무인 비행체의 공중 충돌 방지를 위한 시각 기반 능동적 회피 기동 시스템

A Vision-Based Active Avoidance Maneuver System for Mid-Air Collision Avoidance of UAVs

초록

소형 무인 항공기의 보급과 맞물려 활발하게 개발되고 있는 자율 비행 시스템에서의 시급한 과제 중 하나는 공중 충돌의 예측 및 예방이다. 본 연구에서는 시각 데이터를 활용한 이동 물체 검출 및 추적을 통해 회피기동을 결정하는 시스템을 개발 및 시험하였다. 이 시스템은 카메라 이동에 따른 영상의 기하학적 구조 변화를 2차원 변환으로 근사하는 기법을 사용해 적은 연산 자원으로도 높은 저지연 성능을 보인다. 또한 시각 기반 시스템에 필연적으로 발생하는 잡음을 처리하기 위해 다양한 이미지 연산을 활용하고, 검출 및 추적 알고리즘에 검출 연속성 기반의 논리적 잡음 필터를 도입하여 오검출 및 탐지 실패의 가능성을 대폭 저감하였다. 이를 통합해 제한된 연산 자원과 추가적인 센서 장착의 어려움이라는 소형 무인 항공기의 제약 조건을 만족하는 공중 충돌 방지 시스템을 개발하였다.

Key Words : 광학 흐름, 특징점, 투영 변환, 군집화, 추적, 칼만 필터, 스테레오 깊이 추정

1. Introduction

The arrival of high-performance electronic components such as flight controllers and electronic speed controllers has led to mass-production and popularization of Unmanned Aerial Vehicles (UAVs). While fixed-wing and helicopter-like vehicles dominated in traditional UAVs, multi-rotor vehicles recently took the lead due to their low prices, good repairability, highly modular structure, and ease of operation. This was possible due to advancements in control engineering and motor technologies that enabled flight of vehicles with very simple mechanical structures. UAVs are now providing services in areas such as aerial photography, surveillance, site monitoring and delivery [1~3], and the market is expected to expand beyond 58 billion dollars by 2026 [4]. Various types of UAVs are shown in Fig. 1.

Lately, advancements in machine learning methodologies have enabled development of vision-based spatial and object recognition systems, and this led to an active research trend for autonomous flight systems. However, currently available autonomous flight systems of UAVs mostly focus on Global Positioning System (GPS)-based waypoint following, and the ability to navigate complex and dynamic environments is just under development [6, 7]. Drone manufacturers, namely DJI and Skydio, have integrated simple autonomous flight systems into their quadrotors (Fig. 2) that can follow subjects while avoiding static obstacles at low flight speeds. But they lack the capability to handle sudden changes in environments, and the standard function of static obstacle avoidance is still prone to failure.

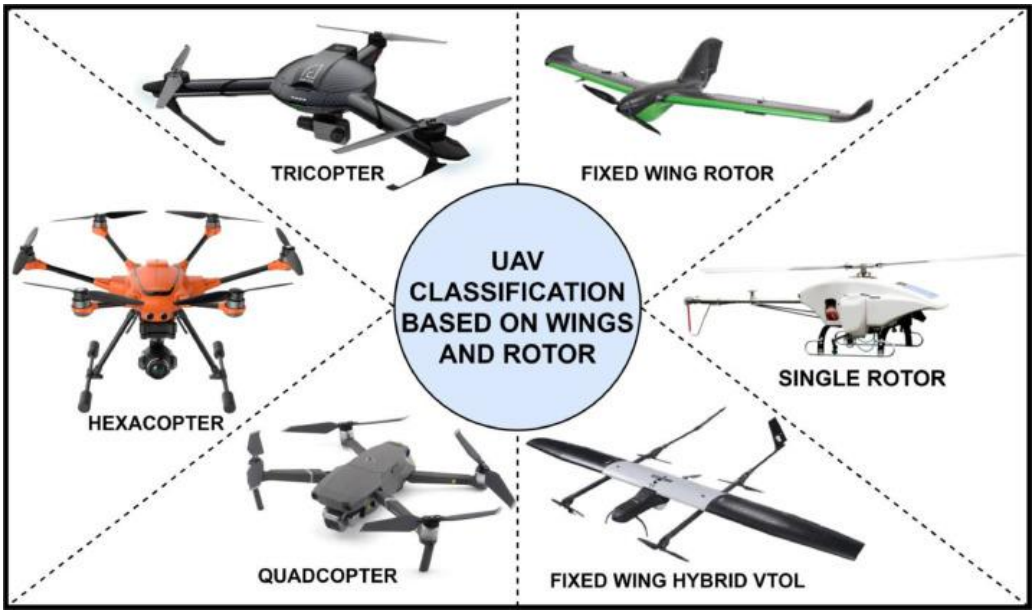


Fig. 1. Various body types of UAVs [5]

Fig. 2. Skydio(left) and DJI(right)'s autonomous quadrotors

One of the key abilities that autonomous flight systems for small UAVs must achieve is mid-air collision avoidance [8]. Unlike large aircrafts which generally operate at heights over a kilometer, small-scale UAVs which typically operate at altitudes of below a few hundred meters are exposed to various thread sources that can lead to mid-air collisions, such as birds (Fig. 3), other small aircrafts and ground-launched projectiles. Mid-air collisions are usually accompanied with vehicle damage, payload damage or loss and, at worst cases, an unrecoverable fall. In particular, multi-rotor UAVs which is the most common form of small aircrafts usually do not possess a way of gliding without power, and rotor damage or failure almost definitely leads to an unrecoverable crash. In this context, an autonomous collision avoidance system that can detect potential hostile objects and perform decision-making for avoidance maneuvers is regarded as a crucial component for autonomous flight systems [9, 10].

Fig. 3. A crashed multi−rotor UAV after a mid−air collision with an eagle.

Mid-air collision avoidance systems are critically different from static obstacle avoidance systems. Those systems mainly focus on mapping the environment and planning a best route without colliding with obstacles. However, mid-air collisions typically occur in very short time periods, and rapid cognition and response are the most crucial components of mid-air collision avoidance systems. Therefore, considerations about computational complexity must be taken into account when designing these systems in order to enable rapid and real-time operation. In addition, it is desirable to perform cognition with existing on-board sensors such as cameras rather than using additional sensors, to minimize impact on payload capacities.

2. Research Background and Objective

A. Collision avoidance systems – sensors

Collision avoidance systems require means of perceiving potential obstacles, and this is performed by one or more types of sensors [11]. Typical sensors employed for obstacle recognition include passive sensors such as cameras, and active sensors such as RADAR, LiDAR and SONAR.

Passive sensors, unlike active ones, do not emit electrical or sound waves and instead detect energy reflected from objects. A most commonly used type of passive sensor is a camera, which can further be classified into monocular, stereo and event-based cameras [12~14]. Cameras typically benefit from small size and ease of usage but are sensitive to lighting conditions. T.-J. Lee [15] employed inverse perspective mapping for object recognition, with the downside of relatively slow operational flight speeds. A. U. Haque [16] implemented a fuzzy controller with stereo cameras for obstacle avoidance. D. Falanga [17] demonstrated low-latency collision avoidance systems with bio-inspired sensors called event cameras. These sensors require no additional image processing operations, making it ideal for resource-constrained environments such as UAVs but suffer from very high prices.

Active sensors emit energy waves that reflects off object surfaces, and measure distances based on round-trip times of these waves. They are robust to lighting conditions and have relatively large operational range, but usually heavier and less easy to mount than passive sensors. Y.-K. Kwag [18] utilized RADAR sensors to obtain position and velocity information about nearby objects and execute appropriate maneuvers based on this information. A. Moses [19] developed a prototype of an X-band RADAR sensor for obstacle recognition from UAVs.

B. Collision avoidance systems – strategies

Collision avoidance strategies can be categorized into two groups of planning-based avoidance and reactive avoidance. Planning-based avoidance aims to build a precise map of the surrounding environment and plan an optimal route for collision avoidance. This strategy is usually employed during missions, and usually require large computational power. K. Bilimoria [20] proposed a system that can analytically plan a best route in 2-D environments. J.-B. Seo [21] devised methods to avoid interference with other UAVs in coordinated flights. Ha, Le N [22] proposed a method to calculate a threat score for detected objects, set safety boundaries based on this information, and calculate an optimal trajectory for avoiding all obstacles. Lin, Zijie [23] calculated safety levels based on perceived information and proposed a method to sequentially avoid obstacles.

Reactive avoidance aims for rapid perception and response to fast-changing environments and regard fast and successive avoidance as the highest priority. M. Wang [24] employed a 2-D LiDAR sensor for distinguishing stationary and mobile objects and estimate object speeds. S. U. Sharma [25] utilized machine learning to detect and avoid certain animals. M. C. Simone [26] utilized SONAR sensors and neural net models to estimate objects' location and perform evasive maneuvers. C. Goerzen [2] presented a simple obstacle perception and avoidance system using SONAR and IR sensors. Y. Yu [27] combined data from a SONAR sensor and a stereo camera for better object detection probabilities.

C. Moving object detection with camera sensors

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 consume little computational resource as possible. 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. K.-M. Yi [28] modeled background objects using a dual-mode single gaussian model to categorize background and foreground objects. This method is capable of running on a mobile platform. S. Lu [29] compared motion vectors of moving and stationary objects to detect moving objects. J.-M. Kim [30] clustered optical flow vectors with K-nearest neighbors clustering and distinguished moving objects based on cluster variance. Y. Yang [31] trained a Generative Adversarial Network (GAN) to distinguish foreground and background pixels. This network is capable of filtering out stationary, background pixels, leaving only pixels of moving objects.

D. Research objective

This research aims to develop and test a perception, prediction, and decision-making system for

reactive mid-air collision avoidance. The system utilizes a stereo 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. This algorithm approximates background motion due to camera movement with 2-D perspective transformation and utilizes background subtraction to extract regions where a moving object is present. To cope with inevitable noise from the approximation process and the visual data itself, various image filters and binarization is utilized. Custom-made clustering and tracking modules perceive and track individual objects for threat assessment. The distance to an object measured by the stereo camera is used as a criterion for decision making. Avoidance maneuvers are performed if the distance to an object falls below predetermined safety thresholds.

To test the system, a quadrotor UAV equipped with a Raspberry Pi 4 low-power computer and a Intel RealSense D435i stereo camera is used as a testing platform. Evasive maneuvers from various conditions are tested, and the results are used to further optimize the operational parameters to better suit flight environments.

3. Vision-Based Moving Object Detection Algorithm

A. Relationship between the 3-D world and 2-D image

A digital camera captures the 3-D world onto a 2-D image by projecting the light from its lens to an image sensor. The sensor quantizes this light to individual brightnesses of red, green, and blue colors (Fig. 4).

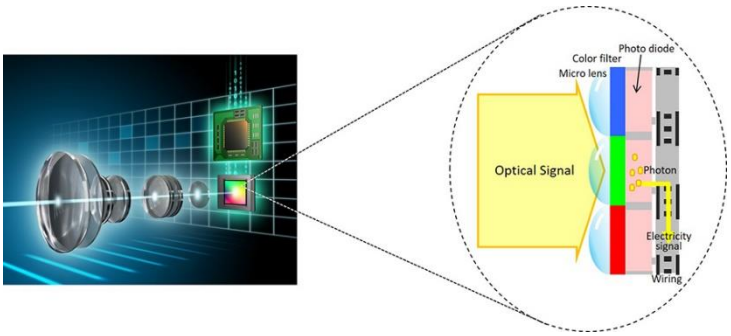


Fig. 4. Image formation of a digital camera

To describe a 3-D world point being projected onto a 2-D image plane, the pinhole camera model is widely used. A light ray from a 3-D world point passes through a “pinhole” with no size and reaches the image plane, creating a one-to-one correspondence between 3-D and 2-D points. As shown in Fig. 5, the 2-D location of a projected 3-D point depends on its 3-D coordinates and the focal length f. In the case when the image plane is behind the center of projection, the 3-D world appears “flipped”. To simplify the understanding, one can instead think of a pinhole camera model with the image plane in front of the center of projection.

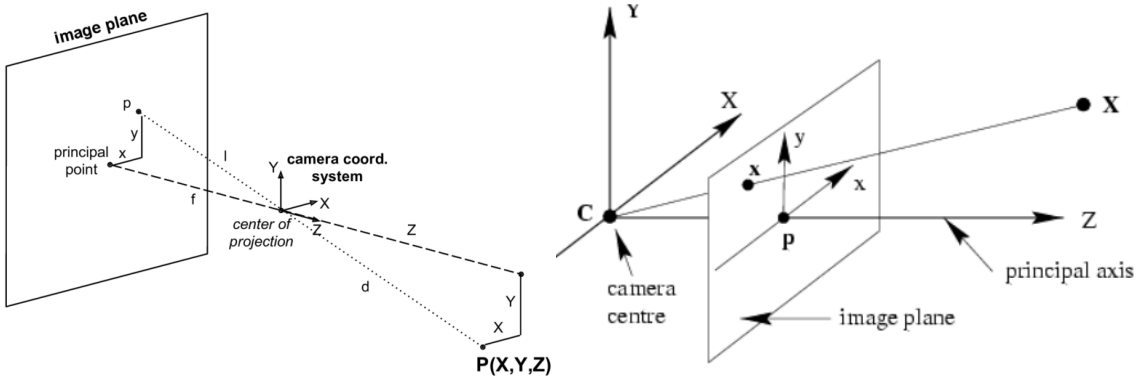


Fig. 5. The standard pinhole camera model(left), and the pinhole camera model with the image plane in front of the center of projection(right)

Thus, according to the pinhole camera model, the relationship between a 3-D point P and 2-D point p is as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1}$$

$$where \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = p \; and \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = P$$

The image plane of a digital camera is the image sensor. This sensor is usually rectangular and consists of square pixels. As shown in Fig.6, the pinhole camera model sets the origin of a 2-D image at its center, while in pixel coordinates, the upper left corner is set as the origin.

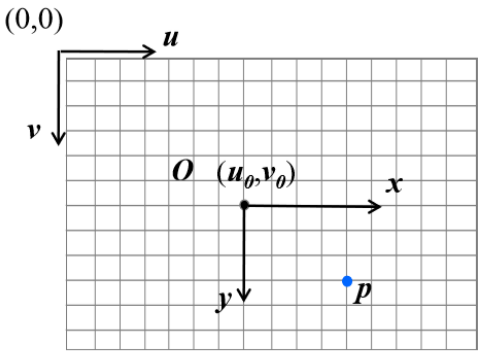


Fig. 6. The 2-D image plane coordinates$(x, y)$and pixel coordinates(u, v).

Thus, the relationship between these two coordinates can be expressed as follows:

$$u = u_0 + m_x \cdot x, v = v_0 + m_y \cdot y \tag{3.2}$$

$$where\ m_x\ and\ m_y\ are\ the\ pixel\ conversion\ factors.$$

In the case when the camera is in motion, it is often convenient to define a reference coordinate system that can express 3-D world points and the pose of the camera (Fig. 7). This system is often named as the world coordinate system, and its origin can be set arbitrarily.

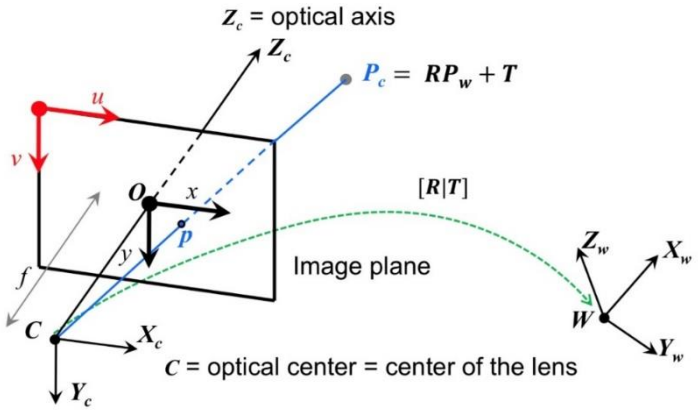


Fig. 7. A 3−D point and the camera, from the world coordinate system.

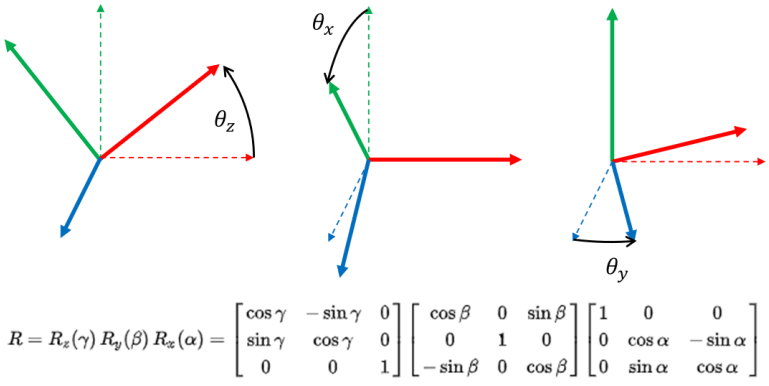


Fig. 8. Compositions of a standard rotation matrix

Transformation from the world coordinate system to the camera coordinate system requires the camera pose to be expressed as a translation vector and a rotation matrix. Fig. 8 illustrates the composition of a standard rotation matrix.

Fusion of these formulas yield the following relationship between a 3-D world point and a corresponding 2-D pixel.

$$x = K[R|-RC] \cdot X \tag{3.3}$$

$$where\ K = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} is\ the\ calibration\ matrix,$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} is\ the\ rotation\ matrix, and$$

$$C = \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} is\ the\ translation\ vector.$$

$$x\ and\ X\ are\ in\ the\ pixel\ and\ world\ coordinate\ systems, respectively.$$

B. Epipolar geometry

Epipolar geometry is a methodology of describing the relation between two images of a same scene taken at different locations. It is utilized in applications such as Structure from Motion (SfM) (Fig. 9).

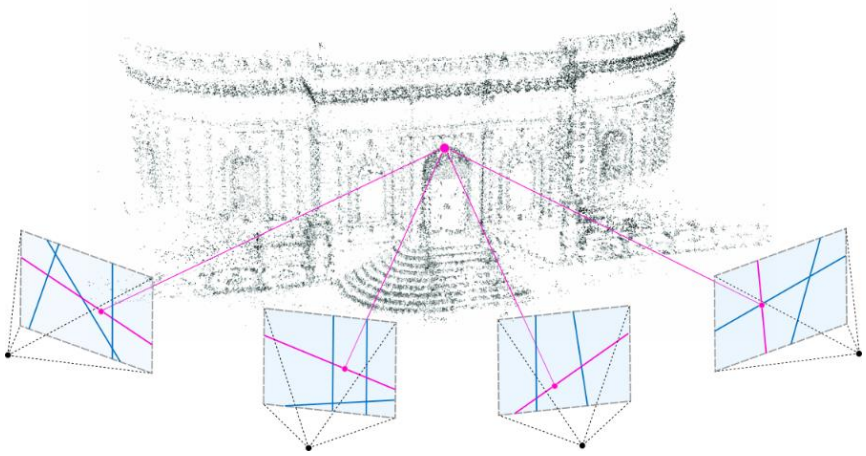


Fig. 9. Structure from Motion(SfM)

Let a 3-D point P be observed from different points of view A and B, as in Fig. 10. P appears as point p on image A, and as point p' on image B. The points e and e' on image A and B that intersects with the line that connects the two centers of projections are called epipoles. The lines that connects e to p and e' to p' are called epilines, and they are uniquely determined for a 3-D point. Epilines for multiple points intersect at a unique point (Fig. 11).

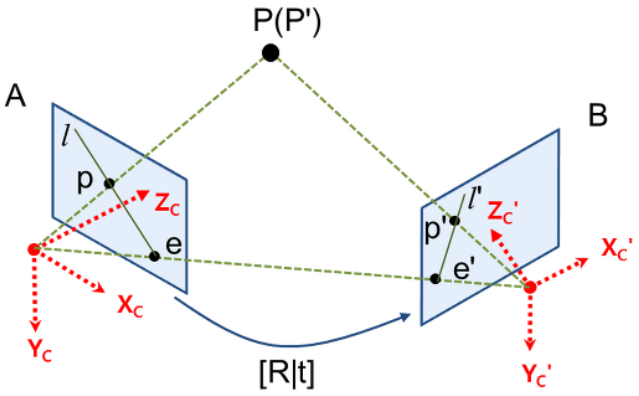


Fig. 10. Epipolar geometry

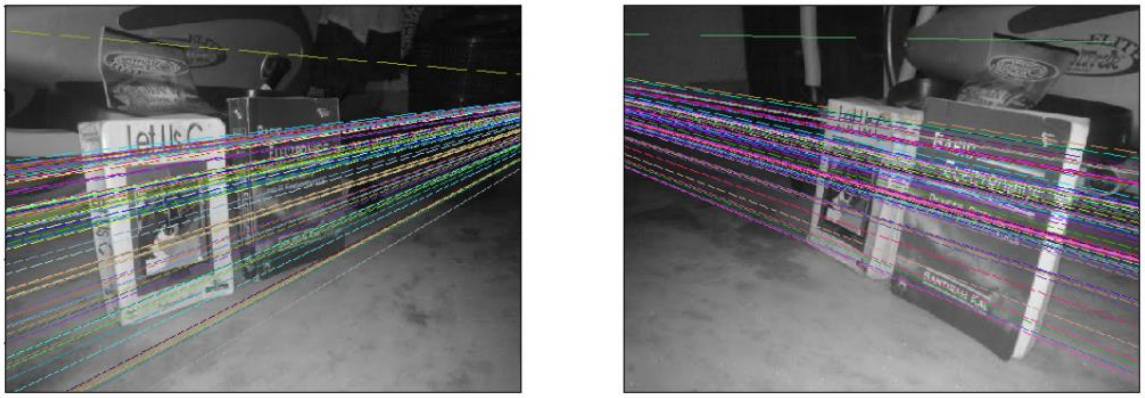


Fig. 11. All epilines for an image intersects at a unique point.

Epipolar geometry can be utilized to geometrically describe camera motion between consecutive video frames. Considering that a video from a moving camera is a set of pictures that were taken at slightly different locations, epipolar geometry can be applied between these frames. Fig. 12 is an example of epilines drawn on video frames taken while the camera is undergoing motion.

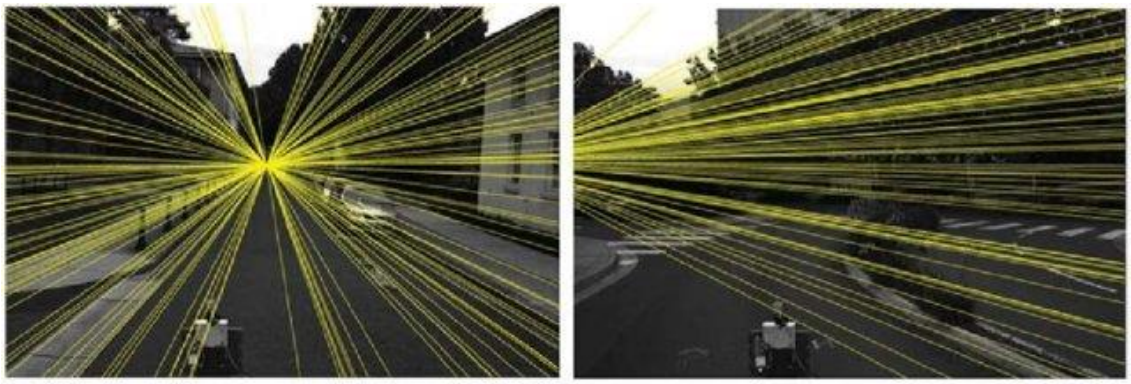


Fig. 12. Epilines for video frames taken while camera translation(left) and rotation(right) (32).

Image points on video frames move along epilines as the camera undergoes motion. That is, the scene transition due to camera motion can be described with epipolar geometry. Detailed methodology is provided in section 3-F.

C. 2−D transformations

Transformation in image processing refers to a function that maps an arbitrary point to another point. The 2-D transformation is a specific example of this, which performs 2-D to 2-D mapping. The most general form of 2-D transformation is the projective transformation (Fig. 12), which can be interpreted as a transformation that can map an arbitrary quadrangle into another quadrangle. Thus, projective transformation can describe the relationship between the two images that observe the same planar object.

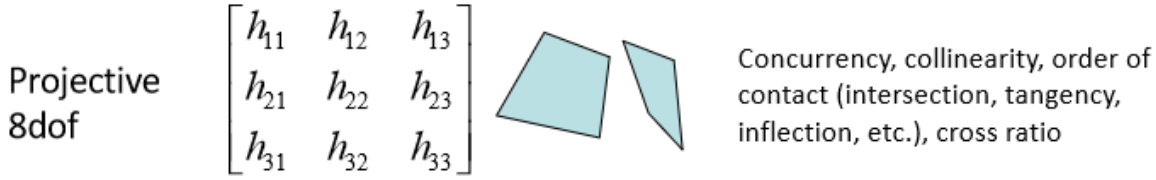


Fig. 12. The projective transformation maps an arbitrary quadrangle into another quadrangle.

The matrix that performs projective transformation is called the homography matrix, and the transformation can be defined as follows:

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.4}$$

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

A homography matrix has 8 degrees of freedom, and at least 4 matches between images are required for its computation. Generally, more than 4 matches are processed with outlier rejection techniques such as RANSAC [33] or LMedS [34] to yield optimal results.

In principle, projective transform can only describe planar, or 2-D objects. However, as in Fig. 13, if 3-D objects are relatively far away from the camera and the pose difference between viewpoints are small, the relationship between the images taken from these viewpoints can be approximated with projective transformation. This is a core assumption of the moving object detection system, and the details on how to utilize this assumption for background motion modeling is provided in section 3-F.

Fig. 13. Approximation and transformation of background change between two viewpoints with projective transformation

D. Feature point tracking with optical flow

Optical flow is a methodology of estimating motion vectors of pixels during two consecutive frames. It was first suggested by James Gibson during the 1940's and was thoroughly utilized for applications such as object tracking [35] and robot localization [36]. Optical flow involves calculation of a differential equation called the gradient constraint equation, and methodologies for solving this equation separates various optical flow algorithms. Some of the popular methods are Lucas-Kanade [37] and Horn-Schunck [38] algorithms. Lucas-Kanade algorithm only tracks specified pixels but computationally less complex and accurate. Horn-Schunck algorithm tracks all pixels in a frame but with less accuracy and speed. Fig. 14 illustrates the two algorithms in operation.

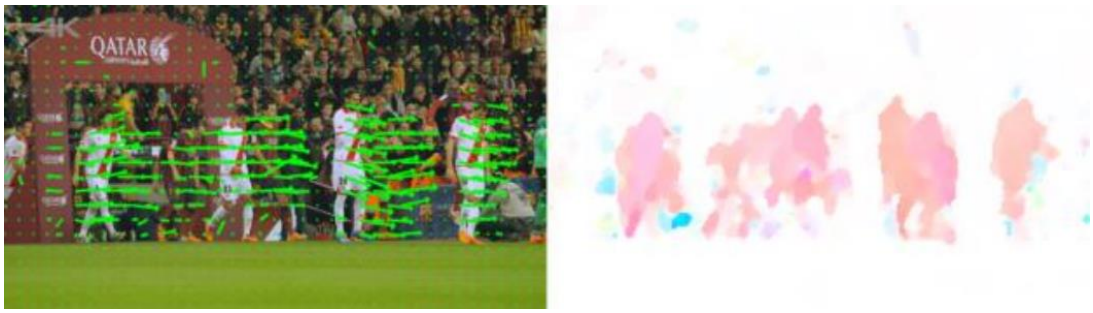


Fig. 14. Lucas-Kanade(left) and Horn-Schunck(right) algorithms in operation

Combination of epipolar geometry, projective transform and optical flow enables background motion modeling and moving object detection. The detailed procedures are suggested in section 3-F.

E. Reducing noise components with image filters

Image filtering is an image processing technique that utilizes specifically designed filters to perform convolution operations on images and replaces pixels with the results (Fig. 15). Various filters exist that performs image processing such as smoothing [39] and edge detection [40].

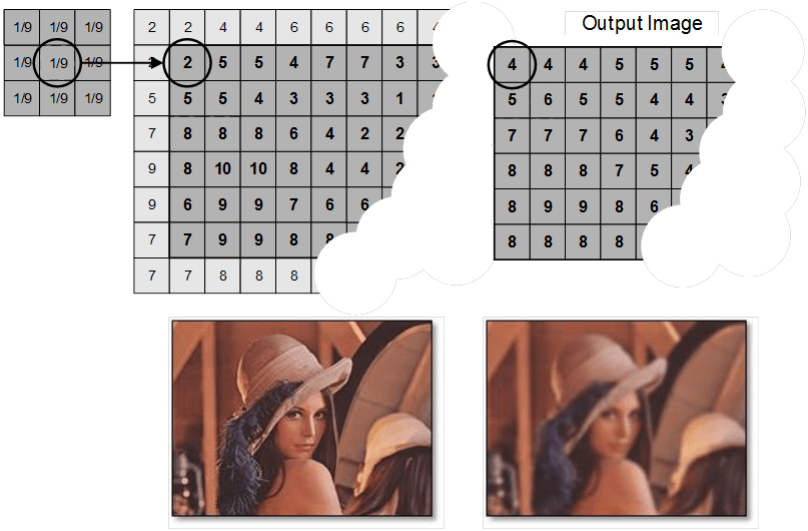
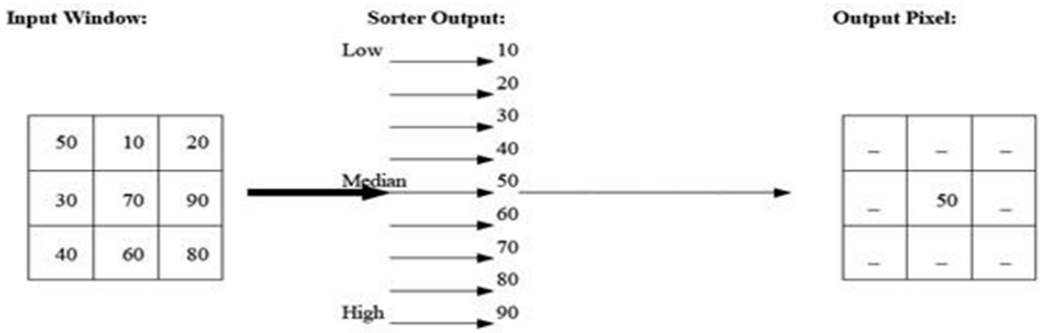


Fig. 15. Image filtering

The proposed system utilizes median filters for denoising operations during moving object detection. As in Fig. 16, median filter sorts pixels in its window in brightness order and replaces the center pixel brightness with the median value, making it ideal for eliminating small noise components.



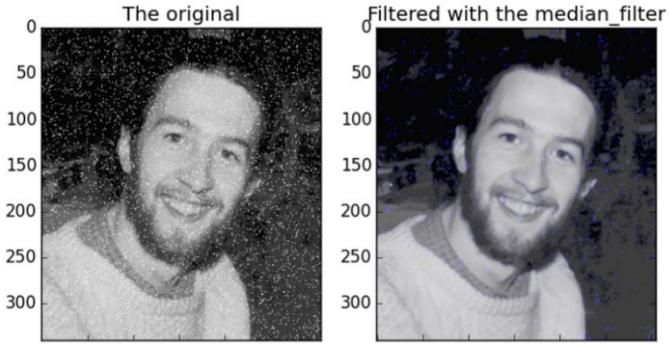


Fig. 16. Denoising operation with median filter

In addition, the dilation filter is applied to better reveal moving objects. As in Fig. 17, the dilation filter expands small and thin image components to better visualize them.

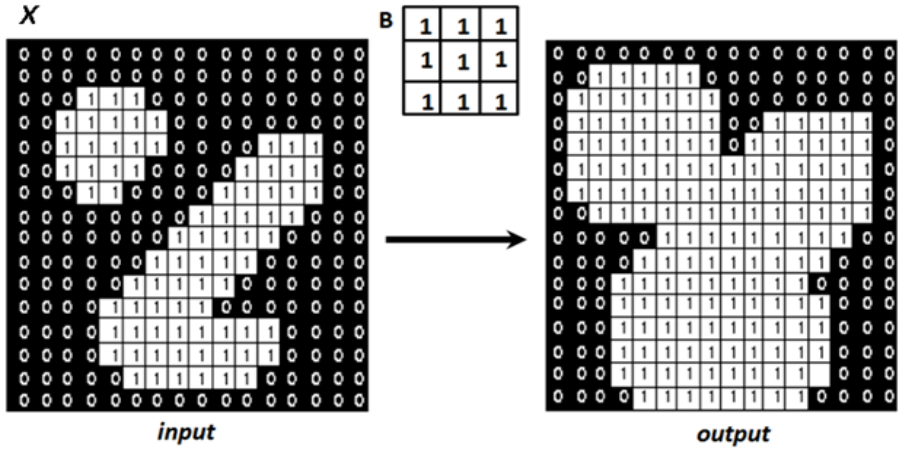


Fig. 17. Application of the dilation filter on a binary image

F. Moving object detection from a mobile camera

The mid-air collision avoidance system that this paper proposes 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: $I_{t-1}, I_t, I_{t+1}, H_{t-1,t}$ ▶Three consecutive image frames and the homography matrix between previous and current frame

Output: R_t ▶Binary image that displays the region where moving objects are present

1: $\boldsymbol{FP} \leftarrow [0\ 0\ 0\ 0\ 0\ 0]$ ▶Number of feature points for edge areas, where optical flow is calculated
2: $i = 0$
3: ***while*** $i < 6$:
4: $\boldsymbol{FP}[i] \leftarrow number\ of\ feature\ points\ in\ \boldsymbol{I_t}[i]$ ▶ Number of feature points for an edge area
5: $if\ \boldsymbol{FP}[i] < n$: ▶Threshold to search for new feature points
6: $Find\ at\ least\ n - \boldsymbol{FP}[i]\ feature\ points\ in\ \boldsymbol{I_t}[i]$
7: ***end if***
8: ***end while***
9: $\boldsymbol{OF} \leftarrow optical\ flow\ between\ \boldsymbol{I_t}[i]\ and\ \boldsymbol{I_{t+1}}[i]$ ▶Calculate optical flow
10: $\boldsymbol{H_{t,t+1}} \leftarrow estimate\ homography\ between\ \boldsymbol{I_t}[i]\ and\ \boldsymbol{I_{t+1}}[i]\ using\ \boldsymbol{OF}$ ▶Calculate homography matrix
11: $\boldsymbol{R_t} \leftarrow average([(\boldsymbol{I_t} - \boldsymbol{H_{t-1,t}} \cdot \boldsymbol{I_{t-1}}) + (\boldsymbol{I_{t+1}} - \boldsymbol{H_{t,t+1}} \cdot \boldsymbol{I_t})])$ ▶Background subtraction
12: $\boldsymbol{R_t} \leftarrow Binarize(Dilation(MedianFilter(\boldsymbol{R_t})))$ ▶Apply denoising filters and binarization

The purpose of homography matrix calculation between consecutive images is to model the background scene transition caused by camera motion. As mentioned in section 2-C, if 3-D objects are at a relatively far distance and the pose difference between viewpoints are small, the transition between two video frames can be approximated as a projective transformation. Fig. 18 visualizes this approximation process.

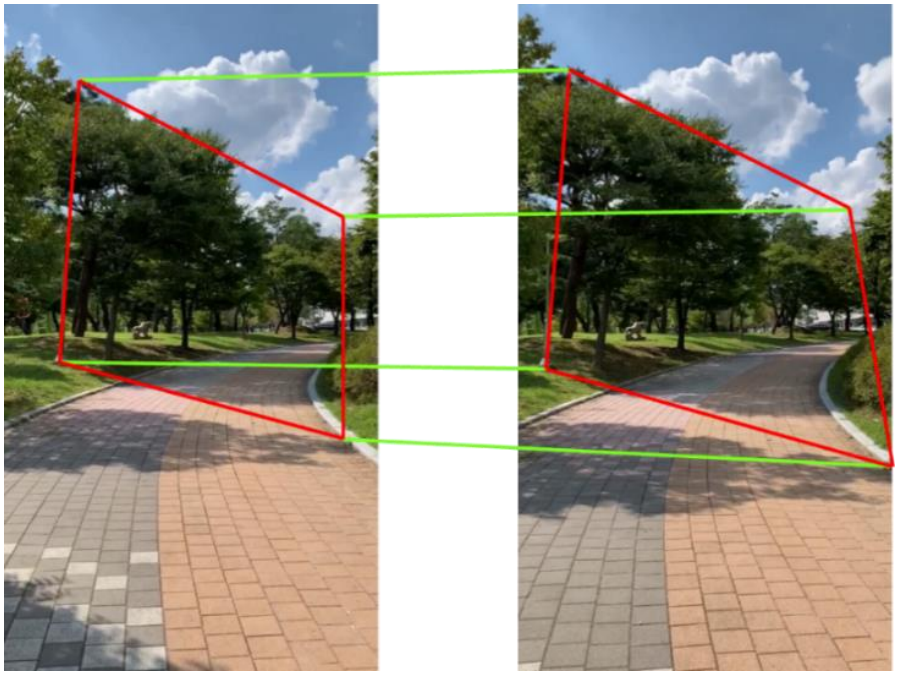


Fig. 18. Projective transformation approximation of two consecutive video frames

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. Considering equation (3.1) from section 3-A, further objects appear smaller and thus exhibit lesser magnitudes of appearance transition. Thus, as in Fig. 19, objects near the edge of the field of view have greater influence on the perspective transformation model and the homography matrix.

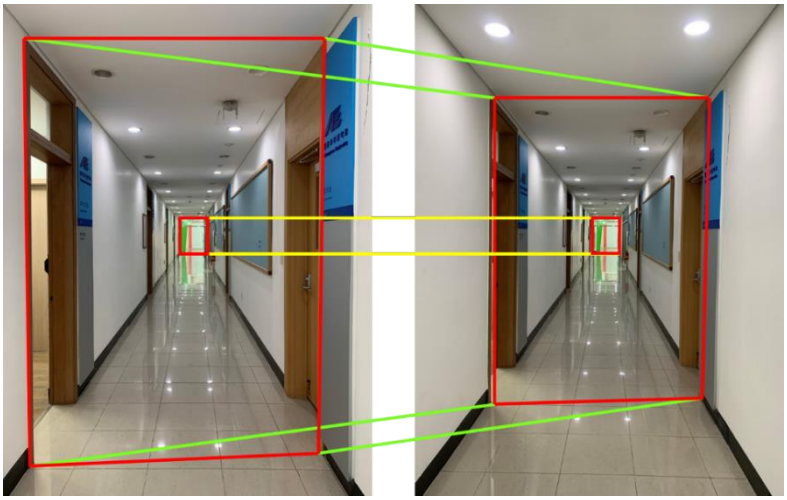


Fig. 19. Appearance transition comparison of center and edge objects

Therefore, it is reasonable to calculate the homography matrix from the transition of pixels that are near the edge of the field of view. In this system, six 70x70 windows were used (Fig. 20). The system constantly monitors number of tracked points in these windows and initiates new tracks if the number falls below a predetermined threshold. This guarantees even distribution of tracked points and guarantees the stability of homography matrix calculation. To filter out false matches and outliers, RANSAC [36] was used for homography matrix calculation. Limiting the optical flow calculation regions reduced computational requirements and improved approximation accuracy simultaneously.

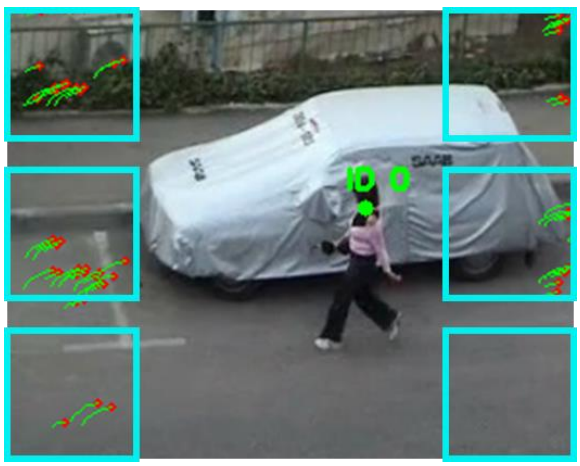
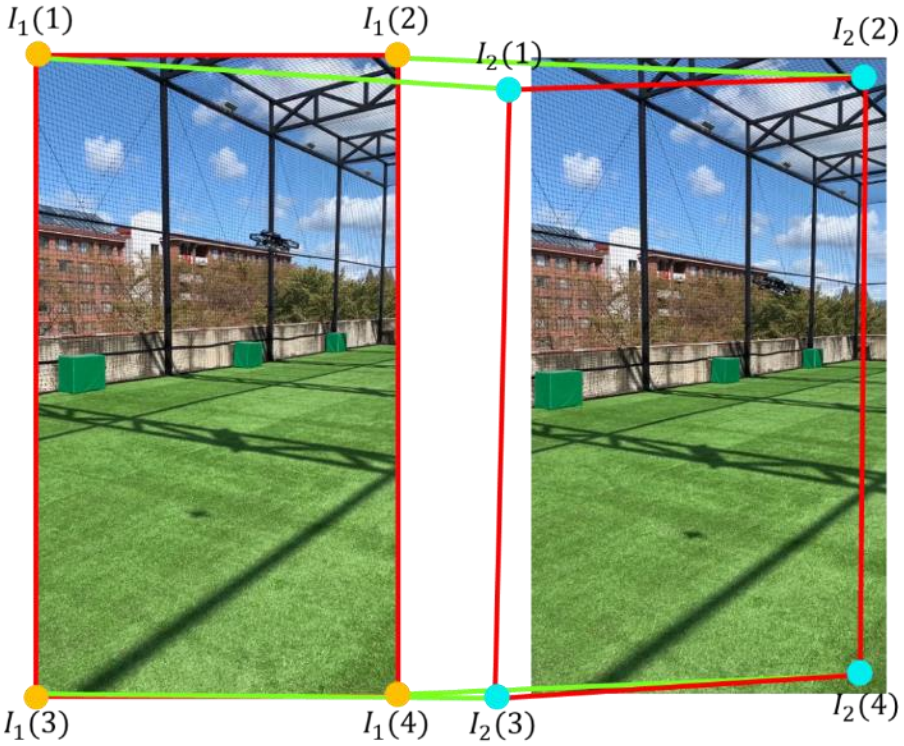


Fig. 20. Six boundary regions where optical flow is calculated

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 (Fig. 21). However, motions of moving objects are not described by the homography matrix and thus cannot be mapped to coincide between frames. Therefore, moving objects can be detected by searching for regions where pixels are not correctly mapped between consecutive frames.



$$[I_1(1) \quad I_1(2) \quad I_1(3) \quad I_1(4)] = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot [I_2(1) \quad I_2(2) \quad I_2(3) \quad I_2(4)]$$

Fig. 21. Mapping of pixels between consecutive video frames

To locate incorrectly mapped pixels and detect moving objects, perspective transform based background subtraction technique is employed. First, the homography matrix between two consecutive frames is computed. Next, this matrix is used to map pixels of the previous frame to match the current frame's. Then, the transformed frame is subtracted from the current frame. Since the background scene and objects are approximately mapped to match the current frame, their brightness approximately becomes zero after the subtraction. However, the pixels of moving objects are not correctly mapped to the current frame since its motion is different from the background's. That is, their brightness value is nonzero after the subtraction, signifying that image regions of moving objects are detected. An example result is shown in Fig. 22.

Fig. 22. Original video frame(left) and background subtracted result(right)

However, since 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, the proposed system utilizes two methods.

The first method utilizes the next frame as well as the previous frame for background subtraction. Each transformed frame is subtracted from the current frame independently, and then averaged (Fig. 23). Since noise elements appear randomly and momentarily, this method improves the Signal-to-Noise Ratio (SNR) of the result image.

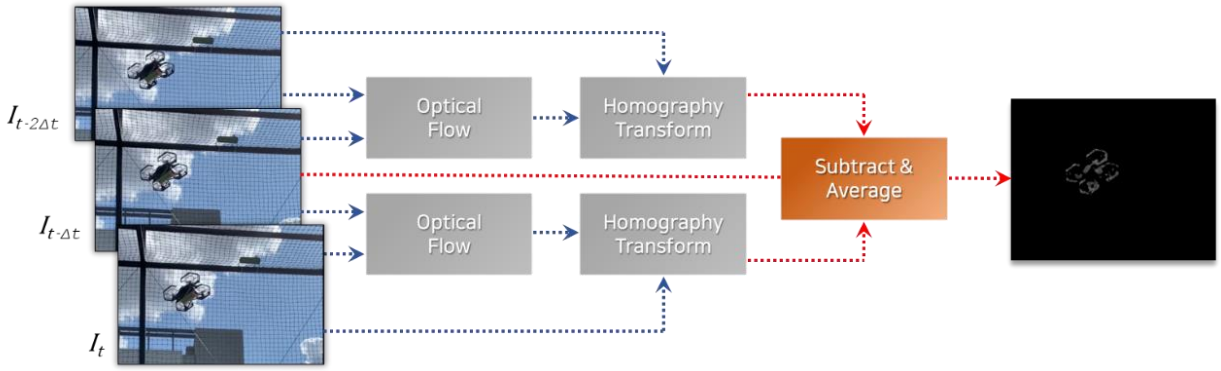


Fig. 23. Utilizing previous and next frames to improve SNR

The second method applies image filters and binarization to the result image after the first method. As mentioned in section 3-E, the median filter eliminates small noise components while the dilation filter expands elements, thus revealing moving object regions more clearly. Then the image is binarized to ignore pixels below certain brightness threshold. The process is illustrated in Fig. 24.

After these operations, image regions of moving objects are revealed. However, further processing is necessary to recognize the “blobs” as shown in Fig. 25 as a single, independent object. Furthermore, additional noise filtering is required to eliminate false positive detections. These procedures are presented in section 4.

Fig. 24. (From upper left, clockwise) The original frame / background subtracted frame / median filtered frame / dilation filtered and binarized frame

Fig. 25. A single moving object shows up as “blobs” after these operations.

4. Independent Object Recognition, Tracking, and Decision Making

A. Independent object recognition

The mid-air collision avoidance system that this paper proposes consists of four operational stages: [Detection], [Recognition], [Tracking], and [Decision making]. The [Recognition] stage utilizes 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: $\boldsymbol{R_t}$ ▶A binary image that contains object regions
Output: $\boldsymbol{C_t}$ ▶A list of prior object coordinates

$\boldsymbol{C_t} \leftarrow \emptyset$
for *unallocatedBlob* ***in*** $\boldsymbol{R_t}$: ▶A region in $\boldsymbol{R_t}$
  if $size(unallocatedBlob) < sizeThresh$:
    continue ▶Disregard small regions
  for *cluster* ***in*** *Clusters*:
    for *blob* ***in*** *cluster*:
      if $distance(blob, unallocatedBlob) < distThresh$:
        allocate unallocatedBlob to cluster
        break ▶Add region to cluster if below distance threshold
  if *unallocatedBlob is unallocated*:
    allocate unallocatedBlob to Clusters and set as cluster ▶Set as new cluster
for *cluster* ***in*** *Clusters*:
  append weighed mean position of cluster to $\boldsymbol{C_t}$ ▶Append representative position of blob to $\boldsymbol{C_t}$

The [Tracking] phase outputs a binary image that displays regions of moving objects as 1. However, the presence of noise is inevitable – some background regions show up as value 1, albeit intermittently. Additionally, a single independent moving object can appear in several "patches", as in Fig. 25. Therefore, there is need for a method that can reject noise components as well as recognize multiple nearby patches as a single object.

For the proposed system, a modified DBSCAN (Density-Based Spatial Clustering of Applications with Noise [41]) clustering algorithm is developed. DBSCAN, unlike center-based algorithms such as K-Means [42], 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. Fig. 25 is a comparison of K-means and DBSCAN.

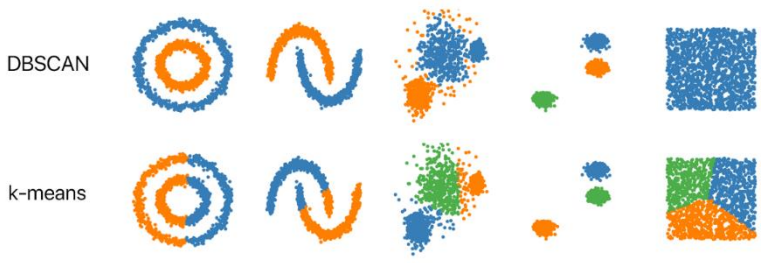


Fig. 25. A comparison of K−means and DBSCAN

DBSCAN is used to cluster the “patches” that constitutes the true moving object, in order to recognize them as a single object. However, vanilla DBSCAN cannot be directly applied for this task, since there are instances where an object shows up as a single patch, and where noise components show up as several patches (Fig. 26).

Fig. 26. Some edge cases that need to be considered

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 a weighed sum of all patches in that cluster. This procedure is summarized in Fig. 27.

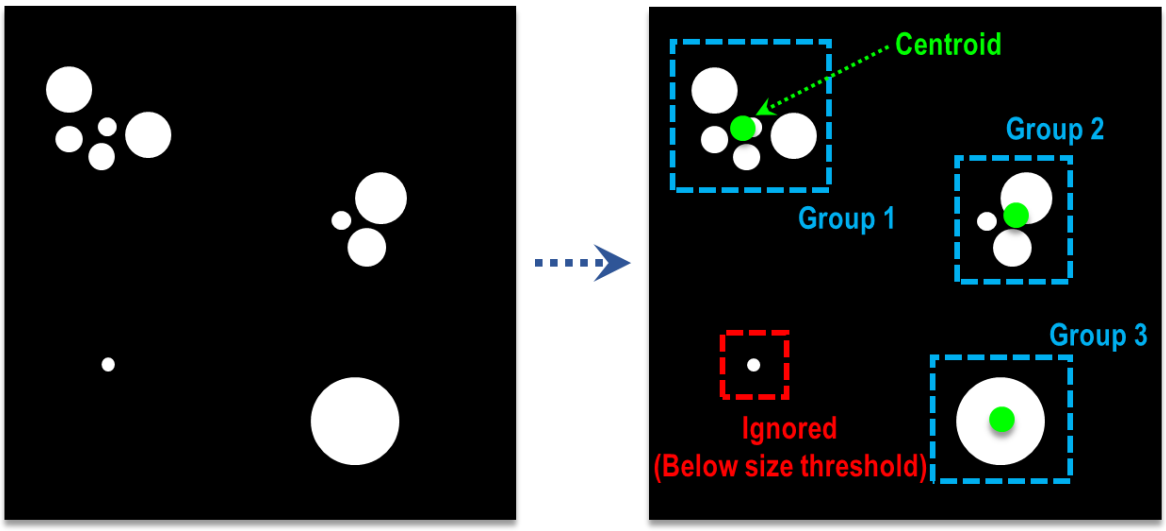


Fig. 27. Clustering procedure of the modified DBSCAN algorithm

B. Independent object tracking

The mid-air collision avoidance system that this paper proposes consists of four operational stages: [Detection], [Recognition], [Tracking], and [Decision making]. The [Tracking] stage receives the objects' locations at each frame from the [Recognition] stage and associates them to the previous frame's. A unique id is assigned to each independent objects, and their trajectories are monitored for threat assessment. This stage also performs further noise compensation. A pseudocode of this stage is given in algorithm 3.

Algorithm 3 Independent object tracking

Input: $\boldsymbol{O_{t-1}, C_t}$ ▶O_{t-1}: object locations and IDs at previous frame

▶C_t: object locations at current frame

Output: $\boldsymbol{O_t}$ ▶O_t: object locations and IDs at current frame

$\boldsymbol{O_t} \leftarrow \emptyset$

$M = size(O_{t-1})$ ▶number of objects at previous frame

$N = size(C_t)$ ▶number of objects at current frame

$Array \leftarrow zeros(M, N)$ ▶An array that stores distances between all objects

for i **in** O_{t-1}:

  for j **in** C_t:

    $Array[i, j] = distance(O_{t-1}[i], C_t[j])$

for k **in** C_t:

  if $ID[k]$ **is not** set:

    if $Array[i, k] ==$ min $(Array[i, :])$: ▶object k: closest object to object i

      if $Array[i, k] < distThresh$: ▶if below distance threshold:

        $ID[k] \leftarrow ID[i]$ ▶assume k and i is the $same$ $object$(tracking success)

        $Tracked\ length\ of\ ID[k]\ +\!= 1$ ▶Increase tracked length

        break

      else:

        $mark\ ID[i]\ as\ lost$ ▶Mark as lost if no object is nearby

for k **in** C_t:

  if $ID\ of\ k$ **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:	***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* $\boldsymbol{O_t}$	

The [Tracking] stage exploits temporal information to distinguish true observations from noise, and to recognize object identity across multiple frames. It operates on a central assumption that the distance that an object travelled between frames is smaller than distances between separate objects. Also, since noise components spawn intermittently and momentarily, they do not appear consistently across multiple frames.

Under these assumptions, the tracking procedure is formulated as follows. First, the distances between previously detected coordinates and the new coordinates are compared, and 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 longer than a time threshold, it is approved as a true object and given a new ID. This procedure is summarized in Fig. 28.

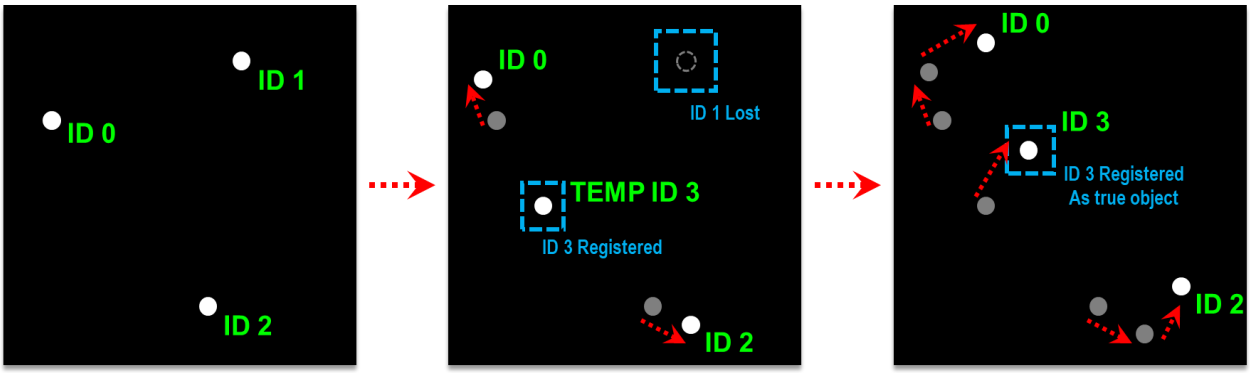


Fig. 28. Tracking procedure of the distance-based tracking algorithm

C. Trajectory data refinement

As in section 4-A, the objects' representative position is calculated as a weighed mean of cluster components. However, this data is inherently indeterministic, and this leads to a noisy trajectory (Fig. 29) that hinders the operation of the tracking algorithm and the stereo distance estimation.

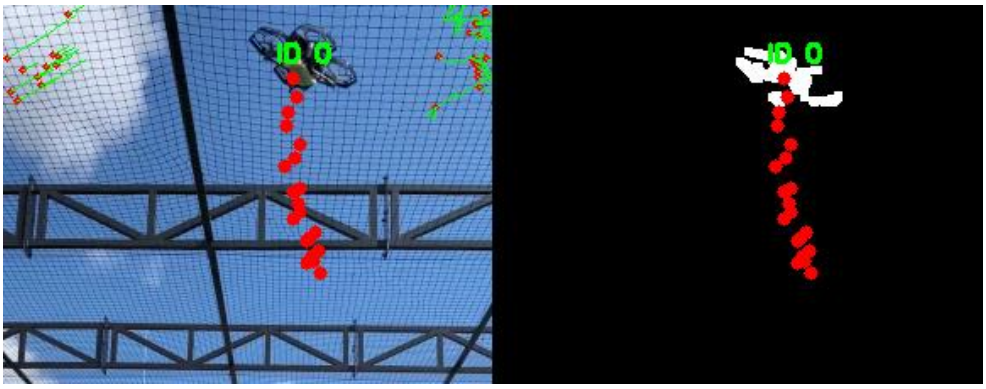


Fig. 29. Noisy trajectory data

This shortcoming is addressed with the Kalman filter. The Kalman filter recursively estimates the state of a linear system under noisy observations, providing an optimal statistical prediction of the system. It was suggested by Rudolf Kalman in the 1960's and was exploited extensively in areas such as object tracking, localization and state estimation [43]. Its operating procedure is illustrated in Fig. 30.

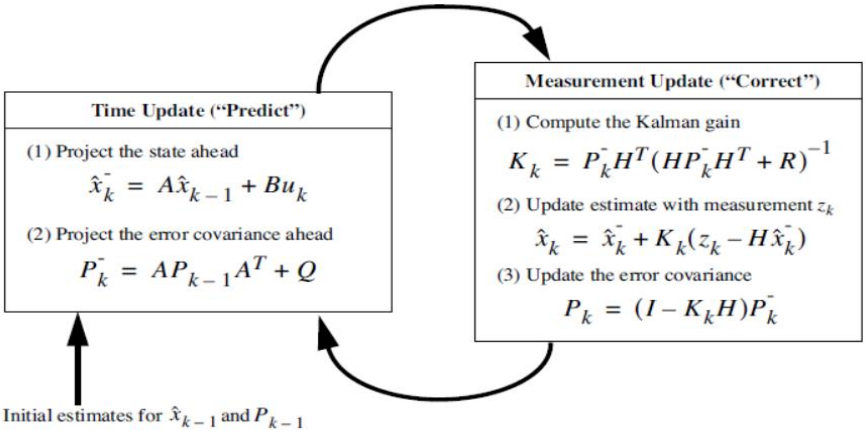


Fig. 30. The recursive operation of the Kalman filter

A Kalman filter is linear or nonlinear depending on the motion and observation models. Linear Kalman filters require less computational load at the cost of accuracy, and this filter is exploited for the proposed system, since it needs the Kalman filter for simple noise reduction and not precise state estimation. A constant-velocity (CV) model is utilized since the motion of an unknown object is unpredictable. The observed state is the x and y pixel coordinates of the object. The equations for these models are given below. Fig. 31 illustrates the tracking data before and after the Kalman filter application.

$$x_k = Ax_{k-1} + w_k = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{k-1} + w_k \tag{4.1}$$

$$z_k = Hx_k + v_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x_k + v_k \tag{4.2}$$

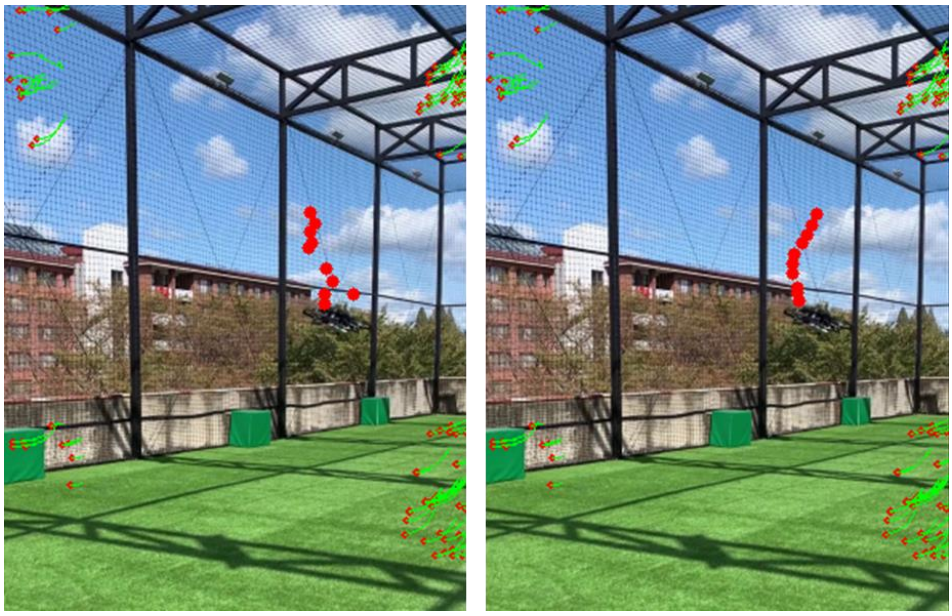


Fig. 31. Before(left) and after(right) Kalman filter application. Trajectory is displayed in red dots.

D. Decision making for avoidance maneuvers

The mid-air collision avoidance system that this paper proposes consists of four operational stages: [Detection], [Recognition], [Tracking], and [Decision making]. The [Decision making] stage receives the trajectory data from the [Tracking] stage and decides whether if avoidance maneuver must be performed, and if so, in what direction it should be. A pseudocode of this stage is given in algorithm 4, and Fig. 32 is a diagram of this algorithm.

Algorithm 4 Decision making for avoidance maneuvers

Input: $\boldsymbol{O_t}$ ▶O_t: object IDs and locations at current frame

Output: $\boldsymbol{A_t}$ ▶A_t: avoidance maneuver direction

1: $\boldsymbol{for}$ $object$ $\boldsymbol{in}$ $\boldsymbol{O_t}$:

2: $position \leftarrow 3D\ position\ of\ object$ ▶acquire 3D position of object

3: $\boldsymbol{if}$ $position.z < emergencyThresh$:

```
    emergency stop before proceeding
if position is inside SafeWindow:
    mark object as hostile                    ▶consider object as a threat if inside SafeWindow
    if position.z < avoidThresh:              ▶execute avoidance maneuver if distance is below threshold
        if position.xy in upperLeft:
            move downward right at v m/s until outside of SafeWindow
        if position.xy in upperRight:
            move downward left at v m/s until outside of SafeWindow
        if position.xy in downwardLeft:
            move upward right at v m/s until outside of SafeWindow
        if position.xy in downwardLeft:
            move upper right at v m/s until outside of SafeWindow
```

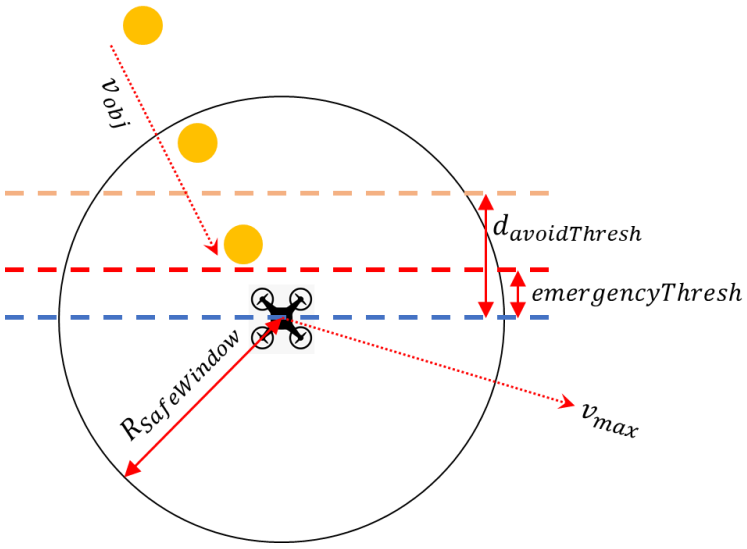


Fig. 32. A diagram of algorithm 4

It is desirable to use depth information for threat assessment and decision making. However, distance measures from stereo cameras have limited range since the disparity from two lenses are inversely proportional to distance. For example, the RealSense D435i stereo camera used for the experiment has an effective range of 10 meters. Therefore, the proposed system trusts depth information only if the object approaches within this range and performs avoidance maneuvers if the 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 is commanded to avoid crashing into the obstacle. Radius of the safe window and the avoidance threshold can be determined from prior information such as objects'

expected maximum velocity and the maximum possible maneuver velocity. A simple relation between the variables is as follows.

$$\frac{d_{avoidThresh}}{v_{obj}} = \frac{R_{SafeWindow}}{v_{max}} \tag{4.3}$$

5. System Integration and Test Results

A. System integration

The testing unit for the proposed system is a PX4 autopilot-based quadrotor UAV (Fig. 33). It is equipped with a Raspberry Pi 4 companion computer, a RealSense D435i stereo depth camera, and a RealSense T265 visual odometry camera for autonomous flight purposes. The system components are interconnected vis ROS, a robotics development software. Fig. 34 is a diagram of the system.

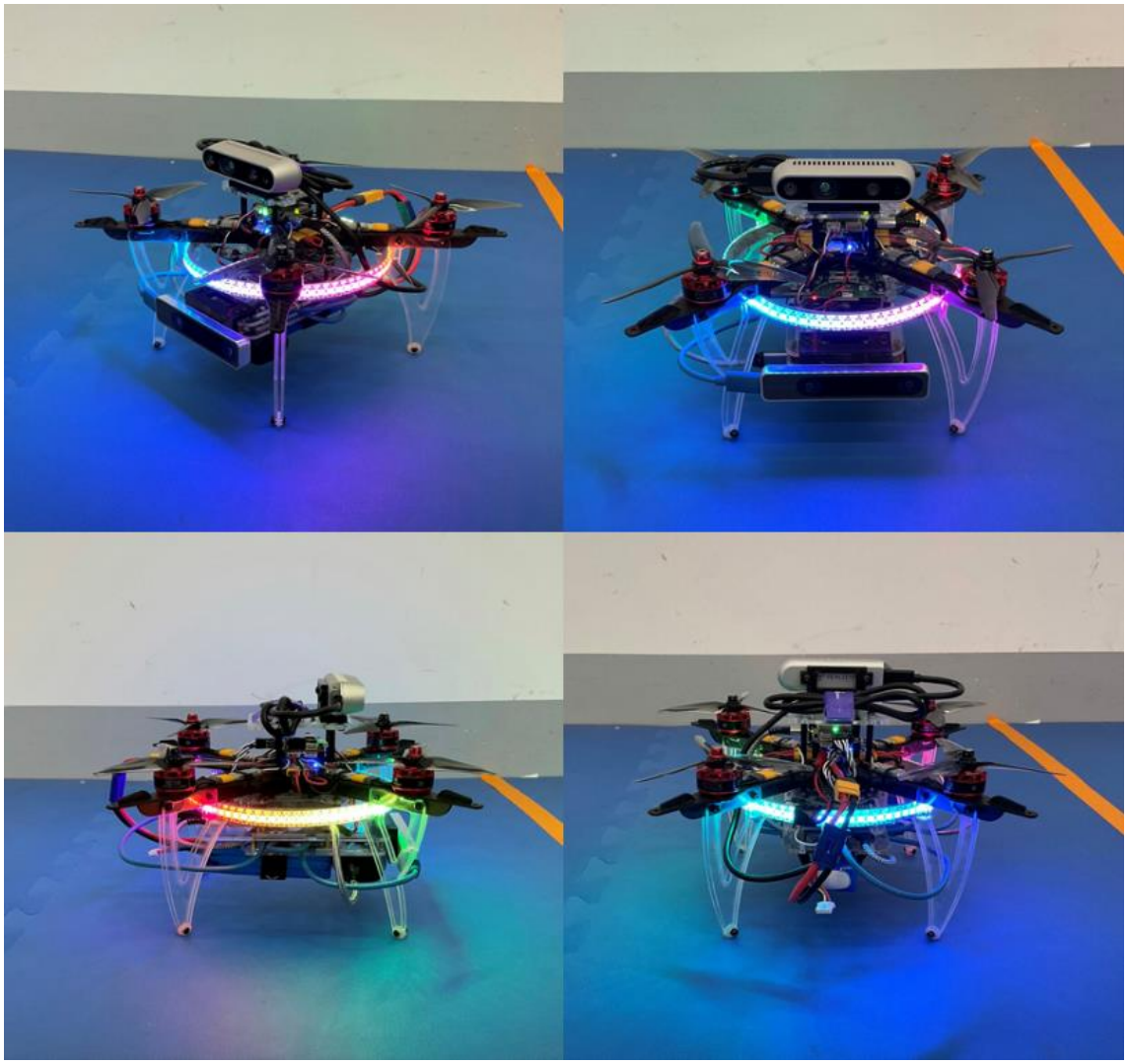


Fig. 33. The testing unit for the proposed system.

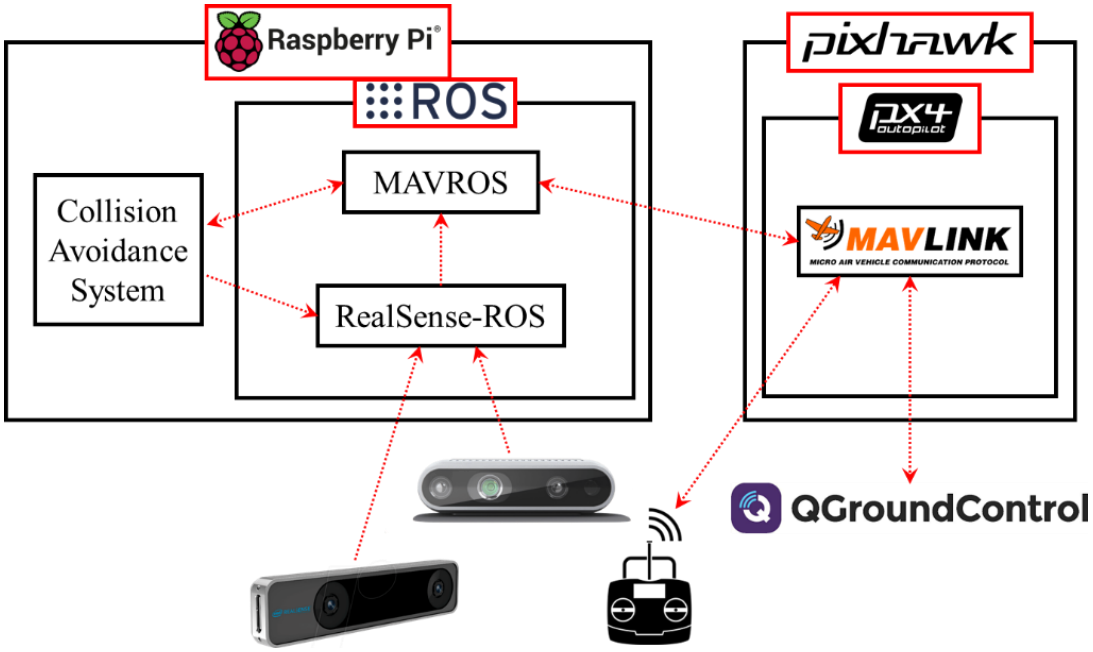


Fig. 34. A schematic diagram of the system.

B. Test results – video sequence

To test the detection and tracking capabilities of the proposed system, a relevant test was conducted. The detection and tracking algorithm was tested on video sequences with moving objects (Fig. 35). Since these videos do not have depth information, the decision making and avoidance procedures were tested separately.

Fig. 35. Video sequences for testing

Tracking results for these video sequences is as Fig. 36. The video results are at https://youtu.be/JvrTRMg2udU. The tracking algorithm works in real-time at 60 FPS at a Raspberry Pi 4 computer. It compensates noise components with various filters and keeps track of objects even in the case of detection failures.

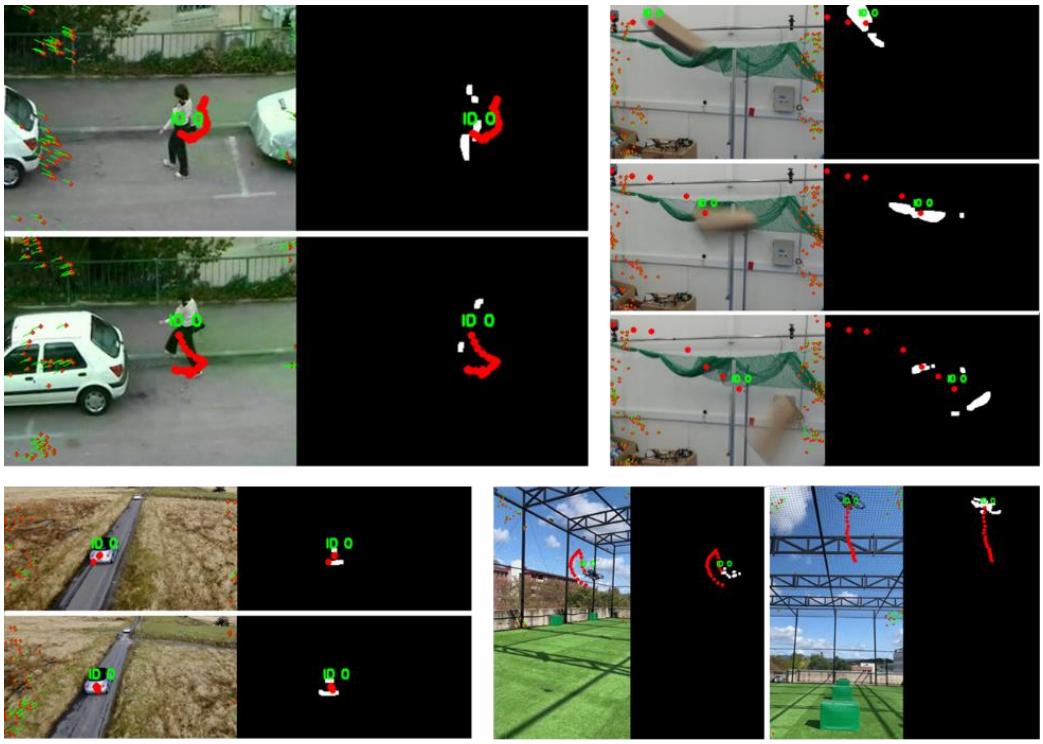


Fig. 36. Detection and tracking test results

C. Test results – collision avoidance

To test the integrated system for collision avoidance, the testing unit was set to fly forwards at 2m/s until avoidance maneuvers were required. Due to safety issues, the safe window radius was set to 1.5 meters, the avoidance threshold to 1 meter, and the maximum maneuver speed to 3 m/s. Fig. 37 is an example detection result from the vehicle's onboard computer. Collision avoidance test results are as Fig. 38, and more detailed video results are at https://youtu.be/JvrTRMg2udU.

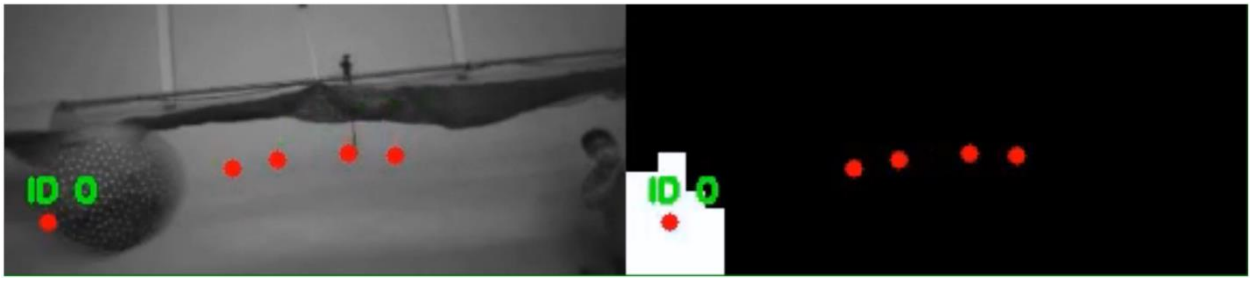


Fig. 37. Detection and tracking a flying ball

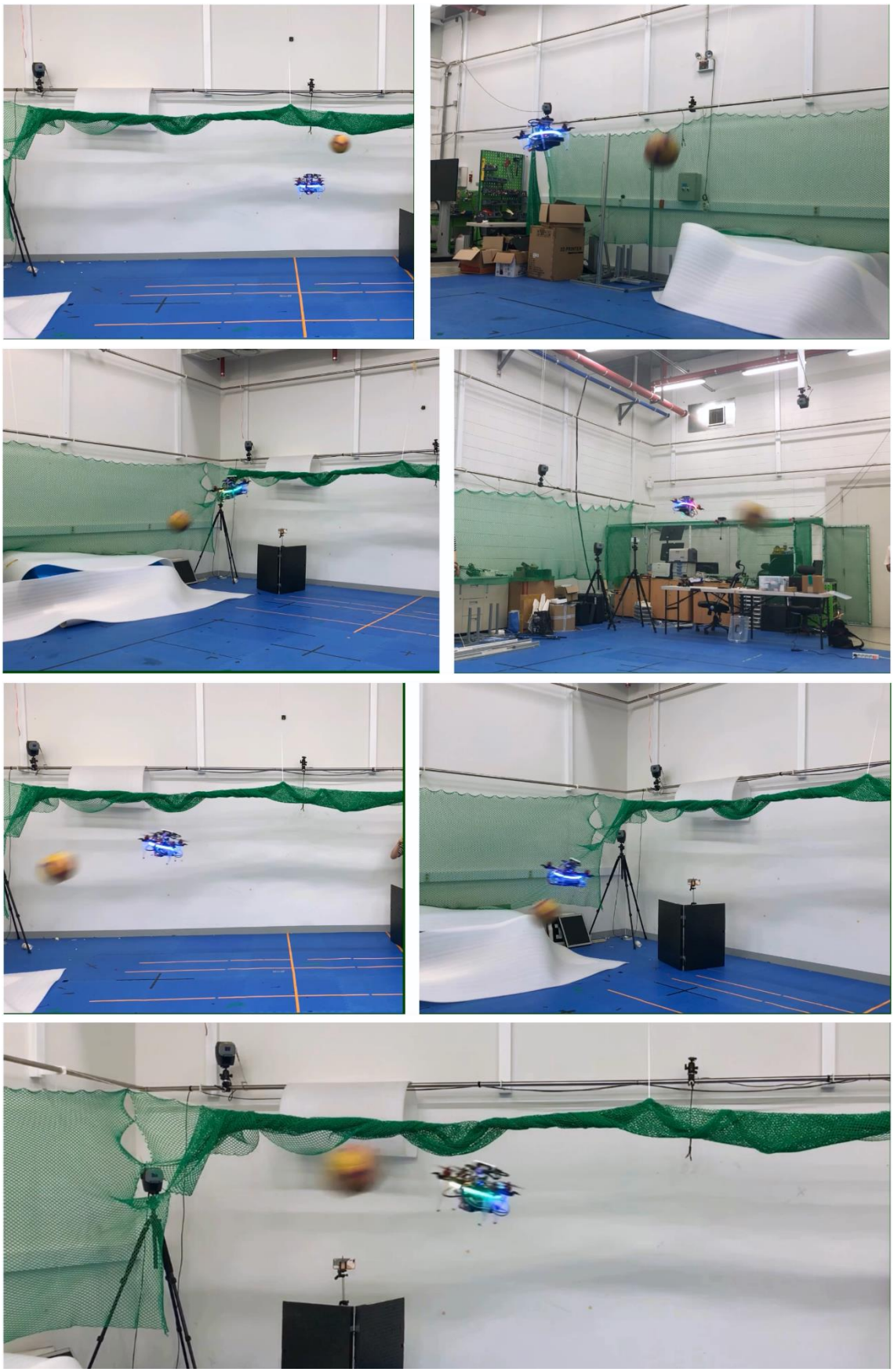


Fig. 38. Collision avoidance test results.

6. Conclusion and Future Works

A vision-based mid-air collision avoidance system for UAVs was proposed and implemented. Video data from a front-mounted camera is utilized for moving object detection. Background scene transition due to ego-motion of the UAV is approximated with 2-D projective transformation, and the homography matrix is computed with optical flow. To reduce required computational load and improve the quality of approximation, optical flow is calculated only at edge regions of video frames. The previous and next frame is 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 correctly identify multiple detected image patches as a single object. A distance-based tracking algorithm assigns object identities and tracks them across frames, and incorporates an additional noise filtering procedure based on tracked period. A stereo camera system measures distances to detected objects, and this information is used for determining whether if avoidance maneuvers must be executed. Limitation in measurement range of stereo cameras is also taken into consideration. The system effectively detects, recognizes and tracks moving objects in its field of view. It is implemented onto a test vehicle and performs mid-air collision avoidance to demonstrate its effectiveness in real-world conditions.

Future works would involve extending the Kalman filter into using full 3-D coordinates and not just 2-D pixel coordinates. This would make the constant velocity (CV) model more reasonable since the 3-D coordinates are real physical quantities. Also, a quantitative analysis of the performance metrics of the proposed system and comparison with other systems would be required.

References

[1] A. McFadyen, L. Mejias. (2016). "*A survey of autonomous vision-based See and Avoid for Unmanned Aircraft Systems*", Progress in Aerospace Sciences, 80, pp. 1-17.

[2] C. Goerzen, Z. Kong, B. Mettler. (2009). "*A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance*", Journal of Intelligent and Robotic Systems, 57, pp. 65.

[3] M. Senanayake, I. Senthooran, J.C. Barca, H. Chung, J. Kamruzzaman, M. Murshed. (2016). "*Search and tracking algorithms for swarms of robots: A survey*", Robotics and Autonomous Systems, 75, pp. 422-434.

[4] MarketsAndMarkets. (2021). "*Unmanned Aerial Vehicle (UAV) Market by Point of Sale, Systems, Platform (Civil & Commercial, and Defense & Governement), Function, End Use, Application, Type, Mode of Operation, MTOW, Range, and Region - Global Forecast to 2026*".

[5] V. Chamola, P. Kotesh, A. Agarwal, Naren, N. Gupta, M. Guizani. (2021). "*A Comprehensive Review of Unmanned Aerial Vehicle Attacks and Neutralization Techniques*", Ad Hoc Networks, 111, 102324.

[6] D. Floreano, R.J. Wood. (2015). "*Science, technology and the future of small autonomous drones*", Nature, 521, pp. 460-466.

[7] C. Zhuge, Y. Cai, Z. Tang. (2017). "*A Novel Dynamic Obstacle Avoidance Algorithm Based on Collision Time Histogram*", Chinese Journal of Electronics, 26, pp. 522-529.

[8] H. Chao, Y. Cao, Y. Chen. (2007) "*Autopilots for Small Fixed-Wing Unmanned Air Vehicles: A Survey*". Proc. 2007 International Conference on Mechatronics and Automation, pp. 3144-3149.

[9] D. Shim, H. Chung, H.J. Kim, S. Sastry. (2005). "*Autonomous Exploration In Unknown Urban Environments For Unmanned Aerial Vehicles*", AIAA Guidance, Navigation, and Control Conference and Exhibit, American Institute of Aeronautics and Astronautics.

[10] A. Foka, P. Trahanias. (2007). "*Real-time hierarchical POMDPs for autonomous robot navigation*", Robotics and Autonomous Systems, 55, pp. 561-571.

[11] C.H.R. Everett. (1989). "*Survey of collision avoidance and ranging sensors for mobile robots*", Robotics and Autonomous Systems, 5, pp. 5-67.

[12] S. Saha, A. Natraj, S. Waharte. (2014) "*A real-time monocular vision-based frontal obstacle detection and avoidance for low cost UAVs in GPS denied environment*". Proc. 2014 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology, pp. 189-195.

[13] L. Mejias, S. McNamara, J. Lai, J. Ford. (2010) "*Vision-based detection and tracking of aerial targets for UAV collision avoidance*". Proc. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 87-92.

[14] S.A.S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, J. Plosila. (2020) "*Towards Real-Time Edge Detection for Event Cameras Based on Lifetime and Dynamic Slicing*". Proc. Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020), pp. 584-593.

[15] T.-J. Lee, D.-H. Yi, D.-I. Cho. (2016). "*A Monocular Vision Sensor-Based Obstacle Detection Algorithm for Autonomous Robots*", Sensors, 16.

[16] A. Uddin Haque, A. Nejadpak. (2017). "*Obstacle Avoidance Using Stereo Camera*" (pp. arXiv:1705.04114).

[17] D. Falanga, S. Kim, D. Scaramuzza. (2019). "*How Fast Is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid*", IEEE Robotics and Automation Letters, 4, pp. 1884-1891.

[18] Y.K. Kwag, C.H. Chung. (2007) "*UAV based collision avoidance radar sensor*". Proc. 2007 IEEE International Geoscience and Remote Sensing Symposium, pp. 639-642.

[19] A. Moses, M.J. Rutherford, M. Kontitsis, K.P. Valavanis. (2014). "*UAV-borne X-band radar for collision avoidance*", Robotica, 32, pp. 97-114.

[20] K. Bilimoria. (2000). "*A geometric optimization approach to aircraft conflict resolution*", 18th Applied Aerodynamics Conference, American Institute of Aeronautics and Astronautics.

[21] J. Seo, Y. Kim, S. Kim, A. Tsourdos. (2017). "*Collision Avoidance Strategies for Unmanned Aerial Vehicles in Formation Flight*", IEEE Transactions on Aerospace and Electronic Systems, 53, pp. 2718-2734.

[22] L.N. Ha, D.H. Bui, S.K. Hong. (2019). "*Nonlinear Control for Autonomous Trajectory Tracking While Considering Collision Avoidance of UAVs Based on Geometric Relations*", Energies, 12.

[23] Z. Lin, L. Castano, E. Mortimer, H. Xu. (2020). "*Fast 3D Collision Avoidance Algorithm for Fixed Wing UAS*", Journal of Intelligent & Robotic Systems, 97, pp. 577-604.

[24] M. Wang, H. Voos, D. Su. (2018) "*Robust Online Obstacle Detection and Tracking for Collision-Free Navigation of Multirotor UAVs in Complex Environments*". Proc. 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 1228-1234.

[25] S.U. Sharma, D.J. Shah. (2017). "*A Practical Animal Detection and Collision Avoidance System Using Computer Vision Technique*", IEEE Access, 5, pp. 347-358.

[26] M.C. De Simone, Z.B. Rivera, D. Guida. (2018). "*Obstacle Avoidance System for Unmanned Ground Vehicles by Using Ultrasonic Sensors*", Machines, 6.

[27] Y. Yu, W. Tingting, C. Long, Z. Weiwei. (2018) "*Stereo vision based obstacle avoidance strategy for quadcopter UAV*". Proc. 2018 Chinese Control and Decision Conference (CCDC), pp. 490-494.

[28] Y. Kwang Moo, Y. Kimin, K. Soo Wan, C. Hyung Jin, J. Hawook, C. Jin Young. (2013) "*Detection of Moving Objects with Non-stationary Cameras in 5.8ms: Bringing Motion Detection to Your Mobile Device*". Proc. 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 27-34.

[29] T. Chen, S. Lu. (2017). "*Object-Level Motion Detection from Moving Cameras*", IEEE Transactions on Circuits and Systems for Video Technology, 27, pp. 2333-2343.

[30] J. Kim, X. Wang, H. Wang, C. Zhu, D. Kim. (2013). "*Fast moving object detection with non-stationary background*", Multimedia Tools and Applications, 67, pp. 311-335.

[31] Y. Yang, A. Loquercio, D. Scaramuzza, S. Soatto. (2019) "*Unsupervised Moving Object Detection via Contextual Information Separation*". Proc. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 879-888.

[32] V. Guizilini, F. Ramos. (2015). "*Online self-supervised learning for dynamic object segmentation*", The International Journal of Robotics Research, 34, pp. 559-581.

[33] M.A. Fischler, R.C. Bolles. (1981). "*Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*", Commun. ACM, 24, pp. 381-395.

[34] D.L. Massart, L. Kaufman, P.J. Rousseeuw, A. Leroy. (1986). "*Least median of squares: a robust method for outlier and model error detection in regression and calibration*", Analytica Chimica Acta, 187, pp. 171-179.

[35] K. Kale, S. Pawar, P. Dhulekar. (2015) "*Moving object tracking using optical flow and motion vector estimation*". Proc. 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), pp. 1-6.

[36] G. Bleser, G. Hendeby. (2009) "*Using optical flow as lightweight SLAM alternative*". Proc. 2009 8th IEEE International Symposium on Mixed and Augmented Reality, pp. 175-176.

[37] B.D. Lucas, T. Kanade. (1981). "*An iterative image registration technique with an application to stereo vision*" Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2, pp. 674-679.

[38] B.K.P. Horn, B.G. Schunck. (1981). "*Determining optical flow*", Artificial Intelligence, 17, pp 185-203.

[39] R.A. Haddad, A.N. Akansu. (1991). "*A class of fast Gaussian binomial filters for speech and image*

processing", IEEE Transactions on Signal Processing, 39, pp. 723-727.

[40] J. Canny. (1986). "*A Computational Approach to Edge Detection*", IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8, pp. 679-698.

[41] M. Ester, H.-P. Kriegel, J. Sander, X. Xu. (1996). "*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, pp. 226-231.

[42] S. Lloyd. (1982). "*Least squares quantization in PCM*", IEEE Transactions on Information Theory, 28, pp. 129-137.

[43] Q. Li, R. Li, K. Ji, W. Dai. (2015) "*Kalman Filter and Its Application*". Proc. 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), pp. 74-77.