Ground-Based Vision Tracker for Advanced Air Mobility and Urban Air Mobility

Evan Kawamura *1, Keerthana Kannan †1, Thomas Lombaerts ‡1, Vahram Stepanyan§1, Chester Dolph ¶2, and Corey Ippolito ‖1

1 *NASA Ames Research Center, Moffett Field, Mountain View, CA 94035*

2 *NASA Langley Research Center, 1 Nasa Dr, Hampton, VA 23666*

Advanced Air Mobility (AAM) and Urban Air Mobility (UAM) require aircraft surveillance and monitoring for safety and security. Persistent tracking of flying objects provides Air Traffic Control (ATC) and Air Traffic Management (ATM) continuous coverage and knowledge of the National Airspace System (NAS). Future AAM and UAM operations will have more aircraft than commercial aircraft in the NAS, coupled with the dense AAM/UAM operations in urban environments, so employing the existing ATC/ATM architectures poses considerable challenges. A first step in creating a similar ATC/ATM architecture for AAM/UAM will require ground-based and airborne sensors to provide monitoring. This paper proposes a vision-based tracking method with static cameras by utilizing image subtraction and blob detection, which avoids adding additional electromagnetic interferences in the environment with sensors such as radar. The Ground-Based Vision Tracker (GBVT) outputs the detected objects' azimuth and elevation angles from Unmanned Aerial System (UAS) flight tests. Future and ongoing work includes sending the detected objects' azimuth and elevation angles as inputs for an extended Kalman filter (EKF) to estimate the position and velocity of the detected object.

I. Introduction

Advanced Air Mobility (AAM) and Urban Air Mobility (UAM) require aircraft surveillance and monitoring for safety and security purposes. Since there will be numerous more AAM/UAM aircraft than commercial airplanes flying in the airspace, it will be challenging to scale conventional Air Traffic Control (ATC) and Air Traffic Management (ATM) procedures and protocol for AAM/UAM concepts of operation. One way to reduce scaling complexities and increase efficiency is to implement distributed sensors in the environment for monitoring the airspace. Then, the distributed sensors can provide aircraft tracking information to the modified ATC/ATM architectures by leveraging automatic and passive tracking capabilities. Therefore, these modified ATC/ATM architectures will have an efficient and scalable solution for monitoring the airspace with increased awareness, path planning, and contingency management. The distributed sensing project uses radar and cameras in urban environments for several applications such as monitoring, surveillance, tracking, and estimation [1].

There are radar-based tracking methods for UAS, which could extend to AAM tracking. Poitevin et al. describe several challenges for detecting UAS by radar due to their speed and size. Consequently, radar-based detection methods need to quickly scan large volumes and eliminate non-UAS targets (false alarms) [2]. One study shows radar-based techniques for detecting UAS: millimeter wave (mmWave), ultra-wideband, 2.4 GHz, MIMO, and non-line-of-sight ("seeing around or beyond the corner") [3]. Accardo et al. conducted UAS flight tests with radar for obstacle detection and tracking for noncooperative UAS collision avoidance [4]. Another study shows how a radar tracking system in urban environments finds multiple non-UAS targets such as birds, people, and cars with filtered micro-Doppler signatures to distinguish UAS targets from non-UAS targets [5]. Samaras et al. utilize a deep neural network to distinguish UAS targets from non-UAS targets after obtaining radar measurements of various targets [6]. Mohajerin et al. demonstrate

*Computer/GNC Engineer, Intelligent Systems Division, NASA Ames Research Center, Moffett Field, Mountain View, CA 94035, USA.

†Software Engineer, KBR Wyle Services, Intelligent Systems Division, NASA Ames Research Center, Moffett Field, Mountain View, CA 94035, USA.

‡Aerospace Research Engineer, KBR Wyle Services, Intelligent Systems Division, NASA Ames Research Center, Moffett Field, Mountain View, CA 94035, USA.

§Aerospace Research Engineer, KBR Wyle Services, Intelligent Systems Division, NASA Ames Research Center, Moffett Field, Mountain View, CA 94035, USA.

¶Aerospace Engineer, Aeronautics Systems Engineering Branch, NASA Langley Research Center, 1 Nasa Dr, Hampton, VA 23666.

‖Aerospace Scientist, Intelligent Systems Division, NASA Ames Research Center, Moffett Field, Mountain View, CA 94035, USA how associating different radar tracks of objects is crucial in crowded airspace, especially identifying UAS tracks from other types of tracks such as airplanes and birds, and simulation results demonstrate 99% accuracy for correctly labeling the Unmanned Aerial Vehicle (UAV) tracks [7].

There are several studies and projects about vision-based tracking. Fu et al. show how a real-time, adaptive visual algorithm with a Multiple-Instance (MI) learning approach, Multiple-Classifier (MC) voting mechanism, and Multiple-Resolution (MR) representation strategy solves online learning and tracking arbitrary aircraft and intruders in the air with experimental results [8]. Sattigeri et al. combined a vision-based target tracker with a neural network and a Kalman filter to create an adaptive target state estimator [9]. Sivaraman and Trivedi conducted a survey of on-road vision-based vehicle detectors to show various monocular and stereo vision-based trackers [10]. Toyama and Hager utilize an Incremental Focus of Attention (IFA) hierarchical architecture, which yields robust, adaptive, and real-time vision-based tracking [11]. Joo et al. combined onboard inertial navigation system (INS) telemetry data and a ground-based monocular camera running optical flow into an Extended Kalman Filter (EKF) to generate robust and accurate state estimation of the tracked UAV [12]. Bharati et al. utilize an adaptive object detection method, which yields fast and robust object tracking for airplanes [13]. Ganti and Kim created a tracking system to identify and detect UAS with image processing and mechanical tracking, which feeds into countermeasures such as keeping malicious or harmful UAS away from restricted and residential areas [14]. Iswanto and Li designed a visual tracking algorithm that combines mean-shift and a particle-Kalman filter such that the particle filter executes only when occlusions occur or the mean-shift tracking result is not accurate [15]. The Edge Company has a system for detecting birds and drones with optical artificial intelligence to identify, track, count, and classify the birds and drones in real-time, which provides insight for monitoring the airspace around airports, vertiports, and spaceports [16].

This paper focuses on vision-based tracking with ground-based distributed sensors in the environment. The proposed research in this paper demonstrates a Ground-Based Vision Tracker (GBVT) for distributed and calibrated cameras in an urban environment to simulate future AAM/UAM operations for monitoring the airspace. Post-processing a ground-based camera's video of a UAV flight includes image subtraction to capture a moving UAV across the camera's field of view. The processed GBVT results feed into a Ground-Based Object Tracker (GBOT) EKF (see Ref. [17, 18] for more details) for estimating the tracked object's position and velocity. Figure 1 shows the high-level block diagram of how the GBVT takes the distributed camera images and videos, processes them, outputs the tracked object information, and feeds the data into the GBOT EKF (blue box). The block titled, "Tracks of Objects," has the pixel coordinates, azimuth/elevation angles, and object radius for each tracked object. The origin for the pixel coordinates for each tracked object is at the top left corner of the image, and section III shows the computations for converting the target's pixel coordinates into the azimuth and elevation angles in the sensor's frame.

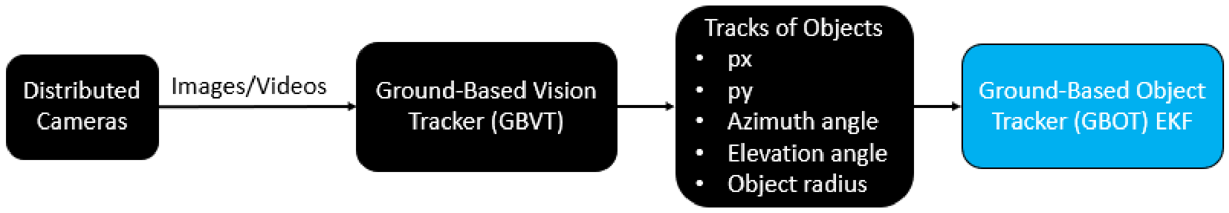


Fig. 1 High-Level Block Diagram: GBVT and GBOT EKF

This paper's organization is as follows. Section II discusses the proposed solution for the GBVT, and section III shows the computations and drawings for the sensor azimuth and elevation angles. Section IV provides the details about the UAS flight tests to test the GBVT, and section V describes the results for those tests. Finally, section VI ends with concluding remarks, future work, and key takeaways.

II. Ground-Based Vision Tracker

There are two different implementations for the GBVT: one uses the MATLAB Computer Vision Toolbox, while the other uses OpenCV in C++. However, both implementations utilize image subtraction and blob detection. There are similar steps in the procedure but with differences due to programming language and data storage. Overall, the MATLAB implementation of the GBVT served as a prototype for preliminary testing, and then re-implemented in C++ using OpenCV with minor adjustments. Therefore, most of the details provided in this paper will focus on the OpenCV implementation.

A. MATLAB

Figure 2 shows the MATLAB image subtraction flowchart. The GBVT in MATLAB uses image subtraction to find the objects that have moved since the last frame. The main steps are in Figure2 and also summarized here:

1) Obtain two frames (consecutive or skip)
2) Apply grayscale to both frames, i.e., convert RGB to grayscale
3) Apply image subtraction between the two frames
4) Increase the intensity (brightness) of the subtracted image to see the objects more easily
5) Apply threshold for the subtracted frames to remove objects with lower intensity
6) Convert the subtracted image to binary because blob detection needs binary images
7) Apply blob detection on the binary image to find the largest blobs (in this particular case, look for the five largest)
8) For each of the five detected blobs
    1) Determine the azimuth and elevation angles
    2) Determine the bounding box locations (pixel coordinates)
    3) Update the centroid log/table with the new information: frame number, time, pixel coordinates, azimuth and elevation angles, bounding box parameters: width, height, and top left pixel coordinates

The main two computer vision methods are: 1) image subtraction and 2) blob detection. The image subtraction is a straightforward method that takes the absolute difference between the pixels in the two images:

$$\Delta Img = |Img_i - Img_{i-1}| \tag{1}$$

such that Img denotes an image and i is the index for the ith image. MATLAB's *regionprops* function looks at connected regions and returns stats about those regions, which includes the area and centroid point. [*]

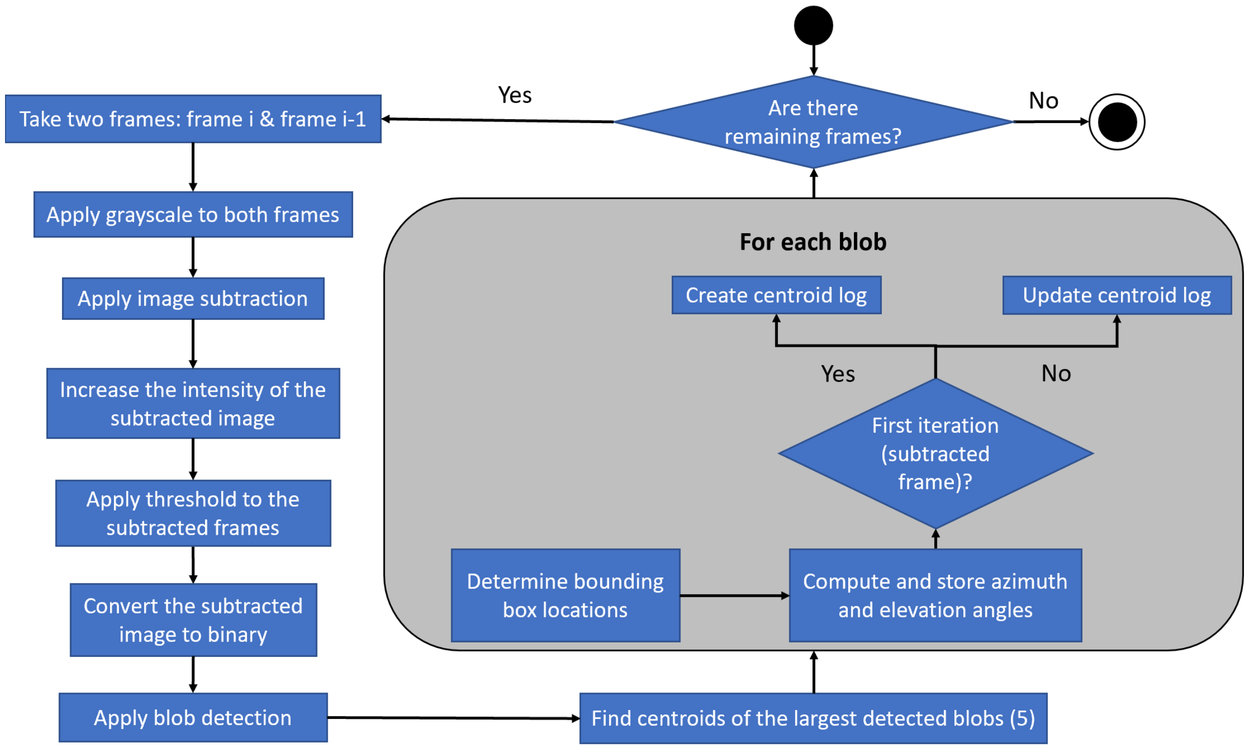

Take two frames: frame i & frame i-1
Yes
Are there remaining frames?
No
Apply grayscale to both frames
For each blob
Apply image subtraction
Create centroid log
Update centroid log
Increase the intensity of the subtracted image
Yes
No
First iteration (subtracted frame)?
Apply threshold to the subtracted frames
Determine bounding box locations
Compute and store azimuth and elevation angles
Convert the subtracted image to binary
Apply blob detection
Find centroids of the largest detected blobs (5)

Fig. 2 MATLAB Image Subtraction Flowchart

[*]https://www.mathworks.com/help/images/ref/regionprops.html

B. OpenCV: C++

Figure 3 shows how the C++ version of the GBVT (CGBVT) works and when it is in the calibration phase or tracking phase based on the current frame number. Future flight tests will need real-time implementation of the GBVT. The default and fixed frame rate options apply for post-processing. The variable frame rate involves real-time processing by skipping to the next available frame after processing the current frame, but it is currently a work-in-progress. Overall, it has three different timing modes:

1) The default rate processes all available frames.
2) The fixed frame rate has a constant frame difference between iterations.
3) After processing the current frame, the variable frame rate jumps to the next available frame.

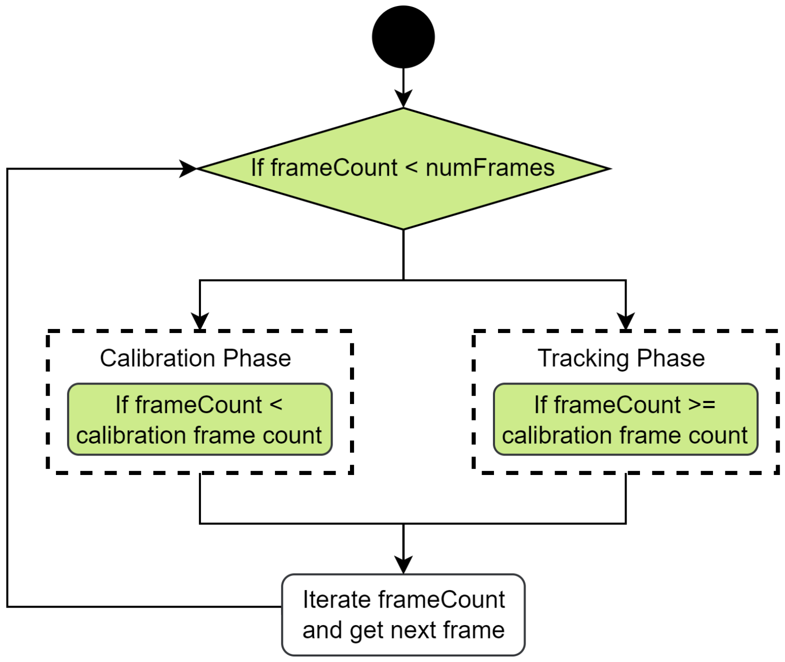

If frameCount < numFrames
Calibration Phase
If frameCount < calibration frame count
Tracking Phase
If frameCount >= calibration frame count
Iterate frameCount and get next frame

Fig. 3 Image Tracker Flowchart: High-Level

Both implementations of the GBVT have calibration and tracking phases, and Figure 4 shows the image processing pipeline for the CGBVT. The calibration phase contains only image/background subtraction and does not include blob detection, i.e., stops at the fourth box in Figure 4. Conversely, the tracking phase includes both image/background subtraction and blob detection, so it follows all the steps in Figure 4.

The CGBVT has a user interface for users to draw ROIs and masks to reduce the visual coverage for tracking. Users can also pause tracking to redraw ROIs and masks to crop out undesired features and move the ROI to follow the desired object, which yields better tracking results. Alternatively, users can draw a large ROI with masks to cover the coverage area for the entire flight without pausing and redrawing. Future work can include automating the ROI and masks to increase scalability to deploy the GBVT at several locations. Figure 5 shows a flowchart for the main steps in the image tracker after feature detection and contains the steps for determining the detected object's ID number.

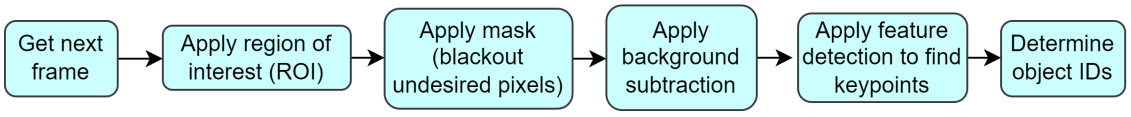

Get next frame
Apply region of interest (ROI)
Apply mask (blackout undesired pixels)
Apply background subtraction
Apply feature detection to find keypoints
Determine object IDs

Fig. 4 Image Processing Pipeline

Overall, the main steps for the CGBVT are:

1) Obtain two frames (consecutive or skip)
2) Apply grayscale to both frames, i.e., convert RGB to grayscale

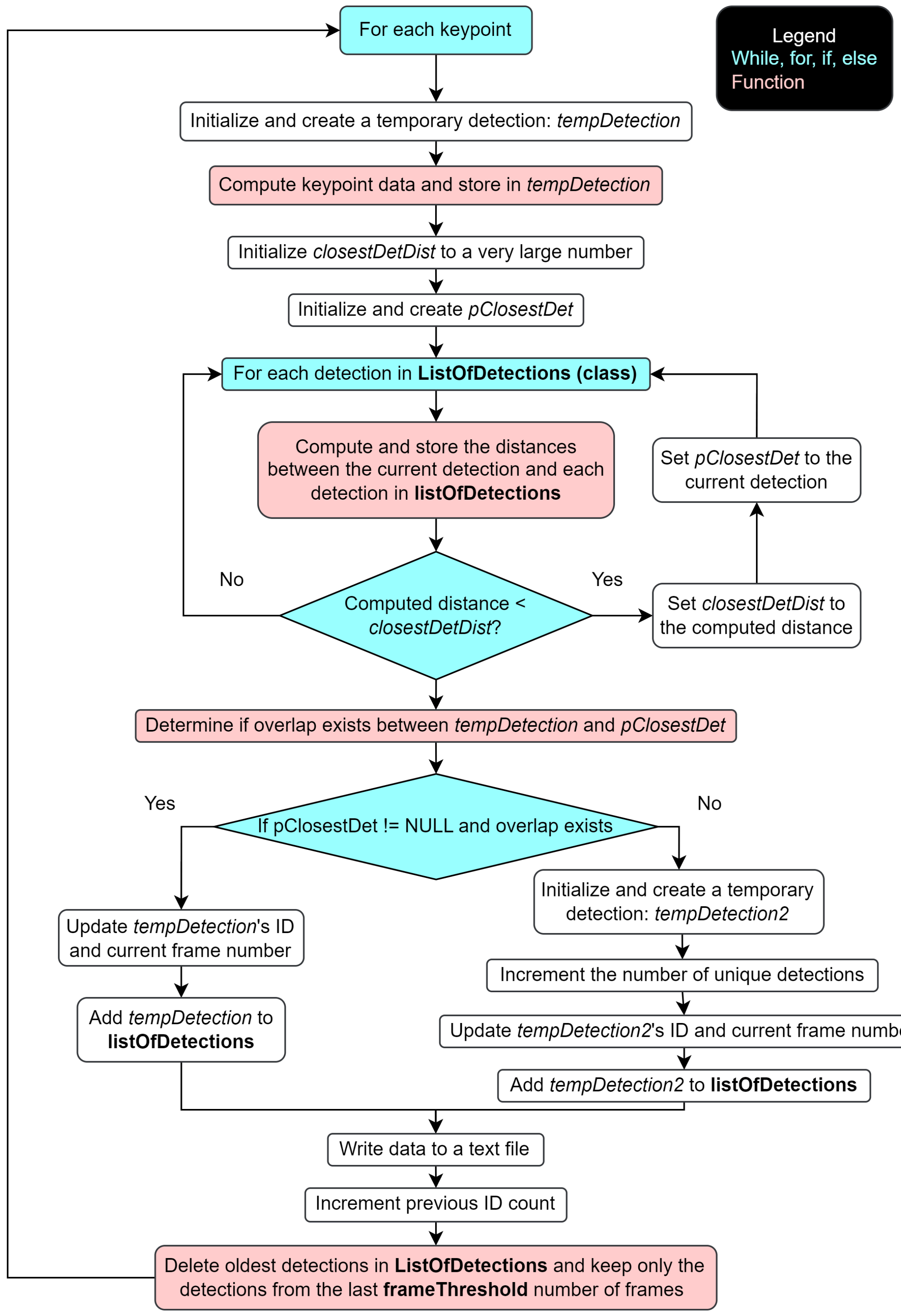

For each keypoint
Legend
While, for, if, else
Function
Initialize and create a temporary detection: tempDetection
Compute keypoint data and store in tempDetection
Initialize closestDetDist to a very large number
Initialize and create pClosestDet
For each detection in ListOfDetections (class)
Compute and store the distances between the current detection and each detection in listOfDetections
Set pClosestDet to the current detection
No
Yes
Computed distance < closestDetDist?
Set closestDetDist to the computed distance
Determine if overlap exists between tempDetection and pClosestDet
Yes
No
If pClosestDet != NULL and overlap exists
Initialize and create a temporary detection: tempDetection2
Update tempDetection's ID and current frame number
Increment the number of unique detections
Add tempDetection to listOfDetections
Update tempDetection2's ID and current frame number
Add tempDetection2 to listOfDetections
Write data to a text file
Increment previous ID count
Delete oldest detections in ListOfDetections and keep only the detections from the last frameThreshold number of frames

Fig. 5 **CGBVT: Tracking Phase Flowchart - pClosestDet is a pointer to the closest detection, tempDetection and tempDetection2 are instances of the Detection class, ListOfDetection has a list of all the previous detections, frameThreshold is set to 540 ("forget" detections after 540 frames pass)**

3) Apply OpenCV's background subtraction (MOG2) [†] between the two frames to learn the background model, which helps remove static objects
4) Apply blob detection on the binary image [‡]
5) Run the tracking phase (see Figure 5)
    1) Compute the distance between the current and previous detections, i.e., pairwise comparison for all current to previous blobs
    2) Assign the same ID number for close detections that overlap
    3) Assign a new ID number for current detections that do not overlap with previous detections

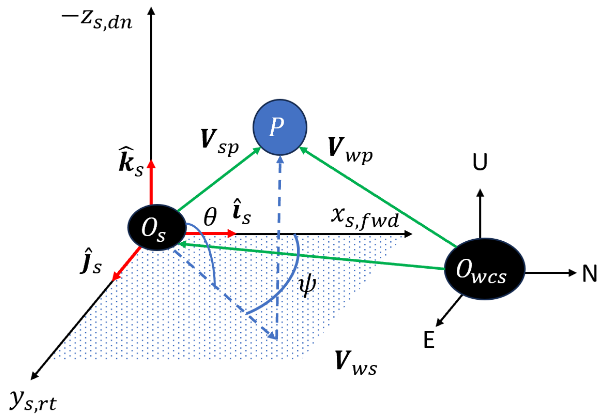
III. Distributed Cameras and Radar: Azimuth and Elevation Angles

This section follows Ref. [19] for computing the azimuth and elevation angles.

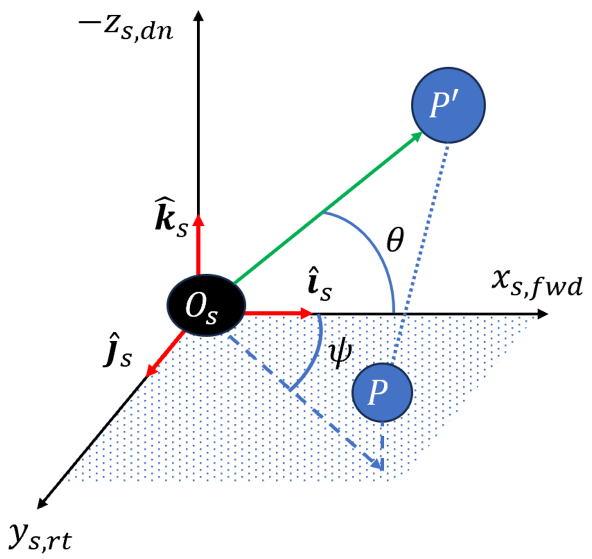
Let East North Up (ENU) be the World Coordinate System (WCS) and defined as an inertial frame fixed on the ground with an origin, O_{WCS}. Let the sensor frame, s, be defined as an x-fwd, y-right, z-down coordinate system with its origin O_s at the center of projection for the sensor. The x-axis points in the direction of the sensor, y-axis points to the right, and the z-axis points down from the sensor's point of view to complete a right-handed system.

Figure 17a shows the unit vectors, $\hat{\boldsymbol{\imath}}_s, \hat{\boldsymbol{\jmath}}_s, \hat{\boldsymbol{k}}_s$ in the sensor frame in the x, y, z directions, respectively. Let P denote a target such that $\boldsymbol{V}_{sp}$ be the vector from O_s to the target point P. Let $\boldsymbol{r}_{xy}$ be the projection of $\boldsymbol{V}_{sp}$ to the X-Y sensor axis plane. Let ψ denote the azimuth angle and defined as the angle between $\hat{\boldsymbol{\imath}}_s$ and $\boldsymbol{r}_{xy}$, defined as positive right-hand rotation around $\hat{\boldsymbol{k}}_s$. Then, let θ denote the elevation angle defined as the angle between $\boldsymbol{r}_{xy}$ and $\boldsymbol{V}_{sp}$, positive right from $\boldsymbol{r}_{xy}$ to Vsp (i.e., positive rotation around the cross-product of $\boldsymbol{r}_{xy} \times \boldsymbol{V}_{sp}$).

Finally, projecting the target point onto the X-Z plane provides an elevation angle α_{xz} in the XZ plane, as shown in Figure 17b, which shows the mapping from O_{wcs} to O_s. Figure 6 shows the target's azimuth and elevation angles (ψ, θ) in 3D space and the X-Z plane.



(a) Point in 3D Space: Azimuth and Elevation



(b) Projected Point onto the X-Z Plane: Azimuth and Elevation

Fig. 6 Azimuth and Elevation Angles (resembles figure 5 of Ref. [19])

Using the total pixel width, W, and horizontal field of view in radians, FOV_h, provides the focal length, f, as shown in Eqn. (2).

$$f = \frac{W}{2 \cdot \tan\left(\frac{FOV_h}{2}\right)} \text{(constant)} \tag{2}$$

[†]https://docs.opencv.org/4.x/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html
[‡]https://docs.opencv.org/4.x/d0/d13/classcv_1_1Feature2D.html#aa4e9a7082ec61ebc108806704fbd7887

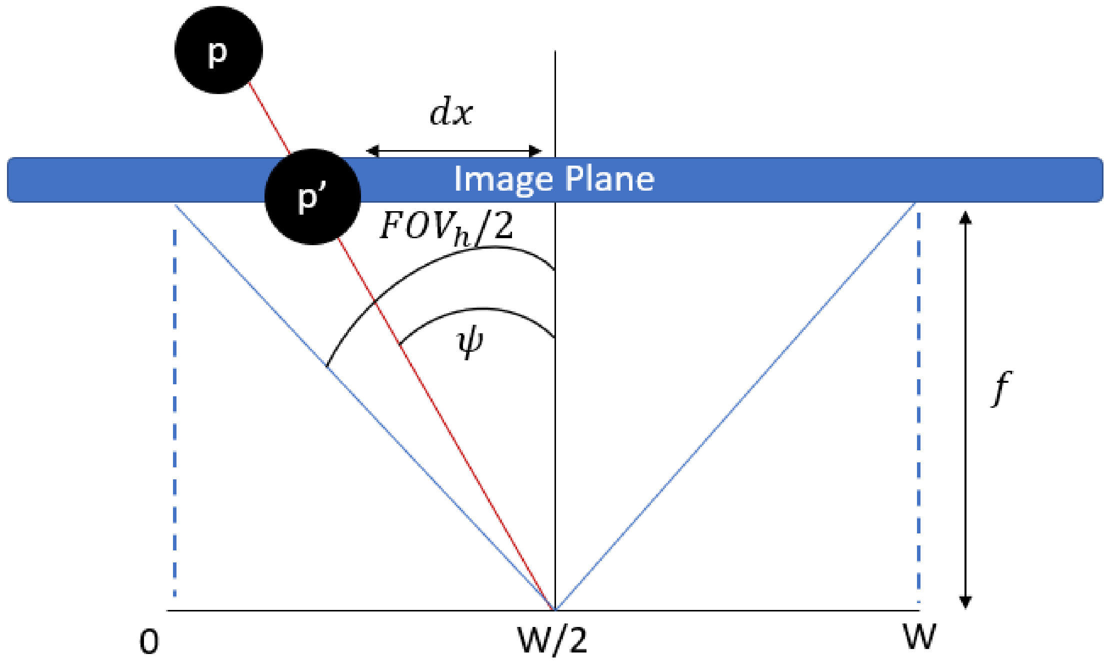


Fig. 7 Azimuth and Elevation Angles to Pixel Coordinates: 1) given the horizontal field of view and image width in pixels, compute the focal length (Eqn. (2)**), 2) compute the horizontal pixel offset, dx (Eqn.** (3)**), 3) compute the azimuth and elevation angles (Eqn.** (4)**)**

The next equations describe the computations for converting the target pixels into azimuth and elevation angles (see Figures 7-8). Equation (3) shows the difference between the target's pixels and the midpoint of the total pixel width and height, $W\&H$, respectively.

$$dx = px - W/2\,, dy = py - H/2 \tag{3}$$

Finally, compute the azimuth and elevation angles using the focal length from Eqn. (2) and the computed target pixel offsets from Eqn. (3). The minus sign in the equation for θ in Eqn. (4) exists since py is positive in the downward direction, i.e., the pixel coordinates' origin is at the image's top left corner. Figures 7-8b show drawings for computing the azimuth and elevation angles from the focal length, pixel coordinates, camera resolution, and fields of view.

$$\psi = arctan\left(\frac{dx}{f}\right),\ \theta = -arctan\left(\frac{dy}{\sqrt{f^2 + dx^2}}\right) \tag{4}$$

IV. Flight Tests

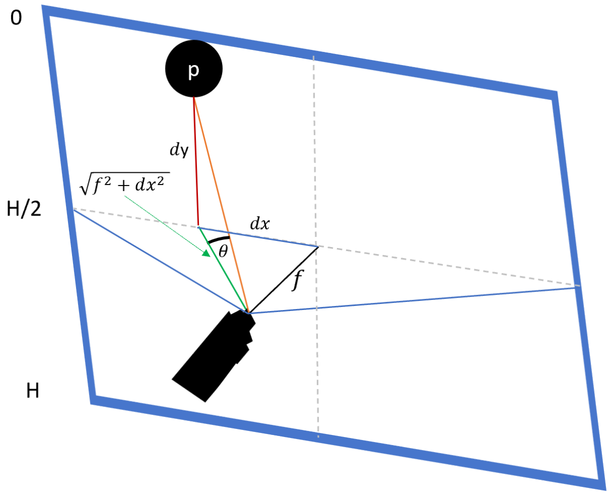
This section describes the UAS flight tests for testing the GBVT. The first subsection describes a preliminary test at NASA Ames with one camera. The second subsection discusses the distributed sensing flight test with four ground cameras.

A. Preliminary Test with One Camera

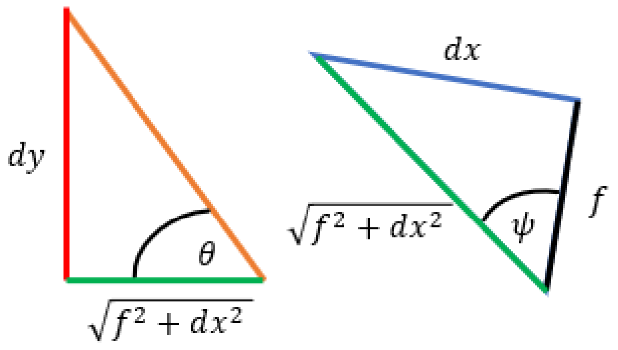
For this preliminary test with one camera, the UAS flies from left to right in the camera's field of view. Figure 9a shows the first frame for the preliminary UAS flight test at Ames, and Figure 9b shows the 7100th frame as the UAS flew to the right and towards the end of the field of view. Table 1 summarizes the camera parameters.

B. Distributed Sensing Test with Four Cameras

Figure 10 shows a generic architecture diagram for the four image trackers for the Ames Build 1 scenario. Each image tracker will run at some preset (1-30 Hz) frequency and receive color images for real-time implementation. Then,

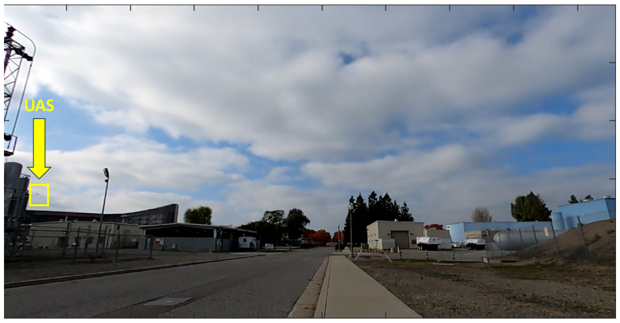


(a) **Pixel Coordinates to Elevation Angle with Camera: elevation angle (θ)**

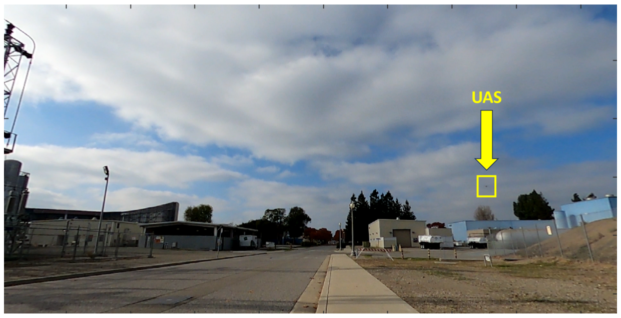


(b) **Pixel Coordinates to Elevation Angles: 1) given dx, dy, and f (blue, red, and black lines, respectively) from Eqns.** (2) **and** (3), **compute $\sqrt{f^2+dx^2}$ (green line); 2) use dy and $\sqrt{f^2+dx^2}$ to solve for θ (elevation angle) from Eqn.** (4)

Fig. 8 Pixel Coordinates to Azimuth and Elevation Angles



(a) Ames Preliminary UAS Flight Test: Frame 1



(b) Ames Preliminary UAS Flight Test: Frame 7100

Fig. 9 Preliminary UAS Flight Test

Table 1 Camera Specs for Preliminary Flight Test

Spec	Value
Horizontal Field of View (°)	110
Vertical Field of View (°)	73.6
Aspect Ratio	1.91
pixel width	1274
pixel height	674
FPS	60

each image tracker will compute the azimuth and elevation angles for detected and tracked objects within their fields of view, which feeds into the data association problem of keeping track of many objects [20]. Finally, the GBOT EKFs presented in Ref. [17, 18, 21] receive multiple cameras' associated objects' azimuth and elevation angles.

Figure 11 shows the sensor locations (yellow circles), and their locations are given in Table 2 [22, 23]. Figures 12 and 13 show the UAV path (blue line) during the flight tests at NASA Ames Research Center. See Ref. [18] for more details about flight test designs for ground-based distributed sensing.

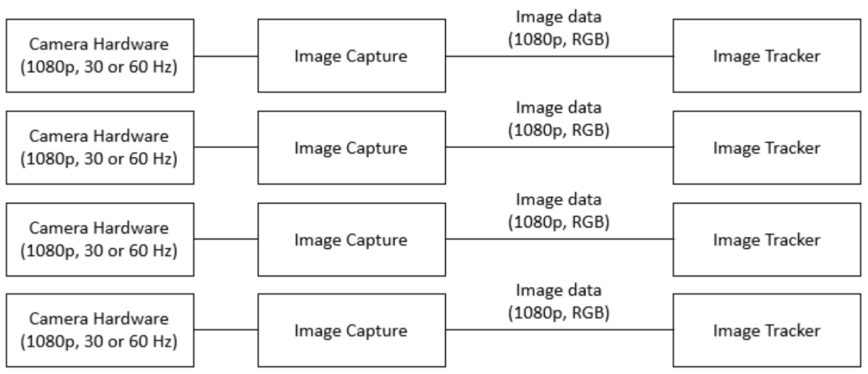


Fig. 10 Generic Distributed Sensing Architecture: frame rates vary between 30 or 60, depending on the camera settings - see Tables 1 & 3

The OpenCV C++ image tracker also utilizes image subtraction [§] and blob detection [¶]. Preliminary tests have a calibration phase that takes the first 150 frames to learn the background model and uses the OpenCV MOG2 Background Subtractor. For rendering and tracking, none of the detections, matching, or plotting occur until after the calibration phase. Two drawbacks for the current implementation of the OpenCV image tracker are: 1) the image tracker does not track the UAV for the frames during the calibration phase, and 2) it must find the UAV in the first frame after the calibration phase, i.e., frame 151 in the preliminary tests. The number of frames during the calibration phase is arbitrary, and future work may consider a more robust and flexible image tracker framework to avoid needing to detect the UAV in the first frame immediately after the calibration phase. Figure 14 shows a diagram of the distance between two detections and the overlap radius. For simplicity, the tracker currently detects the UAV and only considers future detections that are within an overlap distance of:

$$\text{overlap} = d - 2 \cdot \text{overlap radius} \tag{5}$$

with "2" as a scaling factor and "overlap radius" is a tuning parameter, which was set to seven pixels.

Table 2 NASA Ames Build 1 Scenario Sensor Locations and Angles

Location	Latitude (°)	Longitude (°)	Alt. (m)	Heading (°)	Elev. (°)	Range (m)
1	37.429540	-122.067712	1.54	66.53	9.68	600
2	37.425155	-122.059832	1.54	246.54	29.29	600
3	37.423476	-122.061238	1.57	303.14	34.0	600
4	37.426551	-122.067970	1.52	44.78	26.31	600
O	37.426861	-122.066015	0	N/A	N/A	N/A

Different GoPro HERO10 Black camera lenses and modes will be selected and optimized to have the UAV within at least two camera fields of view throughout most of the UAV's flight path (see Figure 11), and the estimated horizontal field of view per camera is 40°. [‖]. The horizontal field of view for each GoPro HERO10 Black camera was set to 110°. The camera at ground node 2 had different settings than those at ground nodes 1, 3, & 4 since it was recording in a

[§]https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html
[¶]https://learnopencv.com/blob-detection-using-opencv-python-c/
[‖]https://community.gopro.com/s/article/HERO10-Black-Digital-Lenses-FOV-Informations?language=en_US

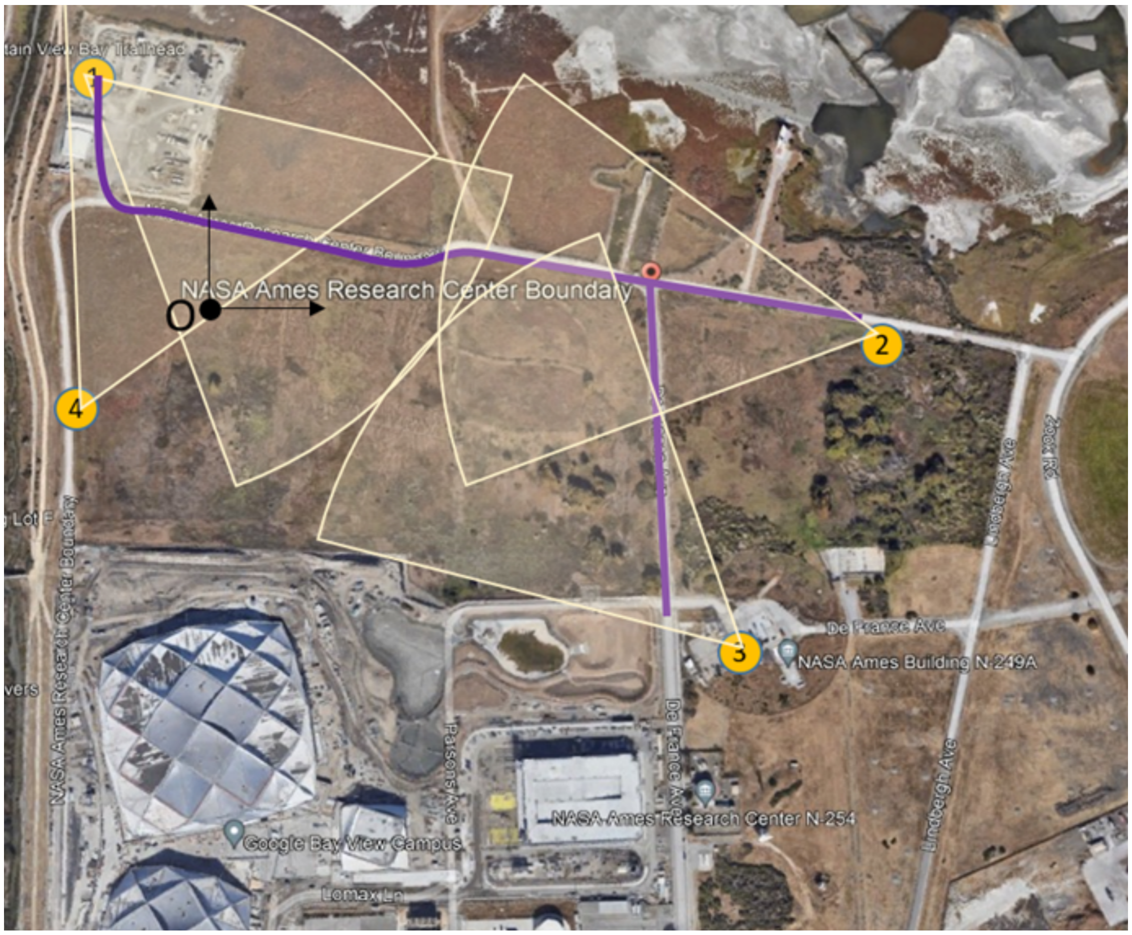

1
2
3
4
O
NASA Ames Research Center Boundary
De France Ave
NASA Ames Building N-249A
NASA Ames Research Center N-254
Google Bay View Campus
Parsons Ave
Lindbergh Ave
Lomax Ln

Fig. 11 Ames Research Center Flight Test - Build 1: sensor stations (yellow) & origin, O (black dot)

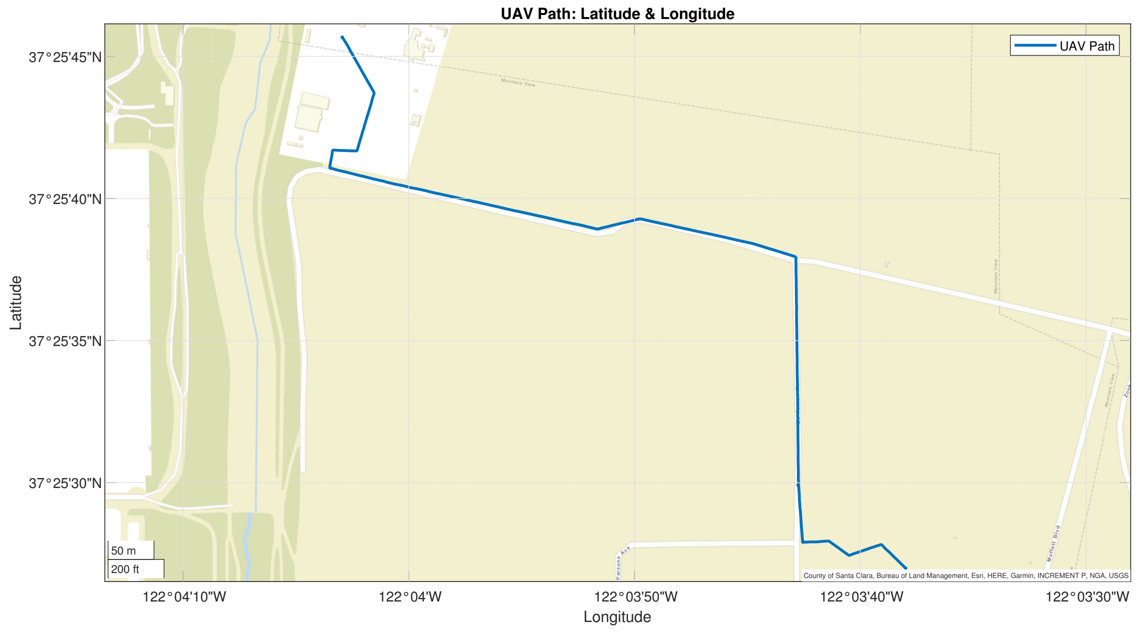


Fig. 12 UAV Path: Latitude & Longitude (thick blue line)

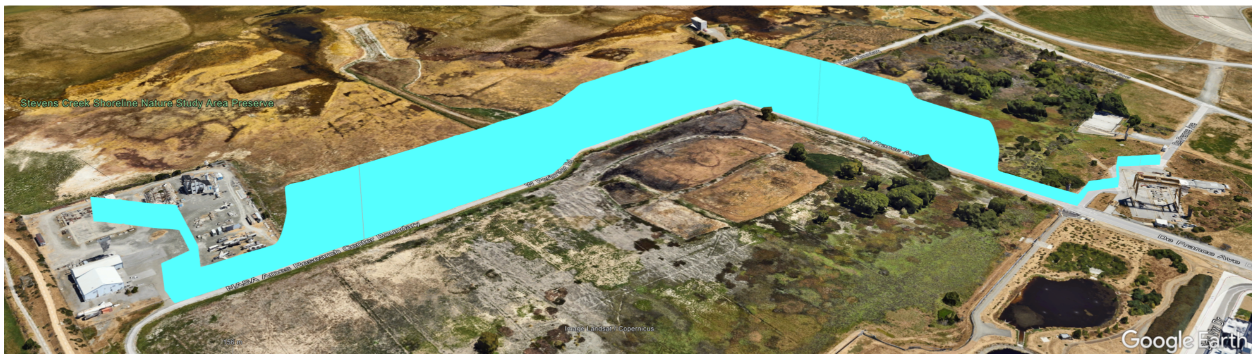


Fig. 13 UAV Path: Google Earth

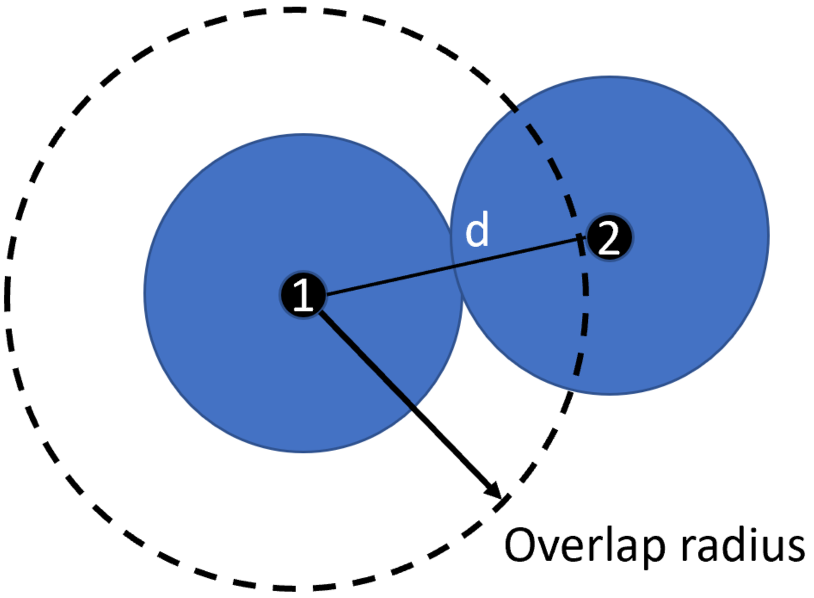


Fig. 14 Overlap Radius Diagram: "1" and "2" are blob centroid points in consecutive frames, d is the distance between the two centroid points, and overlap radius is a tuning parameter, which was set to seven pixels

different mode, which yields different resolution and frames per second values. Table 3 shows the difference in the camera specs between ground nodes 1, 3, & 4 vs. 2.

Table 3 Distributed Sensing Camera Specs for UAS Flight Test

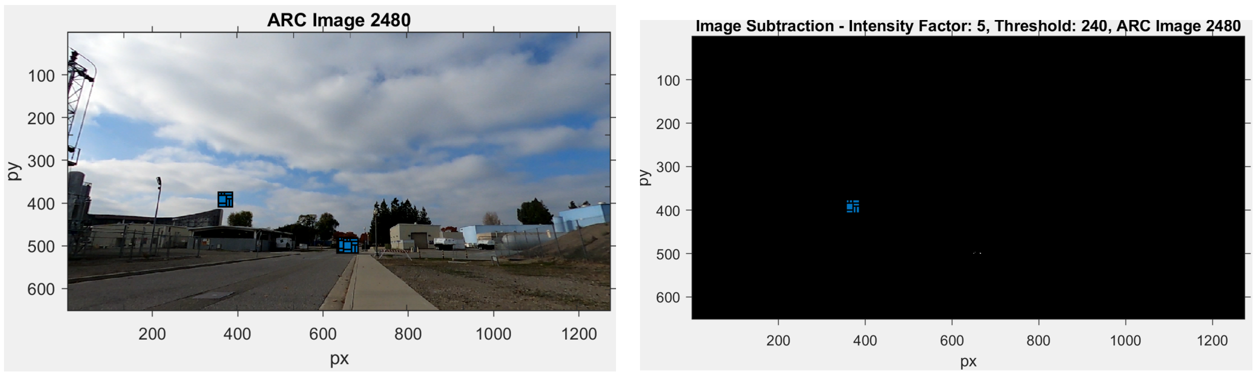
Spec	GN1,3,4	GN2
Aspect Ratio	4:3	16:9
pixel width	4000	3840
pixel height	3000	2160
FPS	60	30

V. Results

This section describes the outcomes from the post-processed GBVT experiments. The first subsection presents the findings of the preliminary test conducted with a singular camera, where the UAV traverses from the leftmost to the rightmost extent within the camera's field of view. The second subsection shows the outcomes from the UAV flight test with a ground-based distributed sensing network of four distinct cameras.

A. Preliminary Test with One Camera

A preliminary distributed sensing test involved using a ground-based, fixed-angle camera to acquire flight test data to evaluate the GBVT. Figure 15 illustrates the intermediate steps of image subtraction, depicting the original and masked images after applying the subtraction process. Figure 15a displays the current image, while Figure 15b exhibits the subtracted image following the steps outlined in Figure 2. Setting the intensity factor to 5 and the threshold pixel value to 240 ensures the elimination of all pixels below this threshold. Preliminary tests of the GBVT in MATLAB involve

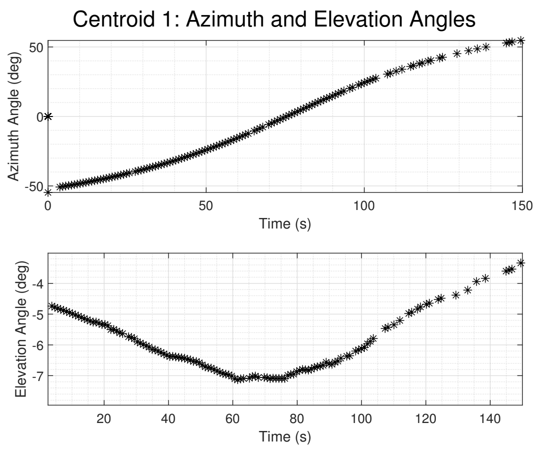


(a) Original Image **(b) Mask after Image Subtraction**

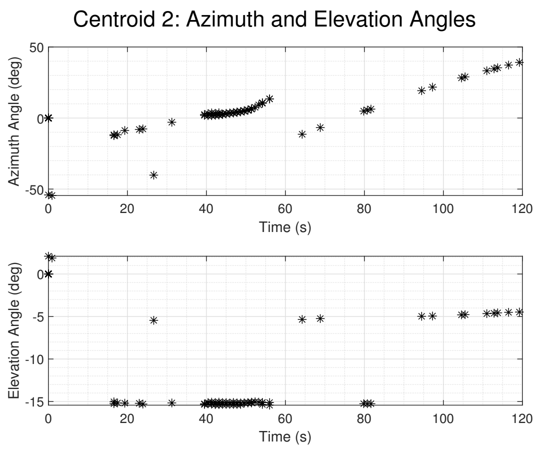
Fig. 15 Results of Image Subtraction: Frame 2480

tracking five objects using image subtraction and blob detection (bwareafilt) **. Objects (centroids) 2-4, exhibiting relatively low and flat elevation angles between approximately 40 and 55 seconds, are likely to be cars, corresponding to the observed vehicles in the video (see Figure 16). The fifth detection displays random outliers, so Figure 16 does not show it for brevity. The average runtime per iteration is 0.2675 seconds, showcasing near real-time capabilities. With the video recorded at 60 frames per second, the frame subtraction step involves 55 frames, allowing sufficient time and difference for detecting UAV motion. It is possible to reduce frame subtraction to 30 frames (0.5 seconds for a 60 fps camera); however, this leads to multiple false positives. Figure 17 presents the azimuth and elevation angles of the tracked UAV for the OpenCV C++ implementation. The OpenCV-tracked UAV azimuth and elevation angles closely resemble the MATLAB implementation from Figure 16 but lack the initial false positives. The CGBVT exhibits smoother results and avoids missing detections, as observed in the gaps in the azimuth and elevation angles for Centroid 1

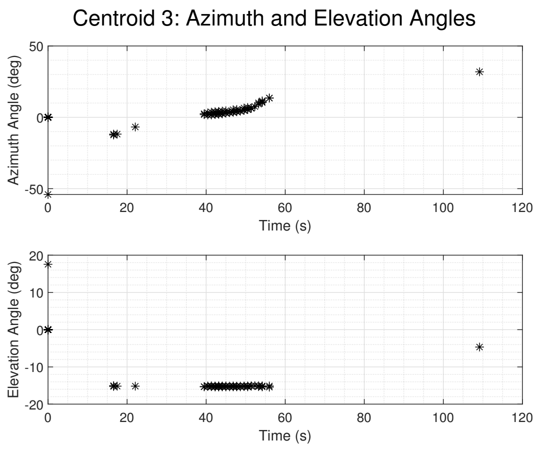
**https://www.mathworks.com/help/images/ref/bwareafilt.html



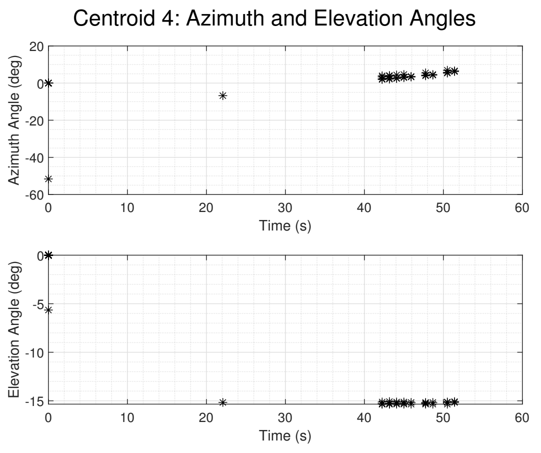
(a) Object 1 (UAV)



(b) Object 2 (car)



(c) Object 3 (car)



(d) Object 4 (car)

Fig. 16 Azimuth and Elevation Angles per Detection: MATLAB Image Tracker for Single-Camera Case - centroid 1 is the UAV, and the others are cars that appear/hide behind trees and buildings

in Figure 16a. OpenCV's background subtraction function notably facilitates learning the background model, enhancing tracking performance by removing static objects. In contrast, the MATLAB implementation lacks background learning, resulting in less consistent tracking, mainly when the UAV appears smaller towards the end. The subsequent step for GBVT involves its execution with multiple cameras (videos), leveraging the OpenCV implementation. Key takeaways

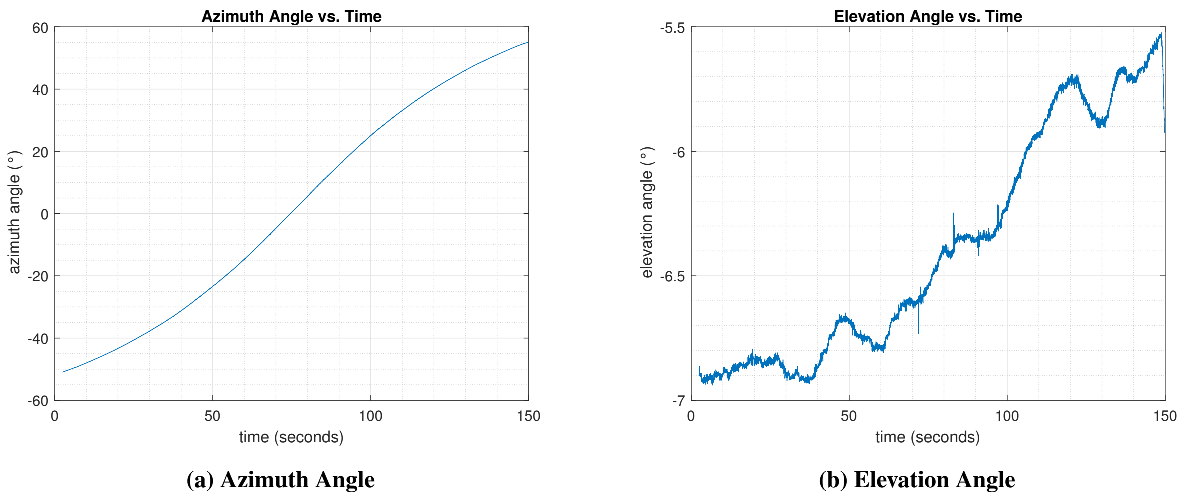


Fig. 17 Tracked UAV Azimuth and Elevation Angles: CGBVT

from this preliminary GBVT test include:

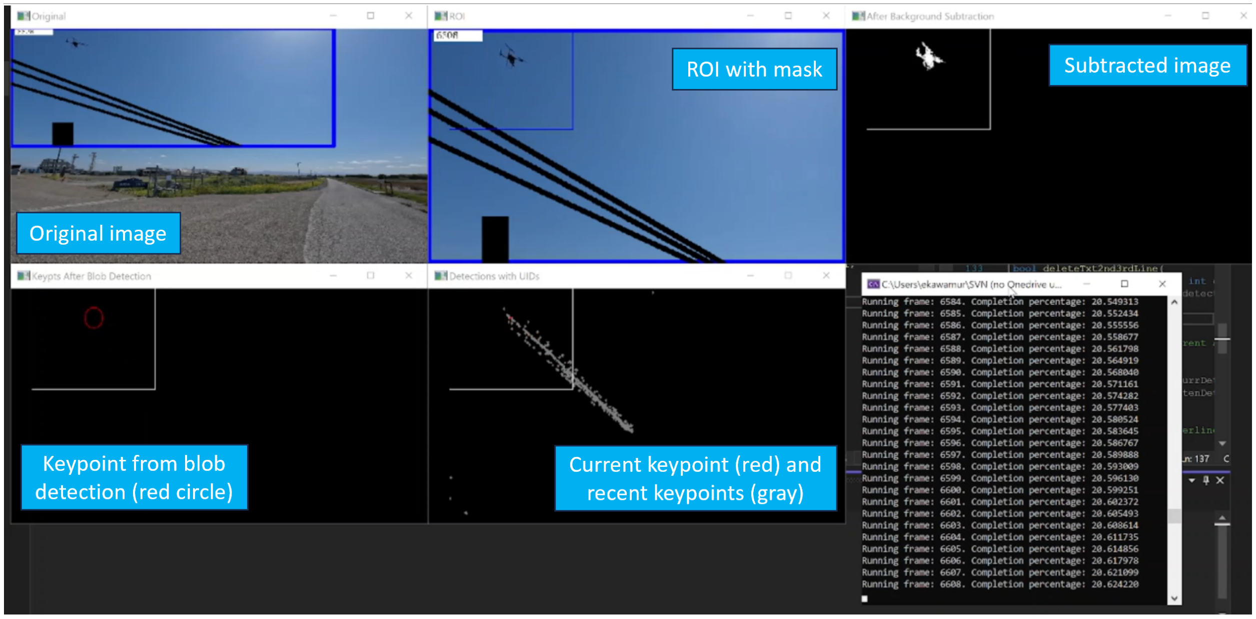
1) Image subtraction effectively removes several static objects, and its combination with blob detection enables multi-object tracking.
2) Objects exhibiting slight movements appearing in the subtracted image emphasize the importance of applying a threshold to eliminate less intense pixels (slight movements like jitter, shaking, etc.) or objects.
3) The incorporation of a region of interest (ROI) allows the GBVT to focus on desired targets by narrowing the search area.
4) The use of masks effectively removes undesired regions where targets are unlikely to be found, such as grass, buildings, and trees.
5) Implementing a purely computer vision tracking method demonstrates promising results, suggesting that incorporating Kalman filters may be unnecessary. However, including estimated object kinematics may enhance tracking performance and object classification, particularly considering the varying flight patterns between birds and quadcopter UAVs, where birds typically exhibit faster and more sporadic movements.

B. Distributed Sensing Test with Four Cameras

This subsection shows the distributed sensing results for the four cameras from Figure 11. The first subsubsection shows the GBVT image processing pipeline, and the second subsubsection describes the tracking performance metrics and definitions.

1. GBVT Image Processing Pipeline

Figure 18 shows a screenshot that details the image tracker's image processing pipeline, following the sequential operations outlined in Fig. 4. The calibration phase lasts for 150 frames for ground nodes 1-3. The CGBVT for ground node 2 uses only 10 frames in its calibration phase because CGBVT tests the features for pausing the video and redrawing the ROI, which provides quick learning for the background model. This figure clarifies the procedural sequence of the GBVT pipeline. The upper-leftmost panel displays the unaltered frame, with black masks excluding undesirable regions and a rectangular delineation designating the region of interest (ROI). The adjacent right panel highlights tracking efficacy within the ROI, indicating active tracking with a conspicuous blue bounding box encapsulating the UAV. The subsequent upper-right panel exhibits the resulting image after background subtraction, eliminating pixels corresponding to static objects and leaving only the UAV (depicted in white). The lower-leftmost panel incorporates a red circle around the latest detection (keypoint) and an encompassing bounding box. The final panel displays multiple white dots, representing previous detections of the object, with the red dot designating the presently tracked object, including its bounding box. Finally, the command prompt in the lower-right corner provides information about the current frame, including the completion percentage computed based on the current frame number and the remaining frames.


Original
ROI
After Background Subtraction
ROI with mask
Subtracted image
Original image
Keypts After Blob Detection
Detections with UIDs
Keypoint from blob detection (red circle)
Current keypoint (red) and recent keypoints (gray)
C:\Users\ekawamur\SVN (no Onedrive u...

```
Running frame: 6584. Completion percentage: 20.549313
Running frame: 6585. Completion percentage: 20.552434
Running frame: 6586. Completion percentage: 20.555556
Running frame: 6587. Completion percentage: 20.558677
Running frame: 6588. Completion percentage: 20.561798
Running frame: 6589. Completion percentage: 20.564919
Running frame: 6590. Completion percentage: 20.568040
Running frame: 6591. Completion percentage: 20.571161
Running frame: 6592. Completion percentage: 20.574282
Running frame: 6593. Completion percentage: 20.577403
Running frame: 6594. Completion percentage: 20.580524
Running frame: 6595. Completion percentage: 20.583645
Running frame: 6596. Completion percentage: 20.586767
Running frame: 6597. Completion percentage: 20.589888
Running frame: 6598. Completion percentage: 20.593009
Running frame: 6599. Completion percentage: 20.596130
Running frame: 6600. Completion percentage: 20.599251
Running frame: 6601. Completion percentage: 20.602372
Running frame: 6602. Completion percentage: 20.605493
Running frame: 6603. Completion percentage: 20.608614
Running frame: 6604. Completion percentage: 20.611735
Running frame: 6605. Completion percentage: 20.614856
Running frame: 6606. Completion percentage: 20.617978
Running frame: 6607. Completion percentage: 20.621099
Running frame: 6608. Completion percentage: 20.624220
```

Fig. 18 Image Tracker Screenshot: Image Processing Pipeline for Ground Node 1 (6/26/23 flight test) - the black rectangle and lines remove the street light and powerlines to help remove false positives

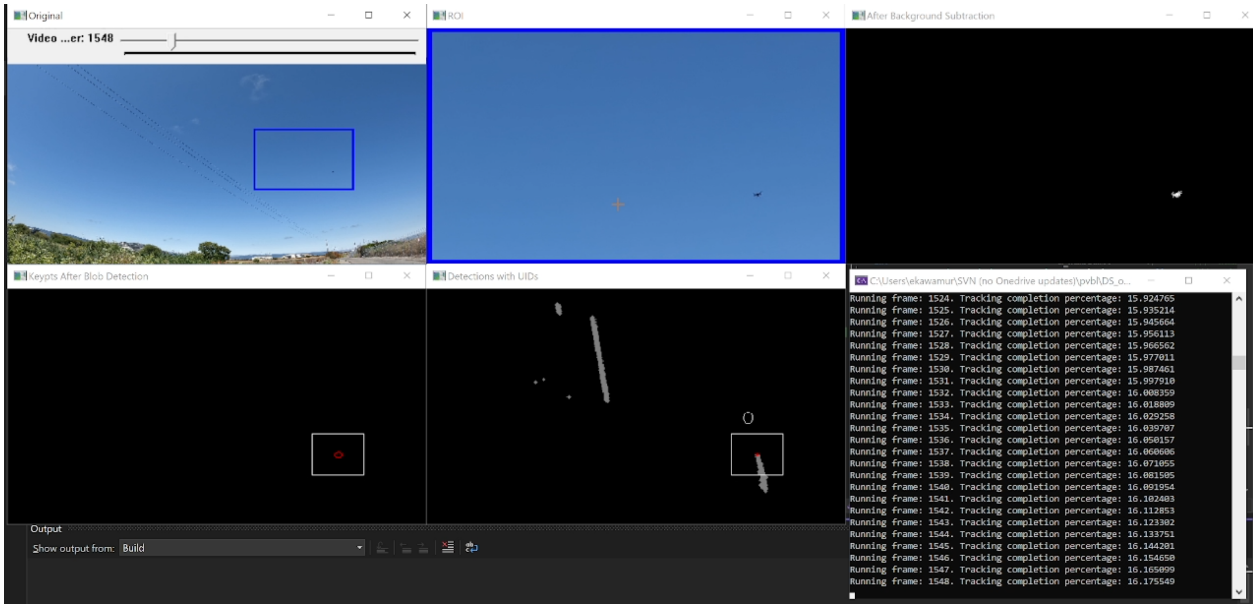
Figure 19 shows a screenshot of the GBVT image processing pipeline and tests the GBVT's tracking capabilities by pausing the video and redrawing the ROI to reduce the number of false positives and negatives. The detection precision and recall metrics shown later will provide more details about how this pause-and-redraw-ROI tracking strategy performs better than the standard method with one ROI that encompasses the entire UAV trajectory.

Figure 20 shows a screenshot of the GBVT image processing pipeline as the UAV approaches the end of its trajectory and heads towards the landing zone. The camera at GN3 captures the last portion of the flight with a few birds and some masks to remove undesired areas such as grass and trees. As the UAV approached the landing zone, it appeared larger and had several blobs instead of only one. Figure 21 illustrates the representation of the UAV as a claw, with two motors resembling outer fingers and an inner finger located near the UAV's center of mass. Given the smaller size of the motors, their blob radii are diminished compared to the center blob (middle finger of the claw). Future investigations could explore methodologies such as selecting the largest blob or centroid while disregarding smaller neighboring blobs. Alternatively, dilating the detected blobs to merge them before determining the centroid of the blob represents another avenue for consideration.

Figure 22 shows a screenshot of the GBVT's image processing pipeline at ground node 4, with many birds in the scenery occasionally overlapping with the UAV. The GBVT has certain limitations when it comes to overlapping objects. An occurrence of note is the potential for labeling issues when tracked objects traverse the same path, specifically when intersecting. For example, in Figure 23, the image tracker assigns label 106 to two distinct objects: a bird (left) and a UAV (right). This labeling discrepancy arises as both entities traverse or intersect at a point along the path identified as 106.

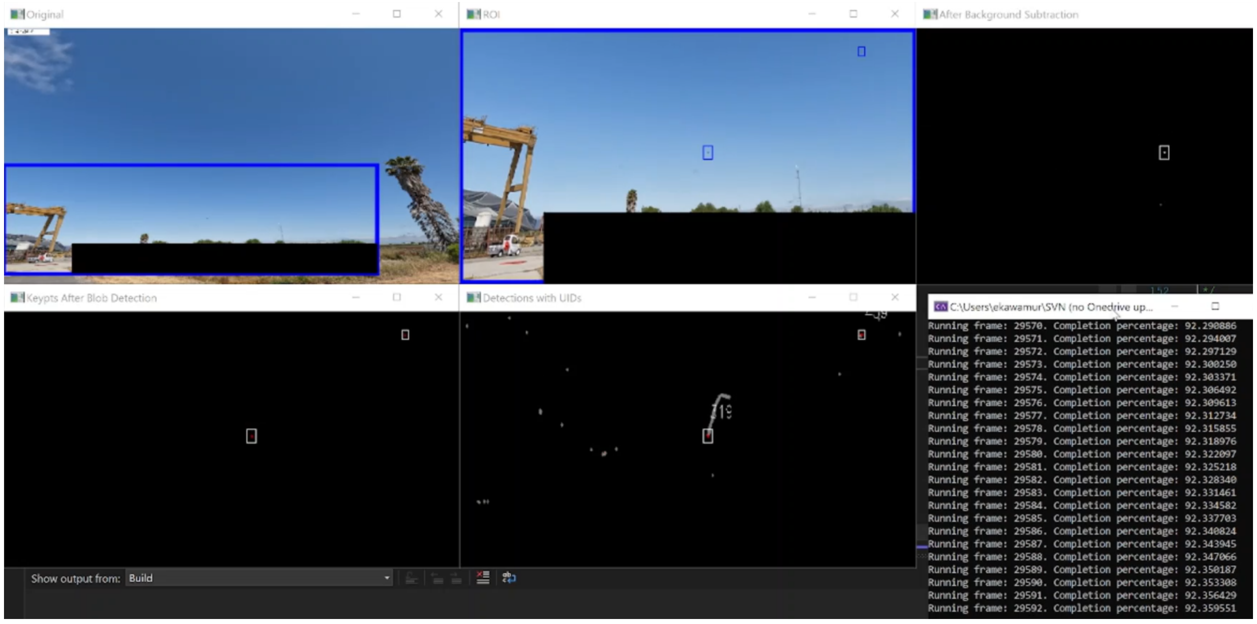
Potential avenues for future research encompass investigating and mitigating the overlap problem. Introducing artificial intelligence or machine learning methodologies may offer a solution by classifying objects based on size, speed, and direction/heading. Alternative approaches may assess the probability of object A retaining its identity compared to a newly detected object.

Another notable phenomenon pertains to the labeling of objects that overlap with one another. As depicted in Figure 24, the image tracker adjusts the UAV's identifier from 106 to 219 due to a path overlapping with a bird (identified by 219). The current image tracker methodology adheres exclusively to computer vision and image processing techniques. However, future investigations may contemplate incorporating a Kalman filter to predict the tracked target's next position


Original
Video ...er: 1548
ROI
After Background Subtraction
Keypts After Blob Detection
Detections with UIDs
0
Output
Show output from: Build
C:\Users\ekawamur\SVN (no Onedrive updates)\pvbl\DS_o...

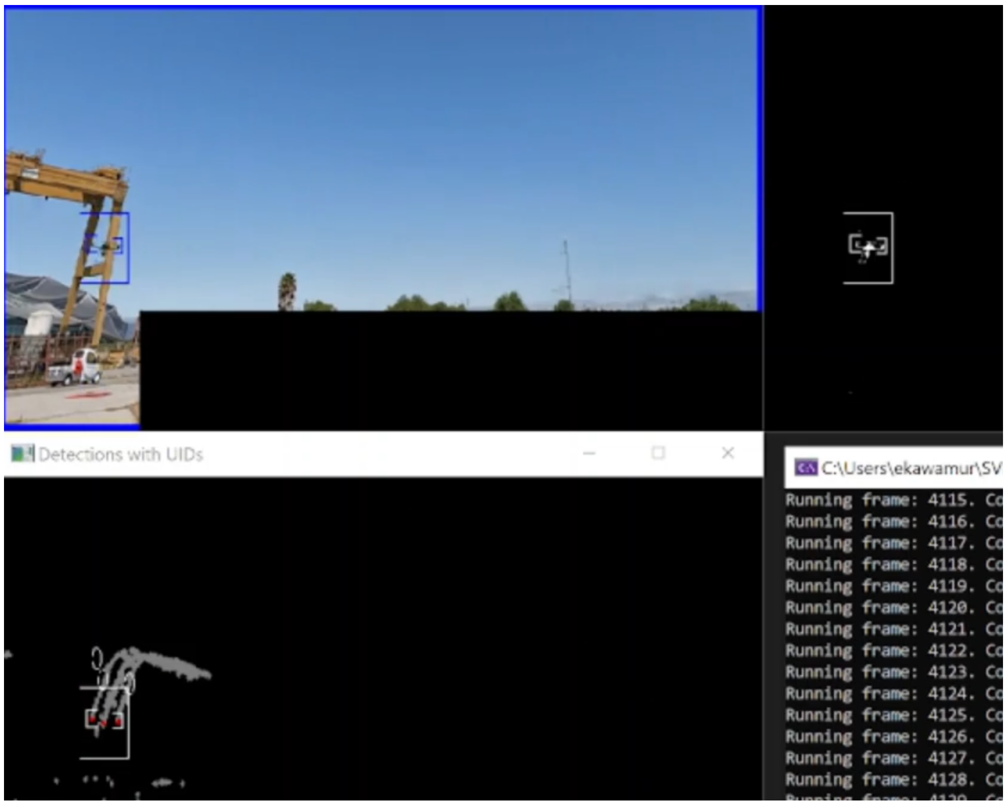
```
Running frame: 1524. Tracking completion percentage: 15.924765
Running frame: 1525. Tracking completion percentage: 15.935214
Running frame: 1526. Tracking completion percentage: 15.945664
Running frame: 1527. Tracking completion percentage: 15.956113
Running frame: 1528. Tracking completion percentage: 15.966562
Running frame: 1529. Tracking completion percentage: 15.977011
Running frame: 1530. Tracking completion percentage: 15.987461
Running frame: 1531. Tracking completion percentage: 15.997910
Running frame: 1532. Tracking completion percentage: 16.008359
Running frame: 1533. Tracking completion percentage: 16.018809
Running frame: 1534. Tracking completion percentage: 16.029258
Running frame: 1535. Tracking completion percentage: 16.039707
Running frame: 1536. Tracking completion percentage: 16.050157
Running frame: 1537. Tracking completion percentage: 16.060606
Running frame: 1538. Tracking completion percentage: 16.071055
Running frame: 1539. Tracking completion percentage: 16.081505
Running frame: 1540. Tracking completion percentage: 16.091954
Running frame: 1541. Tracking completion percentage: 16.102403
Running frame: 1542. Tracking completion percentage: 16.112853
Running frame: 1543. Tracking completion percentage: 16.123302
Running frame: 1544. Tracking completion percentage: 16.133751
Running frame: 1545. Tracking completion percentage: 16.144201
Running frame: 1546. Tracking completion percentage: 16.154650
Running frame: 1547. Tracking completion percentage: 16.165099
Running frame: 1548. Tracking completion percentage: 16.175549
```

Fig. 19 Image Tracker Screenshot: Image Processing Pipeline for Ground Node 2 (6/26/23 flight test) - testing the GBVT's ROI and tracking capabilities by pausing and redrawing the ROI to follow the UAV throughout its trajectory, which eliminates false positives and negatives; future work can include a function to automatically redraw ROIs when the object is nearing the end of the ROI


Original
ROI
After Background Subtraction
Keypts After Blob Detection
Detections with UIDs
319
Show output from: Build
C:\Users\ekawamur\SVN (no Onedrive up...

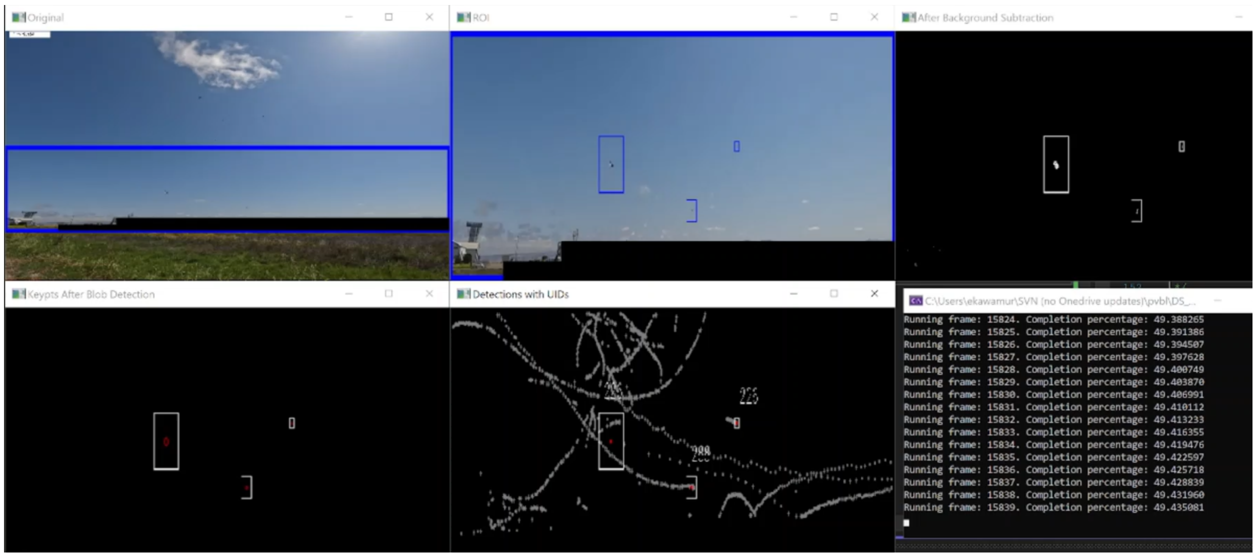
```
Running frame: 29570. Completion percentage: 92.290886
Running frame: 29571. Completion percentage: 92.294007
Running frame: 29572. Completion percentage: 92.297129
Running frame: 29573. Completion percentage: 92.300250
Running frame: 29574. Completion percentage: 92.303371
Running frame: 29575. Completion percentage: 92.306492
Running frame: 29576. Completion percentage: 92.309613
Running frame: 29577. Completion percentage: 92.312734
Running frame: 29578. Completion percentage: 92.315855
Running frame: 29579. Completion percentage: 92.318976
Running frame: 29580. Completion percentage: 92.322097
Running frame: 29581. Completion percentage: 92.325218
Running frame: 29582. Completion percentage: 92.328340
Running frame: 29583. Completion percentage: 92.331461
Running frame: 29584. Completion percentage: 92.334582
Running frame: 29585. Completion percentage: 92.337703
Running frame: 29586. Completion percentage: 92.340824
Running frame: 29587. Completion percentage: 92.343945
Running frame: 29588. Completion percentage: 92.347066
Running frame: 29589. Completion percentage: 92.350187
Running frame: 29590. Completion percentage: 92.353308
Running frame: 29591. Completion percentage: 92.356429
Running frame: 29592. Completion percentage: 92.359551
```

Fig. 20 Image Tracker Screenshot: Image Processing Pipeline for Ground Node 3 (6/26/23 flight test) - the UAV (319) heads towards the landing zone at ground node 3 (orange mat in the bottom left); for clarity, the black box in the top row's middle window is the same mask in the first window in the top left



```
C:\Users\ekawamur\SV
Running frame: 4115. Co
Running frame: 4116. Co
Running frame: 4117. Co
Running frame: 4118. Co
Running frame: 4119. Co
Running frame: 4120. Co
Running frame: 4121. Co
Running frame: 4122. Co
Running frame: 4123. Co
Running frame: 4124. Co
Running frame: 4125. Co
Running frame: 4126. Co
Running frame: 4127. Co
Running frame: 4128. Co
```

Fig. 21 Image Tracker: "Claw Phenomenon" for Ground Node 3's Camera - tracking subcomponents of the target as distinct objects



```
C:\Users\ekawamur\SVN (no Onedrive updates)\pvbl\DS_...
Running frame: 15824. Completion percentage: 49.388265
Running frame: 15825. Completion percentage: 49.391386
Running frame: 15826. Completion percentage: 49.394507
Running frame: 15827. Completion percentage: 49.397628
Running frame: 15828. Completion percentage: 49.400749
Running frame: 15829. Completion percentage: 49.403870
Running frame: 15830. Completion percentage: 49.406991
Running frame: 15831. Completion percentage: 49.410112
Running frame: 15832. Completion percentage: 49.413233
Running frame: 15833. Completion percentage: 49.416355
Running frame: 15834. Completion percentage: 49.419476
Running frame: 15835. Completion percentage: 49.422597
Running frame: 15836. Completion percentage: 49.425718
Running frame: 15837. Completion percentage: 49.428839
Running frame: 15838. Completion percentage: 49.431960
Running frame: 15839. Completion percentage: 49.435081
```

Fig. 22 Image Tracker Screenshot: Image Processing Pipeline for Ground Node 4 (6/26/23 flight test) - several birds confuse the GBVT, making it difficult to detect the UAV (226) and process the pixel coordinates and azimuth & elevation angles, especially during overlaps or "collisions"

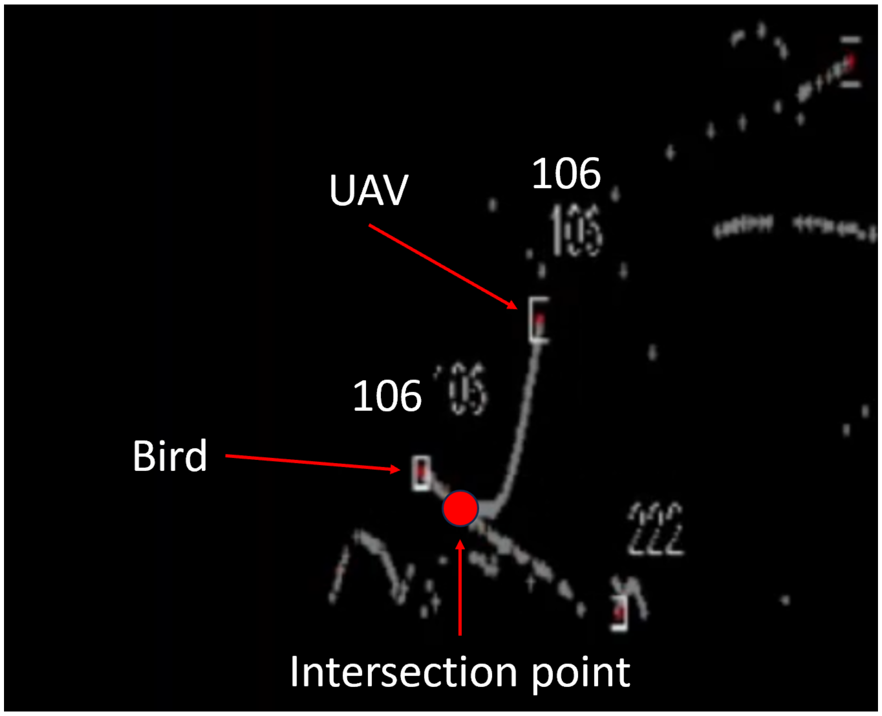
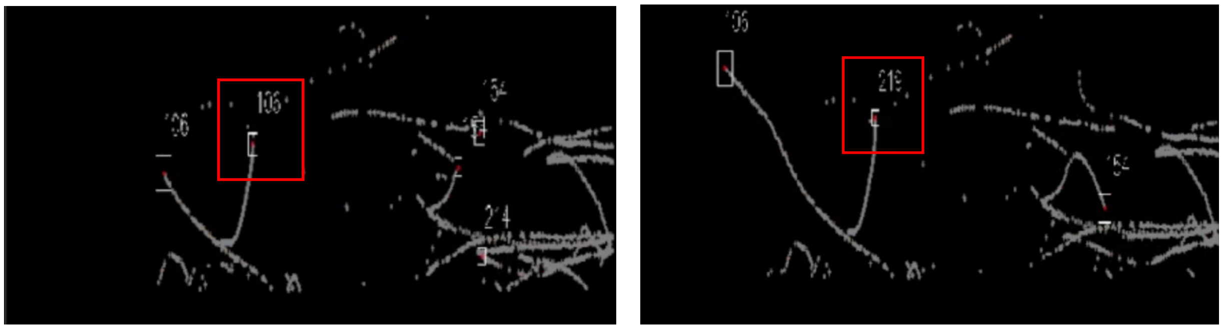


Fig. 23 Image Tracker: Same Label (106) for Two Objects - Label 106 enhances clarity, depicting a bird (left) and UAV (right); IDs are not unique for overlapping objects

based on the current state and kinematics. Implementing this type of Kalman filter could estimate the state of a detected object in subsequent iterations, thereby preventing the assignment of identical identifiers to overlapping objects.



(a) UAV labeled with ID#: 106 **(b) UAV labeled with ID#: 219**

Fig. 24 Image Tracker Label Change During Overlapping Paths at Ground Node 4

2. Detector Precision and Recall

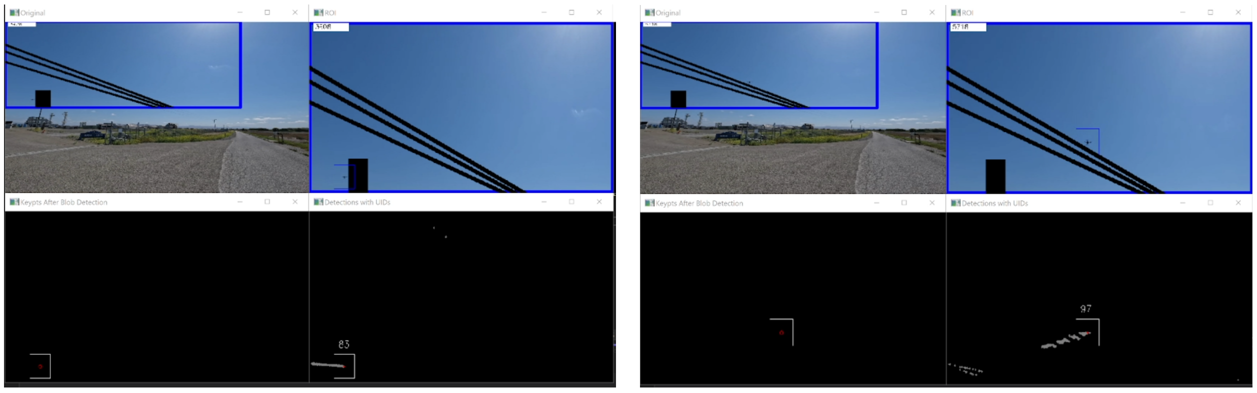
Reference [24] offers comprehensive performance metrics and values concerning the detector's precision and recall. The definitions and performance metrics employed by the GBVT align with those provided in the reference above. True positives (TP) denote the number of drone detections (centroids) located within 100 pixels of the ground truth, false positives (FP) signify the number of detections (centroids) corresponding to clouds, birds, aircraft, and other objects within a 100-pixel proximity to the ground truth, and false negatives (FN) represent the instances of missed drone detections. Notably, one pixel corresponds to an angular resolution of 0.05 degrees, resulting in a 100-pixel radius equivalent to 5 degrees. The detector precision (DP) and detector recall (DR) are defined as follows:

$$DP = \frac{TP}{TP + FP}\ , \ DR = \frac{TP}{TP + FN} \tag{6}$$

Each video encapsulates up to 8 minutes and 54 seconds of footage. Ground nodes 1 and 4's videos capture the UAV throughout one video, while ground nodes 2 and 3's videos span two segments. Table 4 enumerates the detection precision and recall values for ground node 1, demonstrating notable accuracy. Ground node 1 exhibits high precision values, primarily due to false positives (birds) scarcity. However, the occurrence of false negatives, i.e., missed detections, is attributed to two primary factors: 1) the presence of masks over powerlines and a streetlight (refer to Figure 25) and 2) the UAV appears very small when it is far away from the camera. Ground node 2 exhibits a slight

Table 4 Detector Precision and Recall for Ground Node 1's Camera

ID	True Positives (TP)	False Positives (FP)	False Negatives (FN)	Detection Precision (DP)	Detetion Recall (DR)
83	11674	0	8	100%	99.93%
91	921	34	11	96.44%	98.82%
97	1243	224	93	84.73%	93.04%
139	1504	40	419	97.41%	78.21%
165	4898	86	1580	98.27%	75.61%



(a) Ground Node 1: ID 83 **(b) Ground Node 1: ID 97**

Fig. 25 Ground Node 1: ROI and Masks - Mask placement increases the number of false negatives, i.e., missed detections

camera shift at the commencement of the initial video, resulting in false positives, false negatives, and various other detections within the scenery, such as powerlines, trees, and grass. Post-processing and cleaning of the raw data contribute to an increased tracking duration for the UAV. Several false negatives persist, as indicated in Table 5. After the camera stabilizes, most detection precision and recall values exceed 90%. ID 16, however, demonstrates a lower detection precision value at 71.13% due to birds flying within 100 pixels of the UAV. The GBVT's methodology of pausing the video and redrawing the ROI for tracking a small search area yields accurate results. Future work may contemplate implementing a dynamic ROI or bounding box that dynamically follows the tracked target to enhance tracking performance regarding detection precision and recall. Table 6 provides the detection precision and recall values for ground node 3's camera, capturing the UAV's final trajectory and landing segment. The initial stages depict the UAV as diminutive, rendering detection challenging and inconsistent. Consequently, the UAV's ID (319) in the first video assumes a substantial value. Furthermore, a multitude of false negatives leads to a diminished DR value. As the UAV approaches the landing zone, it appears larger, enabling the tracker to detect the UAV's arms, motors, or propellers. However, the GBVT treats these components as separate detections, culminating in more false positives, even though

Table 5 Detector Precision and Recall for Ground Node 2's Camera (V1 = 1st video & V2 = 2nd video, subscript is the ID number)

ID	True Positives (TP)	False Positives (FP)	False Negatives (FN)	Detection Precision (DP)	Detetion Recall (DR)
$V1_0$	339	563	368	37.58%	47.95%
$V1_4$	2898	274	1159	91.36%	71.43%
$V1_{16}$	101	41	3	71.13%	97.12%
$V2_0$	3266	132	132	96.12%	96.12%
$V2_{58}$	924	65	22	93.43%	97.67%
$V2_{74}$	783	5	9	99.37%	98.86%
$V2_{94}$	965	23	30	97.67%	96.98%

they do not correspond to birds, powerlines, or other aircraft. Fortunately, as the UAV nears the landing zone, the arms, motors, or propeller detections deviate farther from the UAV's center, surpassing the 100-pixel radius specified in the false positives definition. Consequently, the count of false positives for detection precision lacks these detections. Table

Table 6 Detector Precision and Recall for Ground Node 3's Camera (V1 = 1st video & V2 = 2nd video, subscript is the ID number)

ID	True Positives (TP)	False Positives (FP)	False Negatives (FN)	Detection Precision (DP)	Detetion Recall (DR)
$V1_{319}$	6590	221	7091	96.76%	48.17%
$V2_0$	4121	1234	461	76.96%	89.94%

7 outlines the detection precision and recall values for ground node 4's camera. Regrettably, numerous false negatives ensue due to the UAV's diminutive size, rendering it a smaller blob than the birds. Ground node 4's camera, hosting the most birds and overlapping objects among all cameras, encounters heightened complexity. Consequently, it features an extensive array of ID numbers. Noteworthy segments where the GBVT achieves commendable precision for the UAV with percentages exceeding 80% occur at IDs 106, 305, and 393. Despite the prevalence of numerous birds flying near the UAV and overlapping its path, resulting in diminished detection precision and recall, there are accurate recall values with percentages higher than 80% at IDs: 219 and 226. A dynamic ROI or bounding box can enhance tracking performance, as shown in Table 5.

Table 7 Detector Precision and Recall for Ground Node 4's Camera

ID	True Positives (TP)	False Positives (FP)	False Negatives (FN)	Detection Precision (DP)	Detetion Recall (DR)
106	4283	632	1577	87.14%	73.09%
219	1715	647	5	72.61%	99.71%
226	2948	1373	577	68.22%	83.63%
305	722	44	1277	94.26%	36.12%
315	172	61	2532	73.82%	6.36%
393	274	35	1421	88.67%	16.17%
444	1426	707	695	66.85%	67.23%

VI. Conclusion

The UAV flight test videos present auspicious outcomes in the context of ground-based distributed sensing within the airspace, particularly in the tracking and monitoring of AAM/UAM aircraft, especially in scenarios such as corridor surveillance. Prospective AAM/UAM operations necessitate the integration of ground-based and airborne sensors to effectively monitor the airspace.

Under the assumption of precise camera calibration, future work may involve the incorporation of an undistortion step within the pre-processing pipeline. This augmentation serves to enhance the system's robustness, thereby refining the accuracy of azimuth and elevation angles pertaining to the target before transmission to the EKF embedded within the distributed sensor system. Other future endeavors involve distinguishing different objects during overlapping segments, clustering nearby blobs together to remove claw effects in the UAV and its motors/propellers, and implementing a dynamic ROI that moves with the target. The ultimate objective of these enhancements is to bolster the precision of estimations regarding the target's states.

Subsequent phases of experimentation involving UAS flight tests, coupled with the real-time implementation of the GBVT, constitute essential strides toward the preliminary development of an AAM/UAM distributed sensing architecture. This architecture aims to mirror characteristics akin to those observed in ATC/ATM systems.

Acknowledgments

The authors would like to thank the Transformative Tools and Technologies (TTT) Project under the NASA Aeronautics Research Mission Directorate (ARMD) for funding this research. The authors would also like to thank Loc Tran (NASA Langley Research Center) for his insight and expertise in image processing and computer vision.

References

[1] Ippolito, C. A., Hashemi, K., Kawamura, E., Gorospe, G., Holforty, W., Kannan, K., Stepanyan, V., Lombaerts, T., Brown, N., Jaffe, A., et al., "Concepts for Distributed Sensing and Collaborative Airspace Autonomy in Advanced Urban Air Mobility," *AIAA SciTech 2023 Forum*, 2023, p. 20.

[2] Poitevin, P., Pelletier, M., and Lamontagne, P., "Challenges in detecting UAS with radar," *2017 International Carnahan Conference on Security Technology (ICCST)*, IEEE, 2017, pp. 1–6.

[3] Güvenç, I., Ozdemir, O., Yapici, Y., Mehrpouyan, H., and Matolak, D., "Detection, localization, and tracking of unauthorized UAS and jammers," *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, IEEE, 2017, pp. 1–10.

[4] Accardo, D., Fasano, G., Forlenza, L., Moccia, A., and Rispoli, A., "Flight test of a radar-based tracking system for UAS sense and avoid," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 49, No. 2, 2013, pp. 1139–1160.

[5] Martinez, M., "UAS detection, classification, and tracking in urban terrain," *2019 IEEE Radar Conference (RadarConf)*, IEEE, 2019, pp. 1–6.

[6] Samaras, S., Magoulianitis, V., Dimou, A., Zarpalas, D., and Daras, P., "UAV classification with deep learning using surveillance radar data," *International Conference on Computer Vision Systems*, Springer, 2019, pp. 744–753.

[7] Mohajerin, N., Histon, J., Dizaji, R., and Waslander, S. L., "Feature extraction and radar track classification for detecting UAVs in civillian airspace," *2014 IEEE Radar Conference*, IEEE, 2014, pp. 0674–0679.

[8] Fu, C., Carrio, A., Olivares-Mendez, M. A., Suarez-Fernandez, R., and Campoy, P., "Robust real-time vision-based aircraft tracking from unmanned aerial vehicles," *2014 ieee international conference on robotics and automation (ICRA)*, IEEE, 2014, pp. 5441–5446. https://doi.org/10.1109/ICRA.2014.6907659.

[9] Sattigeri, R., Johnson, E., Calise, A., and Ha, J., "Vision-based target tracking with adaptive target state estimator," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6828. https://doi.org/10.2514/6.2007-6828.

[10] Sivaraman, S., and Trivedi, M. M., "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE transactions on intelligent transportation systems*, Vol. 14, No. 4, 2013, pp. 1773–1795. https://doi.org/10.1109/TITS.2013.2266661.

[11] Toyama, K., and Hager, G. D., "Incremental focus of attention for robust vision-based tracking," *International Journal of Computer Vision*, Vol. 35, 1999, pp. 45–63. https://doi.org/10.1023/A:1008159011682.

[12] Joo, S., Ippolito, C., Al-Ali, K., and Yeh, Y.-H., "Vision aided inertial navigation with measurement delay for fixed-wing unmanned aerial vehicle landing," *2008 IEEE aerospace conference*, IEEE, 2008, pp. 1–9. https://doi.org/10.1109/AERO.2008.4526557.

[13] Bharati, S. P., Nandi, S., Wu, Y., Sui, Y., and Wang, G., "Fast and robust object tracking with adaptive detection," *2016 IEEE 28th international conference on tools with artificial intelligence (ICTAI)*, IEEE, 2016, pp. 706–713. https://doi.org/10.1109/ICTAI.2016.0112.

[14] Ganti, S. R., and Kim, Y., "Implementation of detection and tracking mechanism for small UAS," *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2016, pp. 1254–1260.

[15] Iswanto, I. A., and Li, B., "Visual object tracking based on mean-shift and particle-Kalman filter," *Procedia computer science*, Vol. 116, 2017, pp. 587–595. https://doi.org/10.1016/j.procs.2017.10.010.

[16] The Edge Company, "BCMS Ventur Bird Drone Detection System," Mare Nostrum Communicazione, 2021. URL https://www.theedgecompany.net/bcms-ventur-bird-drone-detection-system/.

[17] Lombaerts, T., Kawamura, E., Kannan, C., Keerthanaand Dolph, Stepanyan, V., Gorospe, G., and Ippolito, C. A., "Development and Field Test Results of Distributed Ground Sensor Fusion Based Object Tracking," *AIAA SciTech 2024 Forum*, 2024.

[18] Stepanyan, V., Ippolito, C., Kannan, K., Kawamura, E., Lombaerts, T., Holforty, W., and Dolph, C., "Flight Test Design and Implementation for Airspace Independent Surveillance through a Distributed Ground Based Sensor Network," *AIAA SciTech 2024 Forum*, 2024.

[19] Kawamura, E., Dolph, C., Kannan, K., Lombaerts, T., and Ippolito, C. A., "Simulated Vision-based Approach and Landing System Advanced Air Mobility," *AIAA SciTech 2023 Forum*, AIAA-2023-2195, 2023. https://doi.org/10.2514/6.2023-2195.

[20] Russell, S. J., and Norvig, P., *Artificial Intelligence A Modern Approach*, 3rd ed., Pearson Education, Inc., 2010.

[21] Lombaerts, T., Kannan, K., Kawamura, E., Dolph, C., Stepanyan, V., George, G., and Ippolito, C., "Distributed Ground Sensor Fusion Based Object Tracking for Autonomous Advanced Air Mobility Operations," *AIAA SciTech 2023 Forum*, AIAA-2023-0896, 2023. https://doi.org/10.2514/6.2023-0896.

[22] Kannan, K., Baculi, J., Lombaerts, T., Kawamura, E., Gorospe, G., Holforty, W., Ippolito, C., Stepanyan, V., Dolph, C., and Brown, N., "A Simulation Architecture for Air Traffic Over Urban Environments Supporting Autonomy Research in Advanced Air Mobility," *AIAA SciTech 2023 Forum*, AIAA-2023-0895, 2023. https://doi.org/10.2514/6.2023-0895.

[23] Stepanyan, V., Kannan, K., Kawamura, E., Lombaerts, T., and Ippolito, C., "Target Tracking with Distributed Sensing and Optimal Data Migration," *AIAA SciTech 2023 Forum*, AIAA-2023-2194, 2023. https://doi.org/10.2514/6.2023-2194.

[24] Dolph, C. V., Minwalla, C., Glaab, L. J., Logan, M. J., Danette Allen, B., and Iftekharuddin, K. M., "Detection and Tracking of Aircraft from Small Unmanned Aerial Systems," *Journal of Aerospace Information Systems*, Vol. 18, No. 11, 2021, pp. 838–851. https://doi.org/10.2514/1.I010911.