Article

Not peer-reviewed version

A Low-Cost Perception and Control System for In-flight Collision Avoidance of UAVs Based on Background Subtraction

Jeonghwan Park and Andrew Jaeyong Choi *

Posted Date: 29 May 2023

doi: 10.20944/preprints202305.1957.v1

Keywords: unmanned aerial vehicle; collision avoidance; trajectory estimation; feature-point matching; optical flow; background subtraction

Article

A Low-Cost Perception and Control System for In-Flight Collision Avoidance of UAVs Based on Background Subtraction

Jeonghwan Park [1] and Andrew Jaeyong Choi [2,*]

[1] ThorDrive, 165, Seonyu-ro, Yeongdeungpo-gu, Seoul, Republic of Korea; jhpark@thordrive.ai
[2] School of Computing, Dept. of AI-SW, Gachon University, 1342 Seongnam-daero, Sujeong-gu, Seongnam, Republic of Korea; andrewjchoi@gachon.ac.kr
* Correspondence: andrewjchoi@gachon.ac.kr; Tel.: +82-31-750-8667

Abstract: This paper proposes a high-performance and low-cost in-flight collision avoidance system based on background subtraction for unmanned aerial vehicles (UAVs). The pipeline of proposed in-flight collision avoidance system is as follows: (i) dynamic background subtraction to remove the background and to detect moving objects, (ii) denoise using morphology and binarization methods, (iii) Euclidean clustering to cluster the moving object and to remove noise blobs, (iv) distinguish independent objects and track the movement using Kalman filter, and (v) collision avoidance using proposed decision-making techniques. This work focuses on the design and the demonstration of a vision-based fast moving object detection and tracking system with decision-making capabilities to perform evasive maneuvers to replace high vision system such as event camera. The development of high-performance, low-cost unmanned aerial vehicles paired with rapid progress in vision-based perception systems herald a new era of autonomous flight systems with mission-ready capabilities. One of the key features of an autonomous UAV is a robust mid-air collision avoidance strategy. The novelty of our method lies in the motion-compensating moving object detection framework, which accomplishes the task with background subtraction via 2-D transformation approximation. Clustering and tracking algorithms process detection data to track independent objects, and stereo-camera-based distance estimation is conducted to estimate the 3-D trajectory, which is then used during decision-making procedures. The examination of the system is conducted with a quadrotor UAV test vehicle, and appropriate algorithm parameters for various requirements are deduced.

Keywords: unmanned aerial vehicle; collision avoidance; trajectory estimation; feature-point matching; optical flow; background subtraction

1. Introduction

Lately, advancements in machine learning methodologies have enabled the development of vision-based spatial and object recognition systems, and this led to active research in field of autonomous flight control systems, especially for unmanned aerial vehicles (UAVs). However, currently available autonomous flight control systems for UAVs focus mostly on waypoint based global positioning systems (GPS), though the ability to navigate complex and dynamic environments is under active development [1, 2]. Several drone manufacturers have integrated into their quadrotors simple autonomous flight systems that can follow subjects while avoiding static obstacles at low flight speeds, but they lack the ability to handle sudden changes in environments, and the standard function of static obstacle avoidance is still prone to failure.

The recent advances in unmanned aerial vehicles (UAVs) and computing technologies [i.e., artificial intelligence, embedded systems, soft computing, cloud and edge computing, sensor fusion, etc.] have expanded the potential and extended the capabilities of UAVs [3, 4]. One of the key abilities required for autonomous flight systems is the methodology of mid-air collision avoidance [5]. Small-scale UAVs are exposed to various threat sources such as birds and other small aircrafts. Mid-air collisions almost certainly lead to vehicle damage and payload losses. An autonomous collision-avoidance system capable of detecting potential hostile objects and performing decision-making avoidance maneuvers is regarded as a crucial component of an autonomous flight system [6].

Collision-avoidance systems require a means by which to perceive potential obstacles, and this is performed by one or more types of sensors. Typical sensors employed for obstacle recognition include passive sensors such as cameras and active sensors such as RADAR, LiDAR, and SONAR. Cameras, commonly utilized passive sensors, can further be classified into monocular, stereo, and event-based types [7-11]. Cameras typically benefit from their small size and ease of use, but they are sensitive to lighting conditions. Lee et al. [12] employed inverse perspective mapping for object recognition, with the downside of relatively slow operational flight speeds. Haque et al. [13] implemented a fuzzy controller with stereo cameras for obstacle avoidance. Falanga et al. [14] demonstrated low-latency collision-avoidance systems with bio-inspired sensors called event cameras. These sensors require no additional image processing operations, making them ideal for resource-constrained environments such as UAVs. However, event cameras are still in their active development and their prices are still too high for mass deployment.

Active sensors emit energy waves that reflect off object surfaces and measure distances based on the round-trip times of these waves. They are robust to lighting conditions and have a relatively broad operational range but usually heavier and less easy to mount than passive sensors. Kwag et al. [15] utilized RADAR sensors to obtain position and velocity information about nearby objects and execute appropriate maneuvers based on this information. Moses et al. [16] developed a prototype of an X-band RADAR sensor for obstacle recognition from UAVs.

Mid-air collision-avoidance systems are constrained with several requirements. Given that they operate in very short time periods, rapid cognition and response are crucial. Therefore, the computational complexity of the algorithms they use must be considered. In addition, it is desirable to utilize existing sensors such as cameras rather than using additional sensors to minimize the impact on the payload capacity.

Camera sensors benefit from their small size and low power consumption and provide an abundance of real-world information. However, this information is often highly abstracted, and additional processing is required to obtain them. One of the key objectives of vision-based perception algorithms is to require lighter computational power as possible. Furthermore, modern vision-based perception systems are not yet capable of robust and general inference capabilities and require further development.

There are various methodologies for moving object detection from image and video data, including statistical, image geometry-based, and machine-learning-based methods. Lv et al. [17] solved the problem of low accuracy and slow speed of drone detection by combining background difference and SAG-YOLOv5s. This method can be operated on a mobile platform. Chen et al. [18] compared the motion vectors of moving and stationary objects to detect moving objects. Kim et al. [19] clustered optical flow vectors with K-nearest neighbors clustering and distinguished moving objects based on the cluster variance. Seidaliyeva et al. [20] addressed moving objects detection based on background subtraction while classification is performed using CNN.

This research aims to develop and demonstrate a moving object perception and decision-making system for reactive mid-air collision avoidance with low-cost UAV system based on background subtraction method. The concept of applying the proposed system is illustrated in Figure 1.

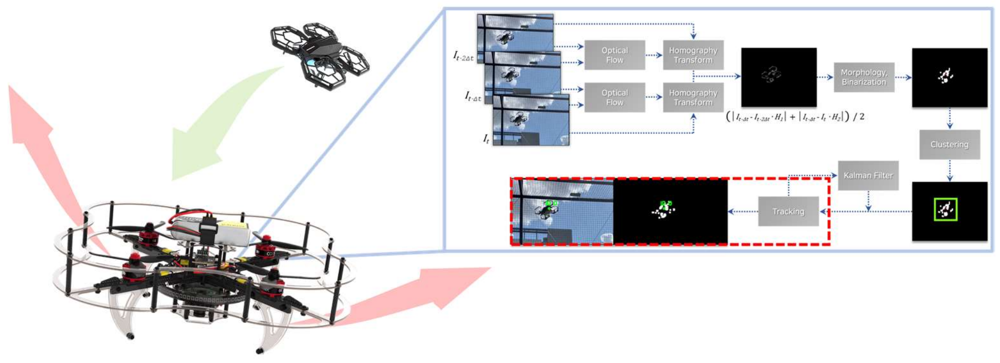


Figure 1. Concept of applying the proposed low-cost moving object avoidance system.

The system utilizes an RGB-D camera to perform image geometry-based moving object detection. To meet the low latency requirements of collision-avoidance systems, a detection algorithm with low computational complexity is devised. The novelty of the proposed system lies in the 2-D perspective transformation approximation of background motion. The corresponding points between images that are needed for approximation calculation is collected using optical flow, and to ensure that the perspective transform model best approximates the background motion, measures such as limiting optical flow estimation regions were employed. These components enable vision-based moving object detection with low computational requirements. The approximated background motion model is utilized for background subtraction to extract regions in which a moving object is present. To cope with inevitable noise originating from the approximation process and the visual data per se, various image filters and a binarization strategy are utilized. Custom-made clustering and tracking modules perceive and track individual objects for threat assessment. The distance to an object is measured by a stereo camera and is then used to estimate the object's 3-D trajectory relative to the ego-vehicle. This trajectory information is utilized for decision making, triggering various commands that allow the ego-vehicle either to avoid the object or perform an emergency stop.

To test the system, a quadrotor UAV equipped with a Raspberry Pi 4 low-power computer and a low-cost Intel RealSense D435i RGB-D camera with visual odometry camera are mounted on the platform. The proposed hardware system is shown in Figure 2. Evasive maneuvers from various conditions are tested, and the results are used to optimize the operational parameters further to better suit each flight environment.

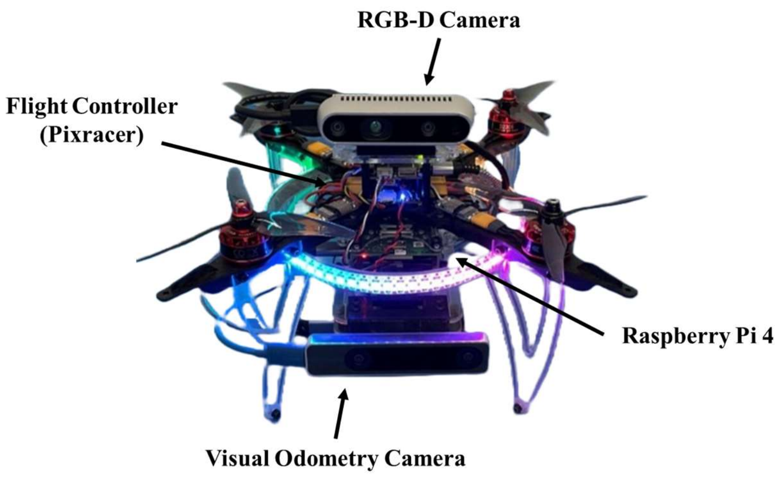


Figure 2. Proposed hardware setting for the low-cost moving object avoidance.

2. Methodology of Moving Object Detection from a Moving Camera

2.1. Projective trnasformation

Transformation in image processing refers to a function that maps an arbitrary point to another point. 2-D transformation is a specific category of transformation, which performs 2-D to 2-D mapping. The most general form of 2-D transformation is projective transformation, which can be interpreted as a type of transformation that maps an arbitrary quadrangle onto another quadrangle. Thus, projective transformation can describe the relationship between two images that show the same planar object. The matrix that performs projective transformation is called a homography matrix, and the transformation can be defined as follows:

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

where $\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix}$ is the pixel coordinate after the transform.

A homography matrix has eight degrees of freedom, and at least four matches between images are required to compute this matrix. Generally, more than four matches are processed with outlier-rejection techniques such as RANSAC [21] or LMedS [22] to yield optimal results.

In principle, projective transform can only describe planar, or 2-D, objects. However, as shown in Figure 3, if 3-D objects are relatively far away from the camera and the posed differences between viewpoints are minor, the relationship between the images taken from these viewpoints can be approximated with projective transformation. This is a core assumption of a moving object detection system. Details about how to utilize this assumption for background motion modeling is provided in the next section.

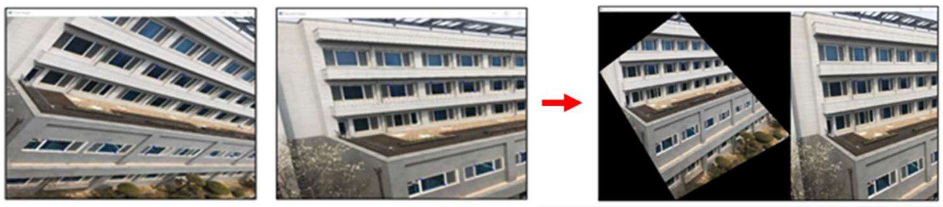


Figure 3. Approximation and transformation of background change between two viewpoints with projective transformation.

2.2. Feature-point matching between images with an optical flow

The optical flow methodology estimates the motion vectors of pixels during two consecutive frames. It involves a calculation of a differential equation called the gradient constraint equation, and methodologies for solving this equation consist of various optical flow algorithms. Some of the popular methods are the Lucas-Kanade and Horn-Schunck algorithms. The Lucas-Kanade algorithm only tracks specified pixels but is computationally less complex. The Horn-Schunck algorithm tracks all pixels in a frame but with less accuracy and at lower speeds. For the proposed system, the Lucas-Kanade optical flow is employed to match pixels between consecutive image frames. A combination of epipolar geometry, projective transformation and an optical flow enables background motion modeling and moving object detection. The detailed procedures are presented in the following section.

2.3. Moving object detection from a moving camera

The mid-air collision-avoidance system proposed here consists of four operational stages: Detection, Recognition, Tracking, and Decision making. The Detection stage detects regions of moving objects in the camera's field of view. A pseudocode of this stage is given in Algorithm 1.

Algorithm 1 Independent moving object detection

```
Input: $I_{t-1}, I_t, I_{t+1}, H_{t-1,t}$
Output: $R_t$
1: $FP \leftarrow [0\ 0\ 0\ 0\ 0\ 0]$
2: $i = 0$
3: while $i < 6$:
4:     $FP[i] \leftarrow$ number of feature points in $I_t[i]$
5:     if $FP[i] < n$:
6:         Find at least $n - FP[i]$ feature points in $I_t[i]$
7:     end if
8: end while
9: $OF_{t,t+1} \leftarrow$ optical flow between $I_t[i]$ and $I_{t+1}[i]$    ► Track background feature points
10: $H_{t,t+1} \leftarrow$ estimate homography between $I_t[i]$ and $I_{t+1}[i]$ using $OF$    ►Calculate homography matrix
11: $R_t \leftarrow average([(I_t - H_{t-1,t} \cdot I_{t-1}) + (I_{t+1} - H_{t,t+1} \cdot I_t)])$    ►Background subtraction
12: $R_t \leftarrow Binarize(Dilation(MedianFilter(R_t)))$    ►Apply denoising filters and binarization
```

The purpose of homography matrix calculation between consecutive images is to model background scene transitions caused by camera motion. As mentioned in section II-B, if 3-D objects are relatively far away and the pose difference between the viewpoints is small, the transition between two video frames can be approximated as a projective transformation. Figure 4 visualizes this approximation process.

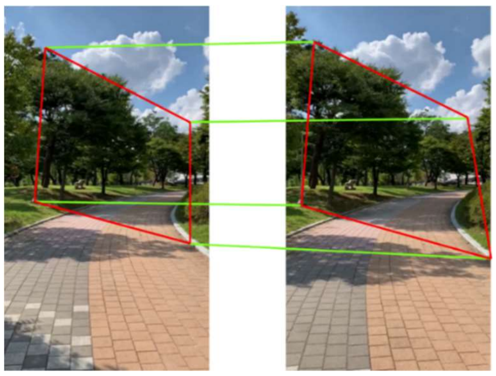


Figure 4. Projective transformation approximation of two consecutive video frames.

The calculated homography matrix is then used for background subtraction. This matrix approximately relates pixels from the previous frame to pixels of the current frame. That is, pixels from the previous frame can be mapped to coincide with those of the current frame. However, the motions of moving objects are not described by the homography matrix and thus cannot be mapped to coincide between frames. Therefore, they can be detected by searching for regions where pixels are not correctly mapped between consecutive frames. To locate incorrectly mapped pixels and detect moving objects, a perspective transform-based background subtraction technique is employed. The homography matrix between two consecutive frames is used to map pixels of the previous frame to match those of the current frame. The transformed frame is then subtracted from the current frame. The pixels of moving objects are not correctly mapped to the current frame given that the motion of the current frame differs from that of the background; accordingly, the corresponding brightness has a nonzero value after subtraction, signifying that image regions of moving objects are detected. An example result is shown in Figure. 5.

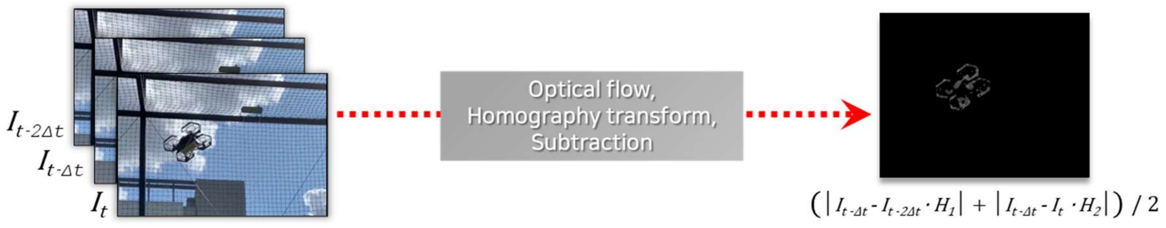
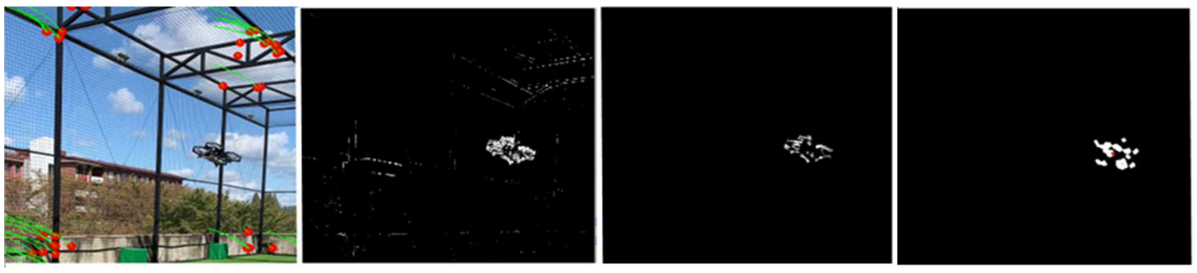

$I_{t-2\Delta t}$
$I_{t-\Delta t}$
I_t
Optical flow, Homography transform, Subtraction
$(|I_{t-\Delta t} - I_{t-2\Delta t} \cdot H_1| + |I_{t-\Delta t} - I_t \cdot H_2|)\ /\ 2$

Figure 5. Original image and background subtracted result.

Also, when calculating the homography matrix, the flight conditions of UAVs were taken into consideration. From the viewpoint of a flying UAV, objects near the center of the field of view are typically further away than those at the edge areas. Further objects appear smaller and thus exhibit weaker magnitudes of the appearance transition. Thus, objects near the edge of the field of view have more influence on the perspective transformation model. Therefore, it is reasonable to calculate the homography matrix from the transition of pixels that are close to the edge of the field of view. In this system, six 70x70 windows were used. The system constantly monitors the number of tracked points in these windows and initiates new tracks if the number falls below a predetermined threshold. This evenly distributes the number of tracked points and guarantees the stability of the homography matrix calculation. To filter out false matches and outliers, RANSAC was used to calculate the homography matrix. Limiting the optical flow calculation regions reduced the computational requirements and improved the approximation accuracy simultaneously.

However, because the projective transformation approximation is not perfect, noise elements inevitably exist. That is, even after background subtraction, some areas of the background may not have zero brightness values. To compensate for this, the proposed system utilizes two methods. The first utilizes the next frame as well as the previous frame for background subtraction. Each transformed frame is subtracted from the current frame and then averaged. Because noise elements appear randomly and momentarily, this method improves the signal-to-noise ratio (SNR) of the result image. The second method applies image filters and binarization to the resulting image after the first method. A median filter eliminates small noise components while a dilation filter expands elements, thus revealing moving object regions more clearly. The image is subsequently binarized to disregard pixels below a certain brightness threshold. This process is illustrated in Figure 6.



(a) (b) (c) (d)

Figure 6. Image Processing for background subtraction (a) The original frame, (b) background subtracted frame (c) median filtered, and (d) dilated and binarized.

3. Methodology for Object Recognition, Tracking and Decision Making

3.1. Object recognition

The Recognition stage applies a clustering algorithm to the result of the Tracking stage to determine the number and locations of all moving objects. A pseudocode of this stage is given in Algorithm 2.

Algorithm 2 Independent object recognition

```
Input: $R_t$
Output: $C_t$
1: $C_t \leftarrow \emptyset$
2: $for$ $unallocatedBlob$ $in$ $R_t$:                                 ▶ A region in $R_t$
3: $if$ $size(unallocatedBlob) < sizeThresh$:
4:     $continue$                                                      ▶ Disregard small regions
5: $for$ $cluster$ $in$ $Clusters$:
6:     $for$ $blob$ $in$ $cluster$:
7:         $if$ $distance(blob, unallocatedBlob) < distThresh$:
8:             $allocate$ $unallocatedBlob$ $to$ $cluster$
9:             $break$                                                 ▶ Add region to cluster if below distance threshold
10:    $if$ $unallocatedBlob$ $is$ $unallocated$:
11:        $allocate$ $unallocatedBlob$ $to$ $Clusters$ $and$ $set$ $as$ $cluster$   ▶ Set as new cluster
12: $for$ $cluster$ $in$ $Clusters$:
13: $append$ $weighed$ $mean$ $position$ $of$ $cluster$ $to$ $C_t$      ▶ Append representative position of blob to $C_t$
```

The Recognition phase outputs a binary image that displays the regions of moving objects as 1. However, as the presence of noise is inevitable, some background regions have a value of 1, albeit intermittently. Additionally, a single independent moving object can appear in several "patches," as shown in Figure 7. Therefore, there is need for a method capable of rejecting noise components while also recognizing multiple nearby patches as a single object.

For the proposed system, a modified DBSCAN (density-based spatial clustering of applications with noise [23]) clustering algorithm is developed. DBSCAN, unlike center-based algorithms such as K-means [24], determines if a data point belongs to a cluster based on its distances to other points. It regards data points with no neighbors as outliers and data points with more than some number of neighbors as inliers. However, a vanilla DBSCAN cannot be directly applied for this task, as there are instances in which an object shows up as a single patch as well as those where noise components appear as several patches. Thus, the vanilla DBSCAN is modified to utilize the area information of patches. Only patches within an area threshold are considered as a true object, and a single patch is considered as a cluster if it satisfies the area threshold. The centroid of a cluster is calculated using the weighed sum of all patches in that cluster. This procedure is summarized in Figure 7.

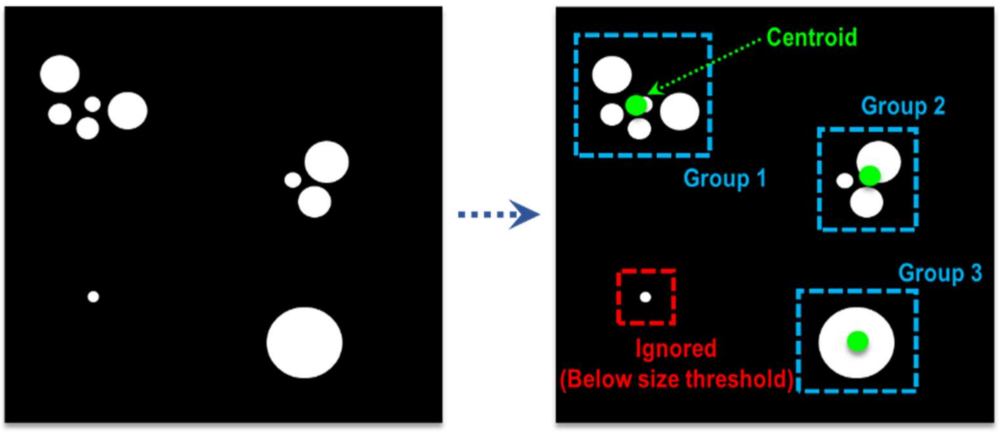


Figure 7. Clustering procedure of the modified DBSCAN algorithm.

3.2. Object tracking

The Tracking stage receives the objects' locations at each frame from the Recognition stage and associates them to those of the previous frame. A unique ID is assigned to each independent object and the corresponding trajectories are monitored for threat assessment. Additional noise compensation is also done in this stage. A pseudocode of this stage is given in Algorithm 3.

Algorithm 3 Independent object tracking

```
Input: $\boldsymbol{O}_{t-1}, \boldsymbol{C}_t$
Output: $\boldsymbol{O}_t$
1: $\boldsymbol{O} \leftarrow \emptyset$
2: $M = size(O_{t-1})$    ▶ number of objects at previous frame
3: $N = size(C_t)$    ▶ number of objects at current frame
4: $Array \leftarrow zeros(M, N)$    ▶ An array that stores distances between all objects
5: $\boldsymbol{for}\ i\ \mathbf{in}\ O_{t-1}$:
6:    $\boldsymbol{for}\ j\ \mathbf{in}\ C_t$:
7:       $Array[i, j] = distance(O_{t-1}[i], C_t[j])$
8: $\boldsymbol{for}\ k\ \mathbf{in}\ C_t$:
9:    $\boldsymbol{if}\ ID[k]\ \boldsymbol{is\ not}\ set$:
10:      $\boldsymbol{if}\ Array[i, k] == \min(Array[i, :])$:    ▶ object $k$: closest object to object $i$
11:         $\boldsymbol{if}\ \ Array[i, k] < distThresh$:    ▶ if below distance threshold:
12:            $ID[k] \leftarrow ID[i]$    ▶ assume $k$ and $i$ is the same object(tracking success)
13:            $Tracked\ length\ of\ ID[k] \mathrel{+}= 1$    ▶ Increase tracked length
14:            $\boldsymbol{break}$
15:         $\boldsymbol{else}$:
16:            $mark\ ID[i]\ as\ lost$    ▶ Mark as lost if no object is nearby
17: $\boldsymbol{for}\ k\ \mathbf{in}\ C_t$:
18:    $\boldsymbol{if}\ ID\ of\ k\ \boldsymbol{is\ not}\ set$:
19:       $assign\ new\ ID\ to\ k$    ▶ assign new $ID$ if new object is detected
20:       $tracked\ length\ of\ ID[k] \leftarrow 0$
21:    $\boldsymbol{if}\ tracked\ length\ of\ ID[k] > noiseThresh$:    ▶ If tracked longer than threshold
22:       $ID[k] \leftarrow mark\ as\ true\ object$    ▶ Approve as true object
23:    $add\ position\ and\ ID\ of\ k\ to\ \boldsymbol{O}_t$
```

Temporal information is used to distinguish true observations from noise and to recognize object identities across multiple frames. This step operates on the central assumption that the distance travelled by an object between frames is shorter than the distances between separate objects. Also, because noise components are spawned intermittently and momentarily, they do not appear consistently across multiple frames.

First, the distances between previously detected coordinates and new coordinates are compared, after which the closest new detection within a distance threshold is recognized as the new position of the object. If there are no new detections within a distance threshold, the object is marked as lost. If no new detections appear near its last known location for a time threshold, the object is deleted.

New detections with no associations are identified as object candidates and are given a temporary ID. If an object candidate is successfully tracked for a time exceeding a certain time threshold, it is approved as a true object and given a new ID. This procedure is summarized in Figure 8.

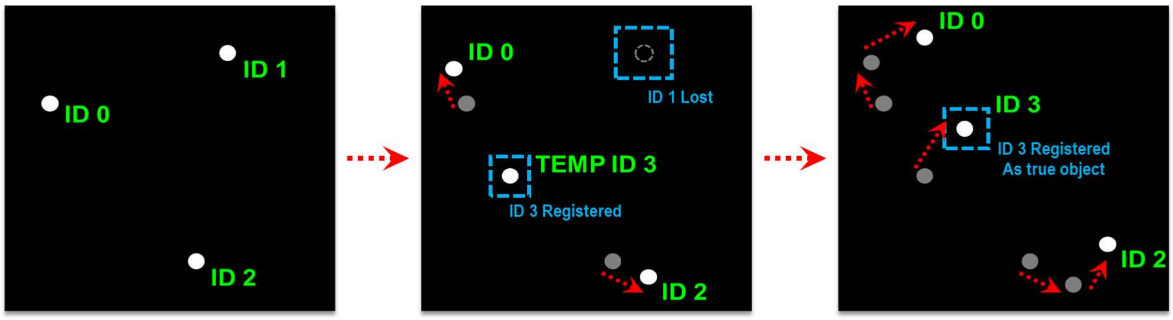


Figure 8. Tracking procedure of the distance-based tracking algorithm.

3.3. Decision making for avoidance maneuvers

The decision-making stage receives the trajectory data from the tracking stage and determines whether avoidance maneuvering must be performed, and if so, in what direction it should occur. A pseudocode of this stage is given in Algorithm 4, and Figure 9 presents a diagram of this algorithm.

Algorithm 4 Decision making for avoidance maneuvers

```
Input: $O_t$                                         ▶ $O_t$: object IDs and locations at current frame
Output: $A_t$                                        ▶ $A_t$: object IDs and locations at current frame
1:  for object in $O_t$:
2:    position ← 3D position of object               ▶ acquire 3D position of object
3:    if position.z < $d_{stopThresh}$ :
4:      emergency stop before proceeding
5:      break
6:    if position is inside $R_{SafeWindow}$:
7:      mark object as hostile                       ▶ consider object as a threat if inside SafeWindow
8:      if position.z < $d_{avoidThres}$ :           ▶ execute avoidance maneuver if distance is below threshold
9:        if position.xy in upperRegion:
10:         move downward at v m/s until outside of $R_{SafeWindow}$
11:       if position.xy in lowerRegion:
12:         move upward at v m/s until outside of $R_{SafeWindow}$
13:       if position.xy in leftRegion:
14:         move right at v m/s until outside of $R_{SafeWindow}$
15:       if position.xy in rightRegion:
16:         move left at v m/s until outside of $R_{SafeWindow}$
17:       break
```

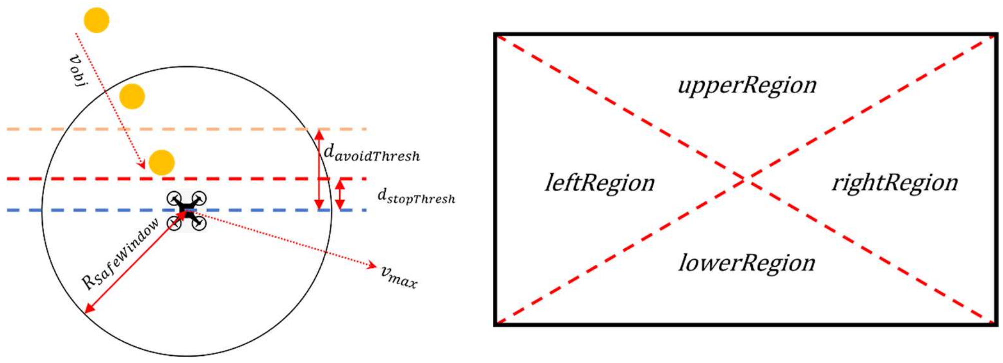


Figure 9. Illustration of avoidance maneuvers.

The proposed system uses the object trajectory information from the tracking system to perform avoidance maneuvers if the relative distance falls below a certain threshold. Additionally, if the initial measured distance to the object is too close (for example, if the moving object approached from the side of the vehicle outside the field of view), an emergency stop occurs to prevent a crash into the obstacle. The radius of the safe window and the avoidance threshold can be determined from prior information, such as the object's expected maximum velocity and the maximum possible maneuver velocity.

4. Performance Validation of In-Flight Collision Avoidance

In this section, the performance validation of the proposed in-flight collision avoidance system will be demonstrated. Before the actual in-flight collision avoidance testing, in-flight dynamic object detection based on background subtraction was conducted. Then, a simulation test was conducted using ROS Melodic with Gazebo environment. After the simulation testing, in-flight dynamic object detection based on background subtraction was conducted. Lastly, the proposed system (in-flight collision avoidance) was tested with actual quadcopter drone.

4.1. System setting for in-flight collision avoidance

The UAV hardware setting for the proposed in-flight collision avoidance shown in Figure 2. It is equipped with a Raspberry Pi 4 companion computer, a RealSense D435i stereo depth camera, and a RealSense T265 visual odometry camera for indoor autonomous flight. The system components are interconnected via ROS (Robot Operating System), a robotics development software package as illustrated Figure 10.

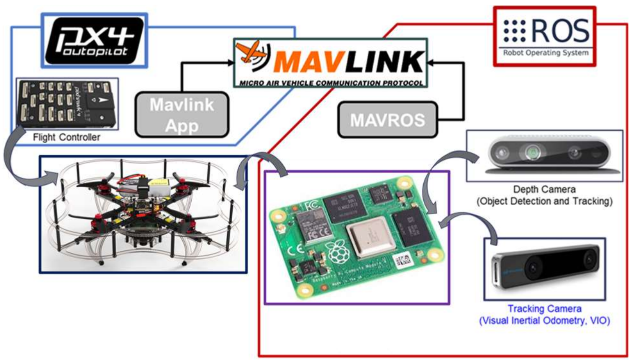
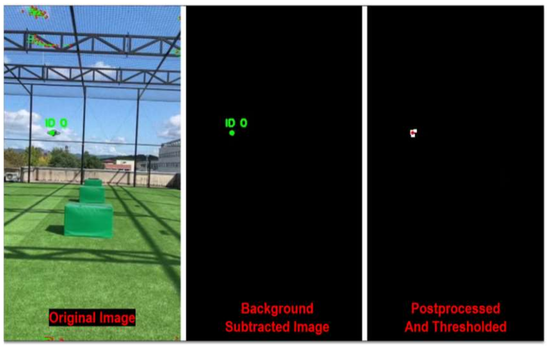


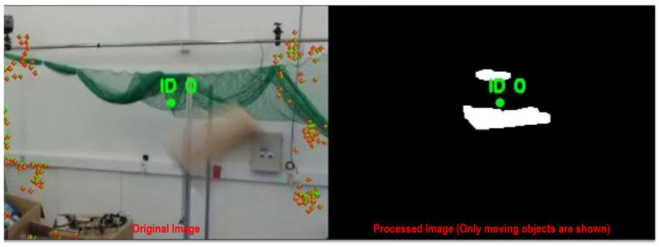
Figure 10. Software and hardware integration for the proposed system.

4.2. Validation of moving object detection and tracking

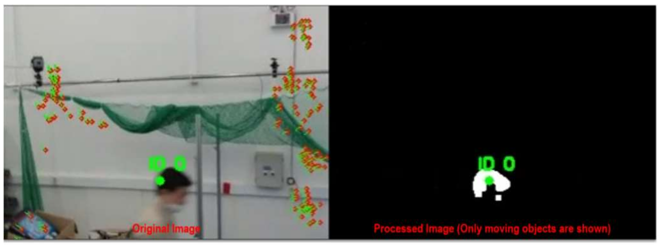
Before the actual in-flight collision avoidance testing, the in-flight dynamic object detection based on background subtraction was conducted as demonstrated in Figure 11. As the Figure 11 shows the in-flight UAV was able to subtraction the dynamic background and detected only the moving object.



(a) Result of moving drone detection



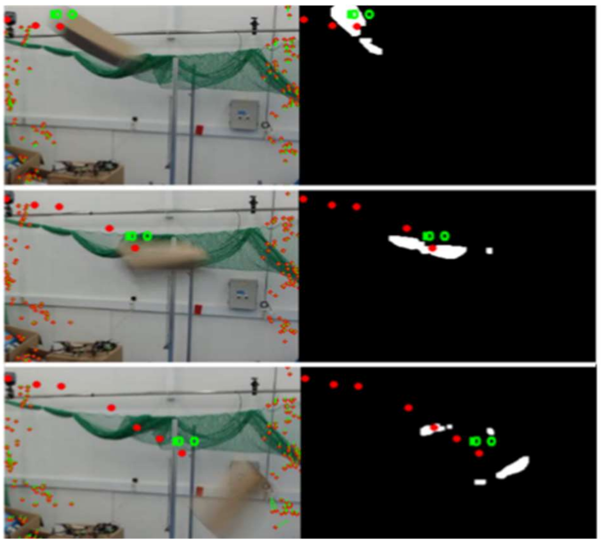
(b) Result of moving box detection



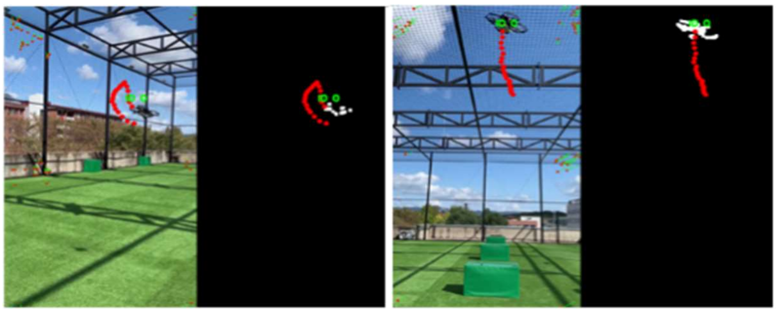
(c) Result of moving human detection

Figure 11. Moving object detection based on background subtraction.

Furthermore, the detection and Kalman filter based tracking algorithm. The tracking results for video sequences are shown in Figure 12. The IDs of objects are assigned for independent objects, and the 2-D trajectory of the object is displayed as red points. The video result is available at https://youtu.be/hRzMDDpLB4s.The tracking algorithm works in real-time on a Raspberry Pi 4 computer. It compensates for noise components and keeps track of the objects even in the detection failure.



(a) Result of moving box detection and tracking



(b) Result of moving drone detection tracking

Figure 12. The results of moving object detection and tracking.

The performance validation for moving object detection and tracking algorithms were also conducted for real-time performance. The fps (frames-per-second) performance comparison results with FastMCD [15] and Neural Network based system [25] are displayed in Figure 13. The blue line represents FastMCD, the orange line represents the neural-network-based system, and the grey line represents the proposed system. As shown in the figure, the proposed system is vastly and continuously superior in terms of real-time processing performance, processing up to 70 fps and 60 fps on average. Note that the stereo camera system utilized for the collision avoidance system streams video data at 30 frames per second, and the proposed algorithms always performed sufficiently above real-time requirements for the task of in-flight moving object detection, tracking and collision avoidance. The validation demonstrated the real-time suitability for in-flight collision-avoidance tasks in real-time.

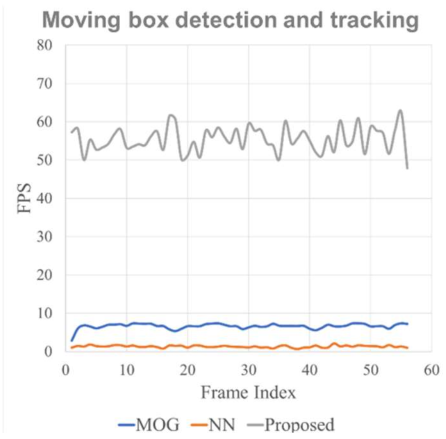


Figure 13. The fps processing performance comparison with other moving object detection systems.

4.3. Validation of in-flight collision avoidance

Before the actual in-flight testing, a simulation test was conducted using ROS Melodic with Gazebo environment as shown in Figure 14.

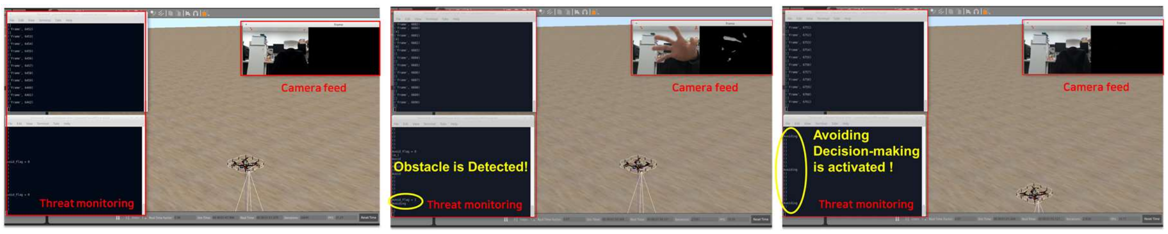


Figure 14. Demonstration of the in-flight collision avoidance in ROS Gazebo simulation environment.

After the simple simulation test, the operational variables were set for the actual in-flight collision avoidance. To validate the proposed system, the drone was set to fly forward until avoidance maneuvers were required. The operational variables were set as listed in Table 1.

Table 1. Operational Variables for the in-flight collision avoidance.

Algorithm 1 Independent moving object detection			
image resolution	boundary region size	Number of regions	
424 x 240	70 x 70	10	
median filter size	dilation filter size	binarization threshold	
5 x 5	5 x 5	40	
Algorithm 2 Independent object recognition			
$sizeThresh$	$distThresh$		
25px	100px		
Algorithm 3 Independent object tracking			
$distThresh$	$noiseThresh$		
150px	3 frames		
Algorithm 4 Decision making for avoidance maneuvers			
$R_{SafeWindow}$	$d_{avoidThresh}$	$v_{[illegible]}$	$d_{emergencyThresh}$
1m x 1m x 1m	1m	3 m/s	1m

For the first and second validation test, the emergency stops were demonstrated. The drone detected an obstacle closer than 0.5 meters away and performed the emergency stop before proceeding. These results are shown in Figure 15, 16, and 17 with plots of the trajectories of the drone and the object. Table 2 is the list of various detection results and the avoidance performance metrics during in-flight performance.

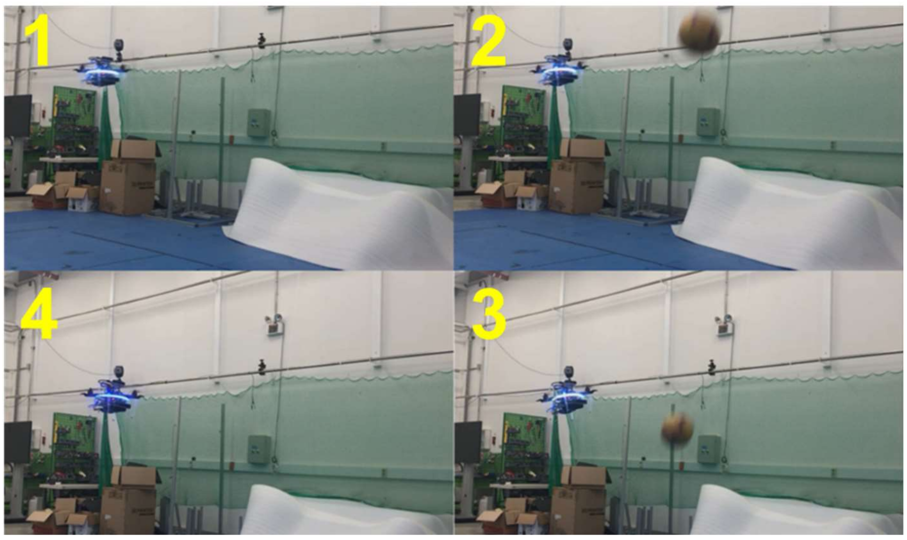


Figure 15. Demonstration of the emergency stop 1.

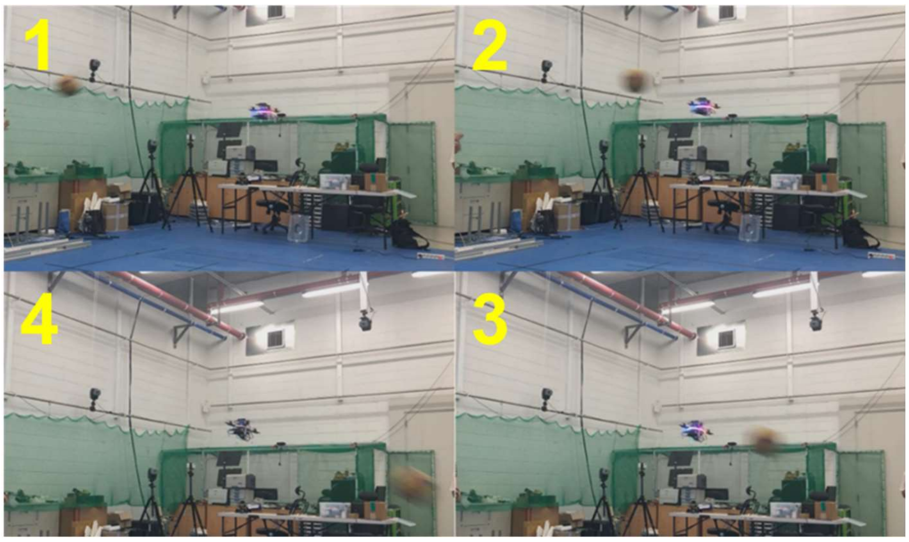


Figure 16. Demonstration of the emergency stop 2.

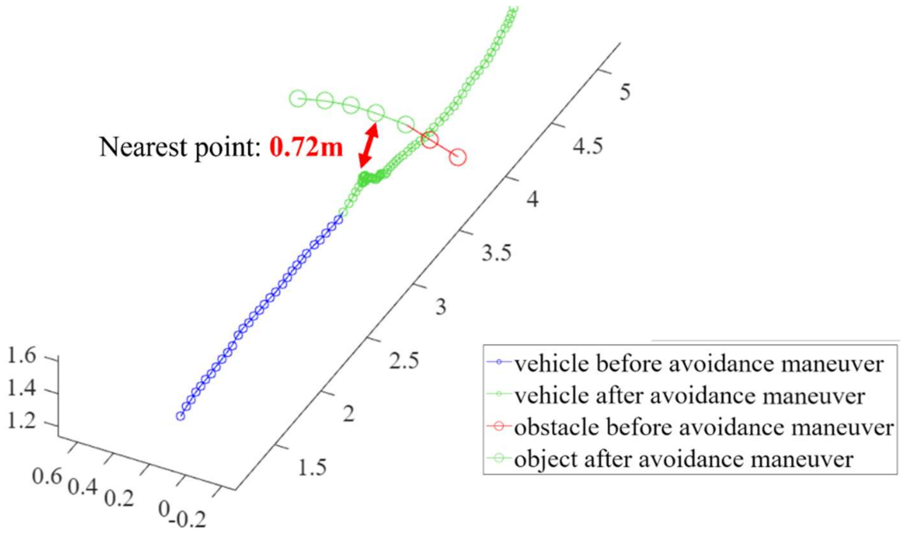


Figure 17. The trajectories of the drone and object.

Table 2. The Detection results and avoidance performance metrics.

Emergency stop		
vehicle speed	obstacle speed	relative speed
1.2m/s	5.9m/s	6.0m/s
time between detection and recognition		minimum distance to obstacle
0.09 seconds		0.72m

For the third and fourth validation test, the avoidance maneuvers were demonstrated. The vehicle detected and tracked an obstacle from outside the safe window and performed avoidance maneuvers when the object breached the avoidance threshold. These results are shown in Figure 18, 19, 20, and 21 present a plot of the trajectories of the vehicle and the object, and the nearest distance between them at the same point in time. Table 3 is a list of various detection results and avoidance performance metrics during the process.

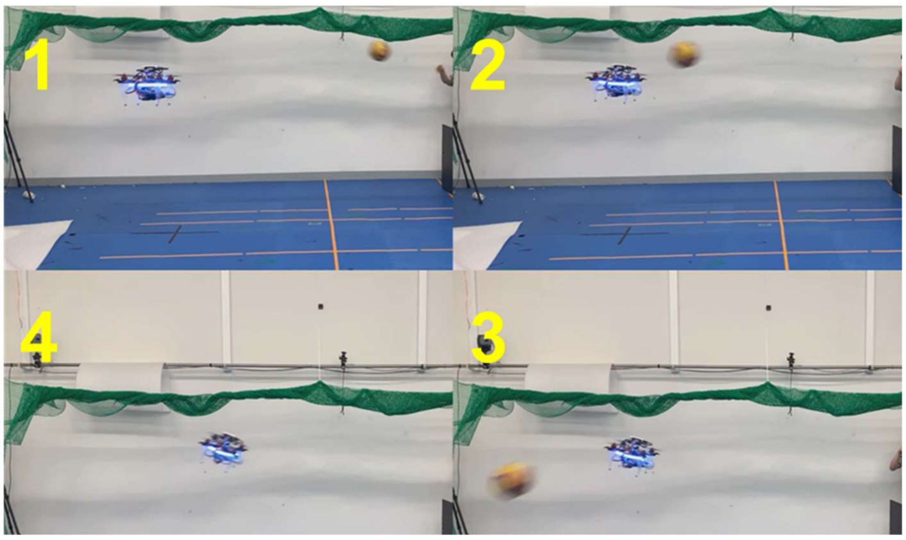


Figure 18. Demonstration of the avoidance maneuver 1.

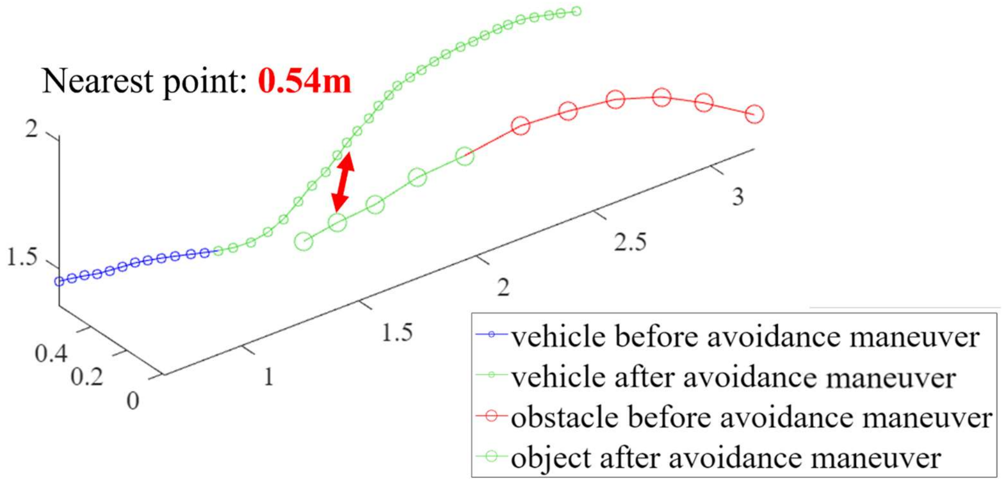


Figure 19. The trajectories of the drone and object for the avoidance maneuver 1.

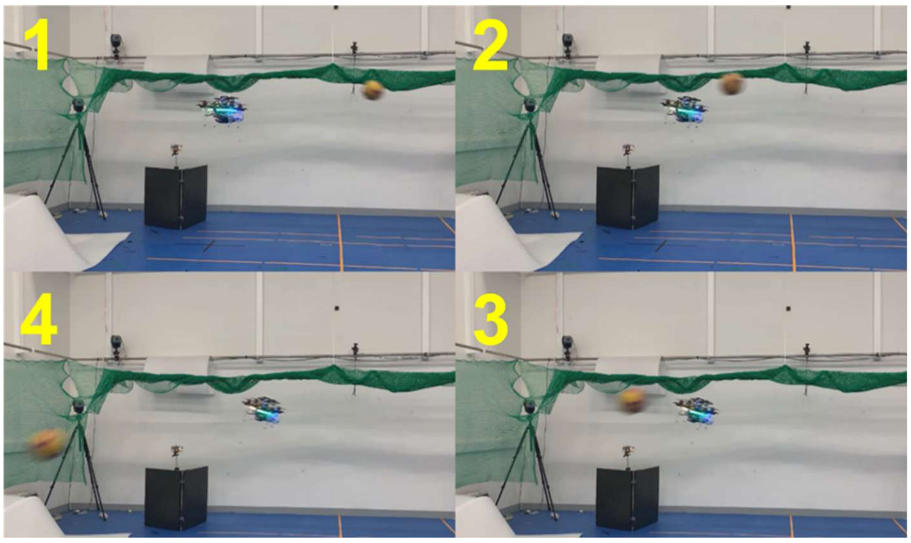


Figure 20. Demonstration of the avoidance maneuver 2.

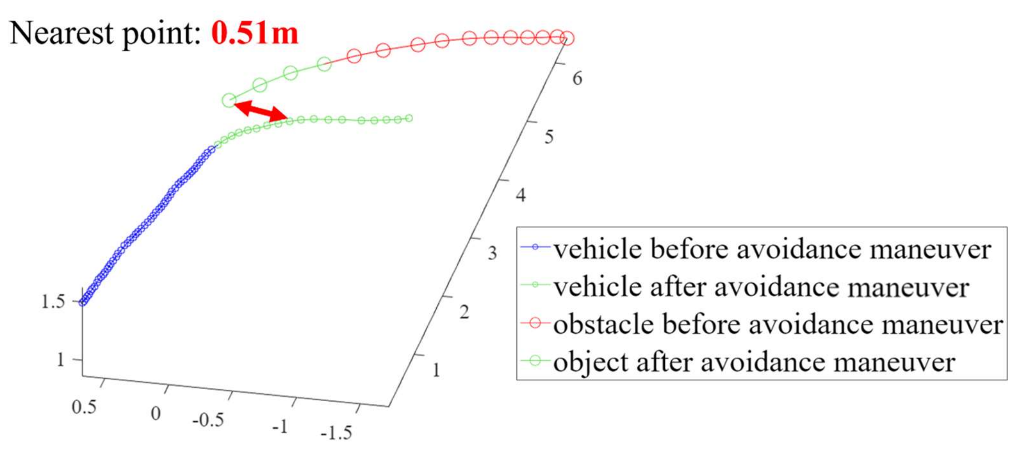


Figure 21. The trajectories of the drone and object for the avoidance maneuver 2.

Table 3. The Detection results and avoidance performance metrics.

Avoidance maneuver 1		
vehicle speed	obstacle speed	relative speed
1.2m/s	5.7m/s	6.7m/s
time between detection and recognition		minimum distance to obstacle
0.1 seconds		0.54m
Avoidance maneuver 2		
vehicle speed	obstacle speed	relative speed
1.2m/s	5.6m/s	5.9m/s
time between detection and recognition		minimum distance to obstacle
0.11 seconds		0.51m

Figure 22. presents the detection, recognition, and tracking result from the vehicle's point of view.

To demonstrate the effectiveness of our approach, we compare some of our test results to those of [25], which is an event camera-based obstacle avoidance system. Although the types of sensors used for each systems differ, the components of the algorithms are very similar: ego-motion compensation, thresholding, morphological operations, and clustering. The algorithm in [25] runs much faster than ours – up to 280 fps, thanks to a simpler ego-motion compensation process that is made possible with the characteristics of the event camera, whereas our approach is bottlenecked by feature point extraction and homography computation processes. Nevertheless, the results for indoor obstacle avoidance for each method are comparable – the approach in [25] was analyzed to be capable of avoiding dynamic obstacles with relative speeds of up to 10 m/s, with reactions times of below 0.2 seconds. Our approach was tested and demonstrated with dynamic obstacles with relative speeds of up to 6m/s, with similar reaction times. We presume that a higher-powered computer, such as the NVIDIA Jetson TX2 used in [25], our proposed system would be able to process higher-resolution images at a constant time rate, which would certainly help with detecting objects which are further away or with higher relative speeds.

5. Conclusions

This paper proposed a high-performance and low-cost in-flight collision avoidance system based on background sub-traction for unmanned aerial vehicles (UAVs). A novel vision-based mid-air collision-avoidance system for UAVs was proposed and implemented. The pipeline of proposed in-flight collision avoidance system is as follows: (i) dynamic back-ground subtraction to remove the background and to detect moving objects, (ii) denoise using morphology and binarization methods, (iii) Euclidean clustering to cluster the moving object and to remove noise blobs, (iv) distinguish independent objects and track the movement using Kalman filter, and (v) collision avoidance using proposed decision-making techniques. Performance validation tests were conducted in simulation environment and with actual quadcopter drone with vision sensors.

The key contribution of the proposed system is a lightweight, error-compensating moving object detection and recognition algorithm. Specifically, background scene transitions due to ego-motion are approximated with 2-D projective transformation, and a homography matrix is computed using the optical flow. To reduce the required computational load and improve the quality of approximation, the optical flow is calculated only at the edge regions of video frames. The previous and successive frames are transformed to match the current frame, and background subtraction is performed to acquire a primitive estimate of the object location. Image filters and thresholding are utilized to improve the SNR of this result. A modified DBSCAN clustering algorithm is used to identify multiple detected image patches correctly as a single object. A distance-based tracking algorithm assigns object identities and tracks them across frames, incorporating an additional noise-filtering procedure based on the tracked period. A stereo camera system measures the distances to detected objects, and this information is used for determining whether avoidance maneuvers must be executed.

Further research on several aspects of the current system can improve low-cost in-flight collision avoidance. Future work will attempt to develop an object classification system for more reliability and robustness of the system by developing light CNN network with 6-DoF pose estimation algorithm to detect an orientation of the moving objects.

The proposed system effectively detects, recognizes and tracks moving objects in its field of view with low computational requirements and achieves sufficient processing performance on a low-power onboard computer. It is implemented onto a test vehicle and performs mid-air collision avoidance to demonstrate its effectiveness in real-world conditions.

Supplementary Materials: The following supporting demonstration video can be viewed at: https://youtu.be/hRzMDDpLB4s.

Author Contributions: Conceptualization, J.P. and A.J.C.; methodology, J.P.; software, J.P.; validation, J.P. and A.J.C.; formal analysis, J.P. and A.J.C.; investigation, J.P. and A.J.C.; resources, A.J.C.; data curation, J.P. and A.J.C.; writing—original draft preparation, J.P.; writing—review and editing, A.J.C.; visualization, J.P. and A.J.C.; supervision, A.J.C.; project administration, A.J.C..; funding acquisition, A.J.C.. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Gachon University Research Fund of 2022(GCU-202300680001).

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Not Applicable.

Acknowledgments: This research was supported by the Gachon University Research Fund of 2022(GCU-202300680001).

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

I_t	=	t^{th} frame of a video sequence
$I_t\,[i]$	=	i^{th} edge area of frame I_t
N	=	threshold number for a new feature point search
$OF_{t,\,t+1}$	=	optical flow between the edge areas of I_t and I_{t+1}
$H_{t-1,\,t}$	=	homography matrix that maps I_{t-1} to I_t
R_t	=	binary image that displays regions where moving objects are present

C_t	=	list of prior object coordinates
O_t	=	object locations and IDs at the previous frame
$d_{avoidThresh}$	=	avoidance decision threshold
$d_{stopThresh}$	=	emergency stop decision threshold
$R_{safewindow}$	=	safe window radius
v_{obj}	=	maximum expected approach speed of object
v_{max}	=	maximum possible maneuver speed of the ego-vehicle

References

1. Floreano, D., and Wood, R. J. Science, technology and the future of small autonomous drones. *Nature*. 2015, 521(7553), 460-466.
2. Cai, Z., Lou, J., Zhao, J., Wu, K., Lui, N., and Wang, Y. X. Quadrotor trajectory tracking and obstacle avoidance by chaotic grey wolf optimization-based active disturbance rejection control," *Mechanical Systems and Signal Processing*. 2019, 128(1), 636-654.
3. Choi, A. J., Park, J., and Han, J.-H., Automated Aerial Docking System Using Onboard Vision-Based Deep Learning. *Journal of Aerospace Information Systems*. 2022, 19(6), 421-436.
4. Choi, A. J., Yang, H.-H., and Han, J.-H. Study on robust aerial docking mechanism with deep learning based drogue detection and docking. *Mechanical Systems and Signal Processing*. 2021, 154, 107579.
5. Shao, X., Liu, N., Wang, Z., Zhang, W., and Yang, W. Neuroadaptive integral robust control of visual quadrotor for tracking a moving object. *Mechanical Systems and Signal Processing*. 2020, 136, 106513.
6. Shim, D., Chung, H., Kim, H. J., and Sastry, S. Autonomous Exploration In Unknown Urban Environments For Unmanned Aerial Vehicles. *AIAA Guidance, Navigation, and Control Conference and Exhibit. American Institute of Aeronautics and Astronautics*, San Francisco, California, USA, 15-18 August 2005.
7. Qiu, Z., Zhao, N., Zhou, L., Wang, M., Yang, L., Fang, H., He, Y., and Liu, Y. Vision-Based Moving Obstacle Detection and Tracking in Paddy Field Using Improved Yolov3 and Deep SORT. *Sensors*. 2020, 20(15), 4082.
8. Mejias, L., McNamara, S., Lai, J., and Ford, J. Vision-based detection and tracking of aerial targets for UAV collision avoidance. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 18-22 October 2010, pp. 87-92.
9. Al-Kaff, A., Garcia, F., Martin, D., Escalera, A. D. L., Armingol, J. M. Obstacle Detection and Avoidance System Based on Monocular Camera and Size Expansion Algorithm for UAVs. *Sensors*. 2017, 17(5), 1061.
10. Alado, E., Gonzalez-deSantos, L. M., Michinel, H., and Gonzalez-Jorge, H. UAV Obstacle Avoidance Algorithm to Navigate in Dynamic Building Environment. *Drones*. 2022, 6(1), 16.
11. Ahmad, T., Cavazza, M., Matsuo, Y., and Prendinger, H. "Detection Human Actions in Drone Images Using YoloV5 and Stochastic Gradient Boosting," *Sensors*, Vol. 22, No. 18, article number 7020, 2022.
12. Lee, T.-J., Yi, D.-H., and Cho, D.-I. A Monocular Vision Sensor-Based Obstacle Detection Algorithm for Autonomous Robots. *Sensors*. 2016, 16(3), 311.
13. Uddin Haque, A., and Nejadpak, A. Obstacle Avoidance Using Stereo Camera, *arXiv*:1705.04114, 2017.
14. Falanga, D., Kim, S., Scaramuzza, D. How Fast Is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid. *IEEE Robotics and Automation Letters*. 2019, 4(2), 1884-1891.
15. Kwag, Y. K., and Chung, C. H. UAV based collision avoidance radar sensor. *IEEE International Geoscience and Remote Sensing Symposium*, Barcelona, Spain, 23-27 July 2007, pp. 639-642.
16. Moses, A., Rutherford, M. J., Kontitsis, M., and Valavanis, K. P. UAV-borne X-band radar for collision avoidance. *Robotica*. 2014, 32(1), pp. 97-114.
17. Lv, Y., Ai, Z., Chen, M., Gong, X., Wang, Y., and Lu, Z. High-Resolution Drone Detection Based on Background Difference and SAG-YOLOv5s. *Sensors*, 2022, 22(15), 5825.
18. Chen, T., and Lu, S. Object-Level Motion Detection From Moving Cameras. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017, 27(11), pp. 2333-2343.
19. Kim, J., Wang, X., Wang, H., Zhu, C., and Kim, D. Fast moving object detection with non-stationary background. *Multimedia Tools and Applications*. 2013, 67(1), pp. 311-335.
20. Seidaliyeva, U., Akhmetov, D., Ilipbayeva, L. and Matson, E. Real-Time and Accurate Drone Detection in a Video with a Static Background. *Sensors*, 2020, 20(14), 3856.
21. Fischler, M. A., and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM.* 1981, 24(6), pp. 381–395.
22. Massart, D. L., Kaufman, L., Rousseeuw, P. J., and Leroy, A. Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta*. 1986, 187, pp. 171-179.
23. Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.* AAAI Press, Portland, Oregon, 1996, pp. 226–231.

24. Lloyd, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory*. 1982, 28(2), pp. 129-137.
25. Yang, Y., Loquercio, A., Scaramuzza, D., and Soatto, S. Unsupervised Moving Object Detection via Contextual Information Separation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, Jun 15, 2019, pp. 879-888.
26. Falangs, D., Kleber, K., and Scaramuzza, D. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*. 2020, 5(40).