB method can better satisfy the time constraint while meeting the accuracy requirements for obstacle avoidance. When \( u \) takes the second picture after \( \Delta t \), it needs to extract its feature points of the new image and match them with the previous ones. The approximate nearest neighbor search algorithm (FLANN) [33], which has better accuracy and matching speed, is adopted here to conduct feature points matching. Since the obstacle encountered by \( u \) first must be the nearest object to \( u \). The changes of its feature points can be captured. So the feature points of background or other further objects can be removed while those ones of the obstacle will be retained.

1) OBSTACLES RECOGNITION MODEL

By comparing the feature points of two pictures, the object with the biggest changing rate of feature points is treated as an obstacle; the size of the obstacle is then calculated using the quantitative relationship provided by the model we are going to show; at last, a detour which can bypass the obstacle is determined.

After the feature point extraction and matching are completed, the feature point set \( U \) matching two frames can be defined as [33]:

\[
U = \left\{ \left( K_{P_1}^{d(i)}, K_{P_2}^{d(i)} \right) \mid K_{P_1}^{d(i)} \in K_{P_1}, K_{P_2}^{d(i)} \in K_{P_2}, \text{id} = \text{matches} \left( K_{P_1}, K_{P_2} \right) \right\}
\]  

(1)

where \( K_{P_1} \) and \( K_{P_2} \) represent the set of feature points of current and the previous frames respectively, and \( id \) is the index set after the matching of feature points, and each element in the \( id \) set has two values \( x \) and \( y \). It represents that the \( id \) \( x \)-th feature point in \( K_{P_1} \) are the same feature point as the \( id \) \( y \)-th feature point in \( K_{P_2} \).

Fig. 3 shows the quantitative relationship that determines the movement of an object image based on a Pinhole Camera Model, which is a simple and widely used model that
describes the mathematical projection from 3D world coordinates to a 2D image plane [34]. Suppose the projection center of the camera is at the origin \( O \) of a Cartesian coordinate system, the camera faces the positive direction of the \( Z \) axis, and the image plane (or focal plane) is expressed as \( Z = f \) under the current coordinate system. As shown in Fig. 3, a point \( X = (Xc, Yc, Zc)^T \) (in meter) on an object in 3D space is projected to be a 2D point on the image plane, that is, the intersection point \( x = (xc, yc, f)^T \) of the line connecting the camera’s projection center and the 3D point with the image plane. When the object moves toward the camera (origin \( O \)), with the displacement distance \( \Delta Z \), the point comes to \( X’ = (Xc, Yc, Zc - \Delta Z)^T \) with a new intersection point \( x’ \) generated on the image plane. Through similar triangle relationships, we have:

\[
xc = f \frac{Xc}{Zc}, \quad yc = f \frac{Yc}{Zc}
\]  

(2)

Convert this intersection point to the image plane coordinate system, we have:

\[
x = fx \frac{Xc}{Zc} + cx, \quad y = fy \frac{Yc}{Zc} + cy
\]  

(3)

The units of \( f_x, f_y, cx, \) and \( cy \) are pixels, which together constitute the internal parameters of the camera. The internal parameters can usually be obtained by camera calibration. If the inner reference point \( (cx, cy) \) is determined as the centroid of the image plane, the distance \( d = (p, q) \) between the intersection point and the centroid is:

\[
p = fx \frac{Xc}{Zc}, \quad q = fy \frac{Yc}{Zc}
\]  

(4)

After homogenization, we have:

\[
\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}
\]  

(5)

Then, we know that when the obstacle is moving toward the camera (origin \( O \)) along the \( Z \)-axis, \( X \) and \( Y \) are unchanged, so \( \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} \) is a constant value, and it can be known that as \( Z \) is reduced, \( p, q \) are scaled up. Therefore, it is possible to obtain the coordinates of the Euclidean space of the new projection point \( x’ \) of the point \( X’ \) on the image plane after the object is displaced:

\[
x’ = \left( xc \frac{Zc}{Zc - \Delta Z}, yc \frac{Zc}{Zc - \Delta Z}, f \right)^T
\]  

(6)

Its coordinates on the image plane are:

\[
x’ = \frac{Zc}{Zc - \Delta Z} x
\]  

(7)

It can be reflected on the image that as the camera and the object are constantly approaching, the image of the object is continuously enlarged with the centroid \( (cx, cy) \).

In order to separate obstacles closer to the camera and background objects farther from the camera, we need to analyze how the obstacles and background objects differ in the process of approaching the camera.

Suppose there are two relatively stationary points \( A = (Xa, Ya, Za)^T \) and \( B = (Xb, Yb, Zb)^T \), \( B \) is farther from the origin than \( A \). According to (5), the distances from points \( A \) and \( B \) to the centroid \((c_x, c_y)\) are:

\[
\begin{align*}
\tilde{d}_A &= (f_x \frac{Xa}{Za}, f_y \frac{Ya}{Za}) \\
\tilde{d}_B &= (f_x \frac{Xb}{Za}, f_y \frac{Yb}{Za})
\end{align*}
\]  

(8)

When the camera moves \( Za/2 \) to these two points, it can be regarded as that the two points moves towards the original point along the \( Z \)-axis by \( Za/2 \), and the resulting point after the moving is \( A' = (Xa, Ya, Za/2)^T \) and \( B' = (Xb, Yb, 7Za/2)^T \), the distances from points \( A' \) and \( B' \) to the centroid \((c_x, c_y)\) are:

\[
\begin{align*}
\tilde{d}_A’ &= (f_x \frac{Xa}{Za/2}, f_y \frac{Ya}{Za/2}) = 2\tilde{d}_A \\
\tilde{d}_B’ &= (f_x \frac{Xb}{7Za/2}, f_y \frac{Yb}{7Za/2}) = \frac{8}{7} \tilde{d}_B
\end{align*}
\]  

(9)

It can be seen that although the points \( A \) and \( B \) are close to the origin and their projections on the image plane leave away from \((c_x, c_y)\), the projection of the point closer to the origin leaves faster away from \((c_x, c_y)\). Based on this feature, we can filter the matched feature points set \( U \) with the following characteristics: the point (or the point sequence) with the highest changing rate or the obstacle closest to the camera (or the UAV), because the point with the larger changing rate of the distance from \((c_x, c_y)\) is the point closer to the camera. So, when the feature point set \( U \) is calculated, by deleting the point having a small changing rate, a set of points meeting the obstacle feature can be obtained.

We define the changing rate \( \alpha \) as the ratio of the distance of the feature points in the current frame from \((c_x, c_y)\) to the distance in the previous frame from \((c_x, c_y)\):

\[
\alpha = \sqrt{\frac{(Kp^{id(i)}_1.x - c_x)^2 + (Kp^{id(i)}_1.y - c_y)^2}{(Kp^{id(i)}_2.x - c_x)^2 + (Kp^{id(i)}_2.y - c_y)^2}}
\]  

where \( id = \text{matches} (KP_1, KP_2) \)

(10)

Therefore, the set of feature points of the obstacle on the current frame is:

\[
U_{\text{obstacle}} = \begin{cases} Kp^{id(i)}_1 | Kp^{id(i)}_1 \in KP_1, \\ 2 - \beta \} \end{cases}
\]  

(11)

where \( k \) is a fixed value and depends on the UAV’s speed \( v \), the dangerous distance \( d \) between the obstacle and a UAV, and the shooting interval \( t \) between two frames of images:

\[
k = \frac{d + vt}{d}
\]  

(12)

Selecting the feature points with \( \alpha > \beta k \) will filter the far background information and have some tolerance for the possible mistake of feature point position. And with
$O(2 - \beta)k > \alpha$ can filter out some mismatched points generated by the matching algorithm. Because the UAV flies forward at a constant speed, the changing rate of the obstacle is not too large, and the value of the constant $\beta$ depends on the actual filtering effect, which we will explore in the experimental part. After obtaining the obstacle feature point set $U_{obstacle}$ by inversely taking (3) into (13), the pixel coordinates of the image can be converted into the Euclidean space coordinates, thereby obtaining the position of the obstacle in the real space and its size, which means its upper, lower, left, and right borders.

\[
Xc = \frac{Zc}{f_x} (x - c_x), \quad Yc = \frac{Zc}{f_y} (y - c_y)
\]  
(13)

Based on (13), it is possible to calculate the shortest moving distance of a UAV to avoid this obstacle.

2) DOMD ALGORITHM

Determining Obstacle based on Monocular vision feature points and Distance (DOMD) algorithm is illustrated in Algorithm 1.

In Algorithm 1, Step 1 tests whether laser range finder detects an object in front or not. If an object is detected, this algorithm will be executed. Steps 2-3 extract and match feature points of two frames. One is current frame token when calling this algorithm, and the other is the previous frame token several seconds ago. Step 4 calculates the fixed distance $k$ by using the UAV’s speed $v$, the dangerous distance $d$ between the obstacle and a UAV, and the shooting interval $t$ between two frames of images. Steps 5-10 calculate the changing rate of all matching feature points, and steps 7-9 put the point with the appropriate changing rate into the set of obstacle feature points. Step 11 calculates the boundary distance of the obstacle feature point set. Step 12 converts pixel boundary distance to Euclidean space vector. Step 13 chooses the shortest boundary distance vector to return.

The computational complexity of the DOMD algorithm is mainly divided into two parts. The first part is the ORB feature point extraction algorithm, the second part is the FLANN matching and the selection of feature points that meet the requirements of change rate. The computational complexity is about:

\[
O(ORB) + O(n \log(n) + \log(n) + m)
\]  
(14)

For the first part, the computational complexity $O(ORB)$ is positively correlated with the resolution of the image [31]. The resolution of the image in our algorithm is the size of the frame obtained by the monocamera. With the improvement of image resolution, the runtime of the algorithm increases. For the second part, Where $O(n \log(n))$ and $O(\log(n))$ are the computational complexity of building tree and searching for FLANN match [35], $n$ is the number of feature points extracted by ORB algorithm, $O(m)$ is the computational complexity of screening feature points and searching boundary points, and $m$ is the number of remaining feature points after FLANN match.

Algorithm 1 DOMD

| Input: |
| Current frame, $frame1$; |
| Previous frame, $frame2$; |
| The UAV flight speed($m/s$), $v$; |
| The dangerous distance($m$), $d$; |
| Image shooting time interval(s), $t$; |
| Camera internal parameter, $(fx, fy, cs, cy)$; |

| Output: |
| The distance measured by the current laser ranging radar($cm$), $Z$; |

1: if $Z > d$ then
2: return NONE
3: end if
4: set $KP_1 = ORB(frame1); KP_2 = ORB(frame2)$
5: set $id = FlannMatch(KP_1, KP_2)$
6: set $k = (d + vt)/d$
7: for each $i \in [1, size(id)]$ do
8: set $\alpha = \frac{(K_{p_{1i}}^{id}x - c_x)^2 + (K_{p_{1i}}^{id}y - c_y)^2}{(K_{p_{2i}}^{id}x - c_x)^2 + (K_{p_{2i}}^{id}y - c_y)^2}$
9: if $1.2k > \alpha > 0.8k$ then
10: $U.append(KP_2^{id})$
11: end if
12: end for
13: set $D_{left} = min(U.x) - c_x; D_{right} = max(U.x) - c_x; D_{upper} = max(U.y) - c_y$
14: set $\leftarrow right = (D_{right} \cdot Z/ff, 0, 0); \rightarrow upper = (0, D_{upper} \cdot Z/ff, 0)$
15: set $\overline{Ob} = min(\leftarrow left, \rightarrow right, \rightarrow upper)$
16: return $\overline{Ob}$

3) AN EXAMPLE USING DOMD ALGORITHM

For the scene shown in Fig. 4, the initial distance from an obstacle to $u$ is $Z = 6m$. 2 seconds later, $Z$ reduces to $3m$ (less than the safety distance $d$), and $u$ enters the obstacle avoidance warning zone. $u$ calls DOMD algorithm, obtains the ORB feature points of the two frames of images, and matches them. The left side of Fig. 4 is the first frame, and the right side is the second frame. The size of the obstacle which is closer to the camera lens is nearly doubled in the next frame, while the farther background objects are less changed. Two nodes connected by a green line are the matched points between the two frames obtained after the ORB feature point extraction and the FLANN matching algorithm. Although there are some mismatches, important feature points are mostly matched.
Steps 5-10 in Algorithm 1 test the changing rate of each feature point by the traversal method, delete the points whose changing rate are small. Step 11 calculates the boundary of the remaining points belonging to the obstacle. The white point in Fig. 5 is the feature point filtered by the algorithm, wherein the points with little change or no change and the wrong feature points are filtered out. The area covered by the remaining feature points constitutes the area where \( u \) needs to avoid obstacles. After clustering calculation, the white border is the boundary of the obstacle.

Step 12 determines the boundary of the obstacle and gives the relative position of \( u \) and the obstacle. As can be seen in Fig. 6, the algorithm can obtain that \( u \) is closer to the left boundary of the obstacle with the distance vector \((-1.475m, 0, 0)\), which means if \( u \) flies 1.475m to the left side, it is possible to avoid this obstacle.

### B. OBSTACLE AVOIDANCE ROUTE PLANNING

After determining the obstacle based on the DOMD algorithm, it is necessary for \( u \) to plan a suitable sub-route, which is the shortest route that allows it to safely bypass the obstacle. Based on the information provided by the DOMD algorithm, this paper designs an obstacle avoidance sub-route based on the Dubins path.

#### 1) DETERMINING SUB-ROUTES ACCORDING TO DOMD ALGORITHM

To let the UAV \( u \) bypass obstacles as quickly and safely as possible, we design a suitable Dubins path for \( u \). From DOMD algorithm, the shortest distance from the boundary of the obstacle can be calculated and represented as \( \delta \overrightarrow{b} = (x_{ob}, y_{ob}, z_{ob}) \). Let the current \( u \) flight direction be the positive direction \( \overrightarrow{D_U} = (0, 0, 1) \), the coordinate is \( U = (x_u, y_u, z_u) \), and the ending point of \( u \) flight is \( D = (x_d, y_d, z_d) \), the distance from the obstacle is \( z_d \), and the minimum turning radius of \( u \) when turning is \( r \).

Therefore, the coordinate of the obstacle boundary closest to the \( u \) is:

\[
\overrightarrow{Ob} = (x_u + x_{ob}, y_u + y_{ob}, z_u + z_{ob}) \tag{15}
\]

The obstacle avoidance route of \( u \) has a safe distance to \( Ob \). The safety distance is set to \( 2\sqrt{2} r \), so that \( u \) will not collide with obstacles because it is less than the minimum turning radius, and it will not increase the length of the obstacle avoiding sub-route. That is, the position of the obstacle avoidance point \( P \) is:

\[
P = \begin{cases} 
  (x_u + x_{ob} + 2r, y_u + y_{ob}, z_u + z_{ob})^T, & x_{ob} > 0 \\
  (x_u + x_{ob} - 2r, y_u + y_{ob}, z_u + z_{ob})^T, & x_{ob} < 0 \\
  (x_u + x_{ob}, y_u + y_{ob}, z_u + z_{ob} + 2r), & x_{ob} = 0 
\end{cases} \tag{16}
\]

If \( u \) arrives at the obstacle avoidance point \( P \), its direction is opposite to the destination, and there is no new obstacle during the period, then there is no need to adjust the route. \( u \) only need to continue to fly directly to reach the destination.
Algorithm 2 PLDOMD

Input:
- The starting point of the task, S;
- The ending point of the task, D;

Output:
- The state of the UAV, State;
1: set State = “Accessible Flight”
2: set Path = Initialize (S, D)
3: while $U \neq D$ do
4: if DOMD == NONE then
5: set $U, \overrightarrow{DU} = Path (U)$
6: else
7: set State = “Obstacle Avoidance Flight”
8: set $\overrightarrow{Ob} = DOMD$
9: calculate Ob, $P, \overrightarrow{DP}$
10: set ObstacleRoute = Dubins $(U, P, \overrightarrow{DU}, \overrightarrow{DP})$
11: while State == “Obstacle Avoidance Flight” do
12: set $U, \overrightarrow{DU} = ObstacleRoute (U)$
13: if $U == P$ then
14: set State = “Accessible Flight”
15: end if
16: end while
17: set Path = Line $(P, D)$
18: end if
19: end while
20: set State = “Termination Flight”
21: return State

At this point, the flight direction of $u$ should be:

$$\overrightarrow{DP} = D - P$$  \hspace{1cm} (17)

2) PLDOMD ALGORITHM

In order for $u$ to bypass obstacles as quickly and safely as possible, this section designs a suitable Dubins path for $u$. In general, consider the case where $u$ may encounter multiple obstacles in succession. Planning Lane based on DOMD (PLDOMD) algorithm is described in Algorithm 2.

This algorithm is executed by the UAV $u$ during the whole mission and follows the finite state machine in Fig. 2. Step 1 begins when the mission starts and the state of $u$ is “Accessible Flight”. Step 2 initializes $u$ flight route from the starting point of the task $S$ to the ending point of the task $D$. Step 3-19 loop until $u$ arrives $D$. Step 4 calls the DOMD algorithm. If the DOMD algorithm feeds back no obstacle, step 5 allows $u$ flying along the route, and updating its coordinates and direction. If the DOMD algorithm finds an obstacle, step 7 changes the state of $u$ into “Obstacle Avoidance Flight”, and step 8-10 use the return of DOMD to calculate the coordinate of the obstacle boundary closest to $u Ob$, the position of the obstacle avoidance point $P$, the flight direction of $u \overrightarrow{DP}$ and use them to calculate Dubins route for obstacle avoidance. Step 11-16 loop if $u$ is in “Obstacle Avoidance Flight”. Step 12 allows $u$ flying along the obstacle route calculated by step 10. If it arrives $P$, the state of $u$ reenters the “Accessible Flight” state in step 13-15 and the loop ends, and the algorithm re-plans the route from $P$ to $D$ in step 17. If the algorithm comes to step 20, the state of $u$ comes to “Termination Flight” and the mission finish. Step 21 return the state of $u$ to let the caller know.

3) AN EXAMPLE USING PLDOMD ALGORITHM

Since the obstacle may be an irregular object or there are multiple obstacles on the route, it is necessary to call the PLDOMD algorithm to perform multiple obstacle avoidance until $u$ reaches the destination. Fig. 7 shows an example of $u$ facing a large quadrilateral obstacle to avoid obstacles.

The process of $u$ bypassing large quadrilateral obstacles (Fig. 7) is explained as follows.

1) The UAV $u$ plans a shortest flight $\overrightarrow{S0D0}$ to the ending point $D0$ according to the mission.

2) When $u$ flies to $A1$ point, it is found that there is obstacle on the route. Using the DOMD and PLDOMD algorithm, a Dubins sub-route $A1\overrightarrow{D1}$ is planned on the left side of the original route, where $u$ makes a right-most turn on $\overrightarrow{A1B1}$, then makes a straight flight on $\overrightarrow{B1C1}$, then makes a left-most turn at $C1\overrightarrow{D1}$, and then $u$ flies along the sub-route to point $D1$.

3) $u$ measures no obstacles in front at $D1$ and plans a new route $D1\overrightarrow{D0}$ to the destination.

4) When $u$ reaches $A2$ and measures obstacle on the route, it again uses the DOMD and PLDOMD algorithms, a sub-route $A2\overrightarrow{D2}$ is planned and $u$ flies to point $D2$.

5) $u$ measures no obstacles in front at $D2$ and plans a new route $D2\overrightarrow{D0}$ to the destination, and flies to the ending point $D0$. Until the end of the mission, the route of $u$ is $S0A1\overrightarrow{B1C1D1A2B2C2D2D0}$.

V. EXPERIMENTS AND ANALYSIS

A. SIMULATION EXPERIMENTS

1) SIMULATION SETTINGs

In order to verify the effectiveness of the UAV obstacle avoidance model and DOMD and PLDOMD algorithms, we design a simulation platform on OMNeT++. It has many functions...
such as mobility model, communication model, location positioning model, external expansion model and visualization model that can be utilized. First, several obstacles are placed on the planned route of a UAV; second, when the UAV flies along the route, the obstacle are detected using DOMD algorithm, and the route is re-planned based on DOMD and PLDOMD algorithms until the UAV arrives at its destination. In order to restore the real computing scene on UAV, the algorithm is tested on a microcomputer called Raspberry Pi 3B+,. At the same time, in the subsequent prototype system experiment, we also carried the same computing equipment on the UAV.

2) SIMULATION PLATFORM DESIGN
To build a simulation experiment platform and simulation experiments, we have implemented the simulation platform through conducting the following settings or modifications on the OMNeT++ platform:

1) **The UAV model:** The UAV model is refactoring based on the Mobility module implementation under the INET framework in OMNeT++. We add a route control function to simulate the route change of the UAV during obstacle avoidance flight, and a speed control function to ensure that the speed of UAV always meets the constraints during flight.

2) **Route setting:** In the Euclidean space on OMNeT++ platform, we input a three-dimensional array and use it as the initial route for the UAV to fly from the starting point to the ending point.

3) **Obstacles setting:** By setting obstacle array, the boundary data of the non-passable area is given. Once the position of the UAV is smaller than the detection distance from any value in the array, it is considered that the obstacle has been detected.

4) **Obstacle avoidance process setting:** Once the laser range finder model determines that obstacle avoidance is needed, the DOMD algorithm is called. At this point, the system intermittently reads the sequence of real pictures taken in the real environment according to the the UAV flight speed, and these images include obstacles at different positions. Thereafter, the obstacle avoidance route is planned by calling the PLDOMD algorithm, and the UAV travels on the route until it reaches the ending point.

After completing the design of the platform, in order to test the algorithm, we first verified the feasibility of the platform for obstacle avoidance simulation. Then test the stability of the algorithm by adjusting some key parameters of the algorithm.Fig. 8 shows the interface of the simulation system and some important elements on the interface.

3) METRICS
In this experiment, the following metrics are adopted:

- **Passing Rate:** Because a UAV is not able to fly less than the restraint speed, when it finds obstacle and re-plans the route, it still has the risk of colliding with the obstacle. If the algorithm calculates the obstacle boundary error, it is also possible for the machine to collide with obstacles. In the experiment, we assume that the UAV has a smaller collision radius \( r \). When the distance between the UAV and the obstacle boundary is less than \( r \), the collision is considered and the experiment failed.

\[
\text{Passing rate} = \frac{\text{Number of experiments} - \text{Failure}}{\text{Number of experiments}} \quad (18)
\]

- **Coverage:** In the DOMD algorithm, we treat the filtered points as feature points of the obstacle and use these points to enclose the rectangular obstacle boundary (Fig. 5). In the experiment, we artificially calibrate the feature points that belong to the obstacle before the algorithm filters, and use these points to enclose the obstacle boundary. Therefore, to verify the algorithm, we compare the obstacle boundary selected by the algorithm with the manually calibrated boundary, and have the following relationships:

  1) **Cover:** The range of obstacle boundary calculated by the algorithm is larger than the artificial calibration boundary, which means that a longer avoidance route will be planned when the UAV is doing obstacle avoidance.

  2) **Basic Coincidence:** The range of obstacle boundary calculated by the algorithm is basically coincident with the artificial calibration boundary, which means that the UAV can correctly perform obstacle avoidance.

  3) **Far Less:** The range of obstacle boundary calculated by the algorithm is smaller than the artificial calibration boundary, which means that if the obstacle avoidance flight is performed according to the boundary calculated by the algorithm, it may collide because the actual obstacle is larger than the algorithm measured.

- **Accuracy Rate:** We marked the obstacle area with the help of artificial means, and compared the coverage area of these works in the process of experiment.

\[
\text{Accuracy rate} = \frac{\text{Algorithm calibration area}}{\text{Manual calibration area}} \quad (19)
\]

- **Error Rate:** It represents the margin of error between the obstacle boundary detected by the obstacle recognition
algorithm and the artificial measurement boundary

\[ \text{Error rate} = \text{Accuracy rate} - 1 \quad (20) \]

The negative value means that the obstacle boundary detected by the algorithm is smaller than the actual obstacle boundary, which will lead to collision with the obstacle during obstacle avoidance. The positive value means that the obstacle boundary detected by the algorithm is larger than the actual obstacle boundary, which will cause the UAV to detour further in the process of obstacle avoidance and increase the obstacle avoidance consumption.

**Runtime Required and the Memory Consumption:** We also counted the runtime required and memory consumption of the algorithm.

4) DOMD PARAMETERS INFLUENCES

The constant \( \beta \) in (6) has been mentioned above, and its value will determine the screening range of the feature point changing rate \( \alpha \), which directly affects whether the feature points of the obstacle can be accurately and effectively extracted. Fig. 9 shows the extraction effect of feature points on actual obstacles when \( \beta = 0.6, 0.8, \) and 0.9 respectively.

![FIGURE 9. Effect on feature points with different values of \( \beta \).](image)

TABLE 1 shows the effect of the number and the accuracy of filtered feature points with two frames of \( 480 \times 640 \) resolution, \( \beta = 0 \) indicates that the matching feature points are not filtered.

**TABLE 1. Effect of \( \beta \) value on the judgment of obstacles.**

| \( \beta \) | Number of DOMD filtered feature points | Number of feature points same with artificial filtered feature points | Coverage |
|---|---|---|---|
| 0 | 118 | 45 | Cover |
| 0.5 | 78 | 45 | Cover |
| 0.6 | 62 | 45 | Cover |
| 0.7 | 52 | 45 | Basic Coincidence |
| 0.8 | 44 | 44 | Basic Coincidence |
| 0.9 | 17 | 17 | Far Less |

Comparing and analyzing Fig. 9 and TABLE 1, it can be obtained that when the value of \( \beta \) is between 0.7 \~ 0.8, the effect of extracting obstacles is better. In the subsequent experiments we chose \( \beta = 0.8 \) because it has a smaller gap with the artificial filtered feature points.

We test the actual effect of our algorithm on a microcomputer called Raspberry Pi 3B+. We use its expansion camera module called Raspberry Pi Camera V2 and an external laser range finder called LIDAR-Lite V3. They are all light, cheap and easy to use. The entire hardware system is shown in Fig. 13.

**TABLE 2. Test results of running the DOMD algorithm.**

| Scenes | Number of Tests | Image Resolution | Runtime | Passing Rate |
|---|---|---|---|---|
| Indoor | 10 | 480 \times 640 | 0.14s \~ 0.48s | 100% |
| Outdoor | 12 | 480 \times 640 | 0.27s \~ 0.72s | 91.7% |
| Indoor | 10 | 720 \times 1080 | 0.22s \~ 0.76s | 100% |
| Outdoor | 12 | 720 \times 1080 | 0.40s \~ 1.32s | 100% |

It can be seen from TABLE 2 that in indoor scenarios, regardless of the resolution of the image, DOMD and PLDOMD algorithms could extract obstacles quickly and help the UAV bypass obstacles smoothly. In outdoor scenes, because the picture usually has a complicated background, the DOMD algorithm runs for a long time, and there may be cases where the UAV cannot pass the obstacles correctly (i.e., colliding with the obstacle). The reason for the failure is to use a lower resolution (such as \( 480 \times 640 \)) picture. If the number of feature points extracted is less, some key points for obstacle avoidance will be lost. However, by increasing the resolution of the image (such as \( 720 \times 1080 \)), the passing rate of the DOMD algorithm will increase but the runtime will also increase.

Therefore, when the image resolution is improved, the response time of DOMD algorithm will increase, which is caused by the time-consuming increase of ORB algorithm. When the content of the image becomes complex, ORB algorithm extracts more feature points, and the response time of DOMD algorithm will also increase, which is due to the time-consuming increase of the screening and searching part of the algorithm. These results from Table 2 also support the analysis results of DOMD algorithm response time from (14).

5) MAXIMUM FLIGHT SPEED LIMITATION

There are some limitations in our work, most of them come from the sensor or the UAV itself. However, the maximum flight speed of UAV in obstacle avoidance flight is directly limited by our obstacle avoidance method. Many factors restrict the maximum flight speed of UAV. Here, we ignore the maximum flight speed determined by the hardware of UAV itself, and only consider the quantitative relationship between the maximum flight speed of UAV in our method. When the UAV finds the obstacles and reaches the starting...
point of the Dubins curve, the time used should be greater than the delay of algorithm processing, and the speed should be reduced to the minimum flight speed to meet the requirements of the Dubins curve. Equation (21) shows the relationship about the maximum flight speed and other factors.

\[
\frac{v_m^2 - v_0^2}{2a_0} + v_0 \times \left( t - \frac{v_m - v_0}{a_0} \right) \leq L - r - d \tag{21}
\]

where \(v_m\) is the flight speed of UAV, \(v_0\) is the minimum flight speed, \(a_0\) is the maximum deceleration, \(t\) is the delay of algorithm processing, \(L\) is the obstacle distance that the algorithm starts to run, \(r\) is the minimum turning radius and \(d\) is the collision radius of UAV. After sorting out the formula, we can get:

\[
v_m \leq \sqrt{2a(L - r - d - v_0t)} - v_0 \tag{22}
\]

we can get two limitations from (22). The maximum flight speed of UAV cannot exceed \(\sqrt{2a(L - r - d - v_0t)} - v_0\) and \(L\) must greater than \(r + d + v_0t\).

6) OBSTACLE MAXIMUM VOLUME LIMITATION

For DOMD algorithm, because of the need to extract the boundary of obstacle in the image, when the obstacle fill the whole image, although the lidar can still finish the ranging work between the UAV and the obstacle, the algorithm can not effectively extract the edge of the obstacles, it will lead to the failure of obstacle avoidance. Therefore, in order to ensure the stability of DOMD algorithm, the volume of the obstacle will be limited. Considering the dangerous distance \(d\) set for UAV, the camera internal parameter \((f_x, f_y, c_x, c_y)\) (See Algorithm 1) and its corresponding image resolution \((H \times V)\), we can calculate the maximum volume limit of obstacles with edges in the image:

\[
\text{Width of Obstacle} < H \times d/f_x \tag{23}
\]

\[
\text{Height of Obstacle} < V \times d/f_y \tag{24}
\]

Therefore, when the width and height of the obstacle do not conform to (23) and (24) at the same time, DOMD algorithm may not detect any boundary of the obstacle. This will lead to obstacle avoidance failure. Therefore, when the environment becomes complex and there may be large obstacles, a better method is to change the dangerous distance \(d\) set for UAV, so that the algorithm can extract the boundary of obstacles.

7) OBSTACLE-AVOIDANCE EXPERIMENTS AND RESULTS ANALYSIS

We designed a set of experiments to validate the model and algorithms we designed. Experiment 1 is for the UAV \(u\) to fly through a single small obstacle, and experiment 2 is for \(u\) to fly through complex obstacles, where ‘complex’ refers to more obstacles and larger obstacles. At the same time, in order to compare with other methods, we implement MOPS-SIFT algorithm [26] and improved-PTAM algorithm [27] on our simulation platform and do the same experiment. The MOPS-SIFT algorithm is a method to recognize the obstacle based on monocular vision. This work combine the MOPS and SIFT algorithm together to detect the boundary of the obstacle and the depth of this obstacle. The improved-PTAM algorithm is a method by improving the PTAM algorithm in dealing with poor illumination of indoor environment.

Experiment 1 (Passing a Single Small Obstacle): The experiment let \(u\) fly along the track, where a small obstacle is on the route. The ratio of simulation environment distance to actual distance is 1:1cm. \(u\) starts at coordinate (200, 100), and the obstacle is at (200, 700), and the ending point is at (200, 1000). In the experiment, the actual size of the obstacle is small, so that it only adjusts the route once. We inputted the actual pictures of obstacle to the emulator (same as shown in Fig. 4), and observe whether \(u\) can perform obstacle avoidance flight according to DOMD, MOPS-SIFT and improved-PTAM.

Fig. 10 shows the situation of the simulation platform at different times during the experiment 1. At 0s, \(u\) flies to the ending point and PLDOMD is called. Until 31s, \(u\) enters the dangerous distance of the obstacle, at which time the DOMD is called. The DOMD algorithms cost about 0.14s, and get the result that \(u\) is closest to the right boundary of the obstacle, and plan the Dubins path accordingly. \(u\) starts the obstacle avoidance flight in about 40s along the sub-route obtained by the algorithm. At 60s, \(u\) still on its way to the obstacle avoidance point and arrives at about 65s. At 85s, \(u\) on the re-planned route to the ending point. The mission finishes at about 100s when \(u\) arrives the ending point. We did experiment 1 for 50 times with 10 sets of actual pictures, each set for 5 times. TABLE 3 shows the Passing rate, the Accuracy rate and the Error rate for DOMD, MOPS-SIFT, and improved-PTAM in this experiment. And also we compare with a Machine Learning method called DeepFly [30], which also use the image captured by monocular camera as obstacle avoidance clues. Because we do not have the training data and we can not reproduce the method, we directly use the experimental results given by [30]. We can see that DOMD algorithm has the best passing rate, which is precisely because the algorithm has the highest accuracy rate, to ensure the correct judgment of the shape of obstacles. And we also

---

FIGURE 10. The simulation system displays the process of passing a single small obstacle.
TABLE 3. Passing rate and Accuracy rate of algorithms.

| Algorithm     | Passing rate | Accuracy rate | Error rate   |
|---------------|--------------|---------------|--------------|
| DOMD          | 48/50        | 105%          | =2% ~ 9%     |
| MOPS-SIFT     | 35/50        | 77%           | =37% ~ 9%    |
| improved-PTAM | 44/50        | 120%          | =7% ~ 26%    |

count the Runtime and the Memory consumption for each algorithm to recognize the obstacle in front. Fig. 11 shows that DOMD algorithm has the lowest runtime and memory consumption. MOPS-SIFT algorithm is close to DOMD algorithm in runtime and memory consumption, but it has low accuracy rate in recognizing obstacles (usually the shape of obstacle is smaller than the real), which leads to low passing rate. Improved-PTAM algorithm has the highest runtime and memory consuming. We believe that when dealing with more complex scenes, the number of obstacle avoidance failures caused by judgment timeout will increase, and the passing rate of improved-PTAM will be further reduced. Therefore, compared with the other two algorithms, our DOMD algorithm shows better robustness and cost performance.

Experiment 2 (Passing Complex Obstacles): Fig. 12 shows the complex experimental scenario we designed to validate the PLDOMD algorithm. Ob1, Ob2, and Ob3 are all obstacles, and the volume of Ob3 is larger. P_start and P_end are the starting point and the ending point of the experiment. Their linear distance is 800m. In this experiment, we set up an array of the unreachable areas of Ob1, Ob2, and Ob3, and performed simulation experiment on our platform. The speed of u is set to 4m/s, and the entire PLDOMD algorithm adjusts the flight route for 4 times during the experiment. After 317s, with four obstacle-avoidance flights, u successfully passes the complex obstacle avoidance experiment. The whole flight route is also shown in Fig. 12.

FIGURE 11. (a) Runtime of three algorithms for each picture sets (unit:second); (b) Memory consumption of three algorithms for each picture sets(unit:MB).

We also attempted to use MOPS-SIFT and improved-PTAM on this scenario. But none of them can complete the continuous obstacle avoidance task in this scene. In the face of Ob2, MOPS-SIFT algorithm misjudges the boundary of obstacles, resulting in collision. The performance of improved-PTAM is better than MOPS-SIFT, but in the face of Ob3, the first obstacle avoidance judgment is not enough to completely avoid the obstacle. In the second avoidance judgment, the algorithm runtime is too long, resulting in the inability to immediately avoid the obstacle.

B. PROTOTYPE SYSTEM EXPERIMENT

1) SYSTEM DESIGN

Consistent with the simulation experiment, we use Raspberry Pi 3B+ as computing equipment. At the same time use its expansion camera module called Raspberry Pi Camera V2 and an external laser range finder called LIDAR-Lite V3. They are all light, cheap and easy to use. The entire hardware system is shown in the Fig. 13. We put our hardware system on DJI M100, a medium size quadrotor UAV with external flight control API (Fig. 13). The hardware system also realizes the direct control of UAV flight through the DJI Onboard SDK.

2) REAL SCENE OBSTACLE AVOIDANCE

In order to test the effectiveness of our algorithm in the real scene, we test the whole prototype system in the real world. In order to observe the operation state closely, we put the system in low altitude for flight. In the process of experiment, we constrain the obstacle avoidance algorithm so that it does not choose to reduce the height of the UAV to avoid obstacles, because the lower height may cause the danger of touch the ground due to the GPS positioning error.
FIGURE 13. (a) Hardware system; (b) Prototype system (No wings for safety).

FIGURE 14. Real-world experiment.

The initial state of the UAV is at a low altitude of 1.5 meters above the ground. Its mission is to fly 15 meters to the north. When the UAV starts its mission, its speed will be fixed at 1 m/s for safety, and it will avoid the obstacles that may be encountered during the mission.

Fig. 14 shows the actual situation of our method when facing obstacle in the real scene, the green ellipse marks the position of the UAV and the red square marks obstacle avoidance area. Fig. 14(a) shows initial state of the UAV, there is an obstacle (a tree) in its north direction, the taillights facing the photographer are all green. UAV starts to perform flight mission, flying in the north direction, and keeps close to obstacle (Fig. 14(b)). The UAV recognizes obstacles and begins to avoid it in this second. Because it is necessary to determine whether there are still obstacles in the new obstacle avoidance path, the UAV nose changes with the direction of UAV movement, the taillights facing the photographer change to one red an one green (Fig. 14(c)). The UAV follows the path calculated by the PLDOMD algorithm to reach the obstacle bypass point (Fig. 14(d)). The UAV reroutes to the original target point, changes its nose direction, the taillights facing the photographer change back to all green (Fig. 14(e)). Finally, the UAV reaches the target point and hovers for the next mission (Fig. 14(f)).

VI. CONCLUSION

Based on a low-cost monocular camera and a small-sized laser range finder, this paper proposed an obstacle recognition algorithm and an obstacle avoidance method for resource-constrained UAVs. The obstacle recognition algorithm (DOMD) is designed to extract obstacle boundary from images captured by a monocular camera and laser range finder based on our obstacles recognition model. Then, the obstacle avoidance method (PLDOMD) is proposed to calculate the detour path for the UAV to avoid collision based on the extracted obstacle boundary. To evaluate the performance of our proposal, we design and implement a simulation platform on OMNeT++. Compared with other algorithms, our method has better performance on low-resolution monocular visual images in passing rate and accuracy rate. Moreover, experimental results show that our method has the advantages of low-cost, low computational complexity, and strong robustness. Besides, the lower runtime and memory consumption ensure that our algorithm can run on small embedded systems carried on small UAVs. Finally, our algorithms are validated on a prototype constructed based on DJI M100 UAV to show the practical effectiveness of our methods.

Next, we will try to combine some low complexity optical flow methods and perspective methods to improve the deficiencies of the DOMD algorithm for some image processing. In the future, we will try to install sensors on the side or use Pan-Tilt, and improve the algorithm to recognize obstacles on both sides at the same time, and plan the path to avoid all these obstacles.

REFERENCES

[1] E. Kuiper and S. Nadjm-Tehrani, “Mobility models for UAV group reconnaissance applications,” in Proc. Int. Conf. Wireless Mobile Commun. (ICWMC), Jul. 2006, p. 33.

[2] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, “Supporting wilderness search and rescue using a camera-equipped mini UAV,” J. Field Robot., vol. 25, nos. 1–2, pp. 89–110, Jan. 2008.
[3] Y. Mao et al., “Obstacle Recognition and Avoidance for UAVs,” IEEE Access, vol. 8, 2020, pp. 169421.

[4] E. Karami, S. Prasad, and M. Shehata, “Image matching using SURF, SIFT, FAST and ORB for unmanned air vehicles,” Proc. IEEE Int. Conf. Prog. Informat. Comput. (PIC), Dec. 2015, pp. 543–550.

[5] K. Kelchtermans and T. Tuytelaars, “How hard is it to cross the room? Training (recurrent) neural networks to steer a UAV,” 2017, arXiv:1702.07600. [Online]. Available: http://arxiv.org/abs/1702.07600.

[6] M. Muja and D. G. Lowe, (2017). Fast Library for Approximate Nearest Neighbors. [Online]. Available: https://github.com/mariusmuja/flann.

[7] D. A. Forsyth and J. Ponce, Computer Vision: A Modern Approach. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[8] M. Muja and D. G. Lowe, “Scalable nearest neighbor algorithms for high dimensional data,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 11, pp. 2227–2240, Nov. 2014.
XIANGLIN WEI received the Ph.D. degree from the PLA University of Science and Technology, Nanjing, China, in 2012. He is currently a Researcher with the 63rd Research Institute, National University of Defense Technology, Nanjing. His research interests include mobile edge computing and wireless network optimization. He served as an editorial member or a Reviewer for many international journals and international conferences.

BING CHEN received the B.S. and M.S. degrees from the Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, Jiangsu, China, in 1992 and 1995, respectively, and the Ph.D. degree from the College of Information Science and Technology, NUAA. He is currently a Professor with NUAA. His main research interests include computer networks and embedded systems.