An Object-Track-Driven Distributed System for Real-Time Airspace Monitoring

Oluwafemi Ajeigbe[1], Chenyan Zhu[1], and Sandip Roy[1]

***Abstract*— A distributed system is developed for real-time monitoring of multiple small moving objects in airspace volumes, which we term the TRack-based Airspace Monitoring Internet-of-Things System (TRAMIS). TRAMIS comprises: 1) distributed pods with video sensors and edge devices that transmit video data through a wireless mesh network; and 2) a central control node that extracts multiple object tracks and then implements advanced track-based monitoring functions for object initial detection, classification, and 3D-geolocation. The technical innovation in TRAMIS lies in the customization of the computing and networking solution to exploit the dynamic characteristics of inertial airspace objects, so as to achieve a real-time implementation. TRAMIS has been field tested during multi-UAS flight campaigns in Alaska, and a preliminary performance evaluation has been undertaken.**

I. Introduction

Real-time surveillance of low-altitude airspace with heterogeneous small mobile objects, including uncrewed aircraft systems (UASs), birds, and light crewed aircraft, is increasingly needed for civilian and defense applications [5], [8], [17]. Although some airspace surveillance technologies are on the market, they do not enable comprehensive real-time monitoring of *multiple* objects in close proximity. They are also costly and difficult to deploy at scale, precluding wide adoption. Among various types of surveillance technologies, visual and infrared video camera-based solutions have shown promise for multi-object monitoring, but gaps remain.

Real-time camera-based monitoring of multiple small, variable-speed aerial objects outdoors involves several computing and networking challenges: (i) streaming high-volume video data, (ii) maintaining spatiotemporal synchronization across distributed camera nodes, (iii) processing multiple concurrent video streams in real time, (iv) executing detection, geolocation, classification, and other algorithmic pipelines for substantial numbers of objects (10-50+) in the presence of environment/terrain disruptions. These constraints require new approaches for video processing that can extract essential information efficiently in real time.

Distributed video surveillance has been extensively researched (e.g. [4], [6], [10], [12]–[14], [16], [19]–[21]). These studies bring to bear a range of video-analysis techniques, and also use advanced computing/networking approaches (e.g., adaptive workload orchestration and network, cloud-edge workload distribution, computation-offloading) to reduce latency and improve functionality/robustness. However, many of these methods are not well-suited for surveilling multiple small mobile objects in the far field, e.g., due to high object resolution requirements, the need for excessive training data, or high computational burden.

These gaps have motivated an alternate *track-based* approach for airspace surveillance [3], [11], [15], [18], [23], wherein object tracks are extracted from video data, and then used for further monitoring aims. Studies in this space have demonstrated object-track extraction from video sequences, and the ability to distinguish object types from trajectory features [7], [9], [11]. Recently, we have also developed and demonstrated algorithms for multi-object tracking, improved classification, and 3D geolocation [1], [23]. To date, however, real-time ad distributed/multi-camera implementations have not been achieved for the track-based approach.

This article introduces *TRAMIS* (TRack-based Airspace Monitoring Internet-of-Things (IoT) System), a distributed system for real-time multi-object monitoring in dense airspace. The system employs multiple non-collocated sensing pods streaming video over a local wireless network to a central control node, where efficient multi-stream processing extracts object tracks (trajectories) that are the basis for all subsequent monitoring functions. *TRAMIS* converts multiple video streams into lightweight track representations in a way that is customized for multiple small objects with inertial dynamics, enabling real-time detection, classification, and 3D geolocation on commodity hardware. The development of *TRAMIS* has been a multifaceted effort with algorithmic, software, and networking/system integration aspects (see also [1], [15], [23]). Our focus here is to report on the distributed IoT architecture of *TRAMIS* (Section 2), and to summarize customizations and designs of computing/networking functions which support a real-time solution (Section 3). Performance evaluation of TRAMIS during several multi-UAS flight campaigns is also briefly discussed (Section 4).

II. System Architecture

The system architecture for *TRAMIS* is shown in Figure 1. *TRAMIS* comprises distributed sensing pods which surveil an airspace volume, each with multiple video sensors connected to edge devices. Streaming data is transmitted via a local wireless mesh network to a central control node, which processes the video streams (encompassing track extraction, detection, 3D geolocation, and object classification), and displays or pipes information to users. Details are given next.

[1]The authors are with the Department of Electrical & Computer Engineering at Texas A&M University, College Station, TX, 77843. The first two authors contributed equally to this work. Correspondence can be sent to oluwafemi.ajeigbe@tamu.edu

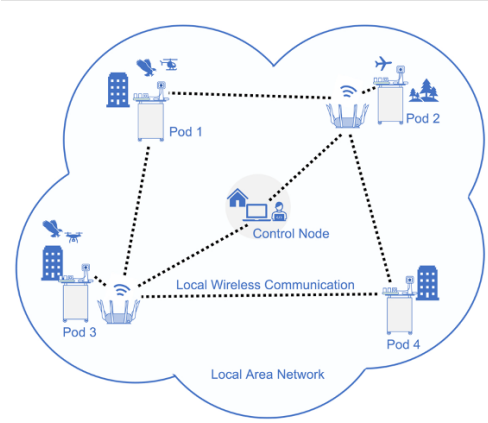

Fig. 1: The *TRAMIS* architecture.

A. Pod-Level Sensing, Computing, and Communications

Each sensing pod (Figure 2) contains sensors, edge computing devices, and supporting components.

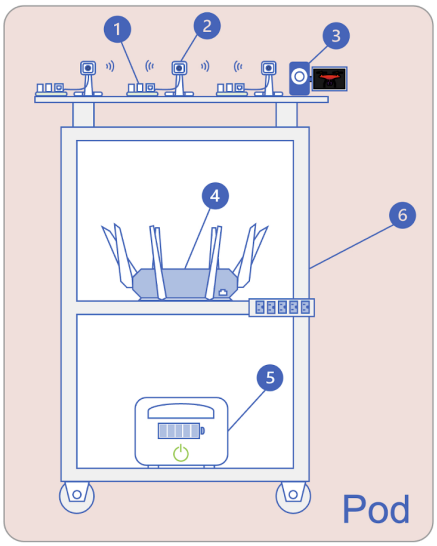

Fig. 2: Architecture of a Sensing Pod.

1) Sensors: Each pod has multiple video cameras. We have commonly used the Raspberry Pi Camera Module 3, which uses a Sony IMX708 sensor (11.9 megapixels, 66° horizontal field of view, f/1.8 aperture), in our deployment. Other cameras, including Camera Module 3 - Wide NoIR, Raspberry Pi Camera Module 2, and Arducam IMX179 USB Camera, have also been integrated. Initial tests have also been conducted using depth cameras such as OAK-D Long Range, and infrared cameras such as TOPDON TC002C Thermal Camera. Sensors capture video at a resolution of up to 1920×1080 pixels at 25 frames per second. All integrated sensors stream data continuously to their connected edge devices. While both visible-spectrum and infrared cameras have been integrated and tested, the results reported in this paper focus on visible-spectrum sensing.

2) Edge Computing: Each sensor is connected to a microcontroller, typically selected as a Raspberry Pi 4 Model B (8GB RAM). The edge device runs two software functions: `rpicam-vid` receives video frames from the connected camera, streaming them to standard output. A Python dispatcher script receives this video stream via a pipe and implements dual-path distribution: frames are simultaneously written to local storage using `ffmpeg` (creating hourly-segmented video files) and transmitted to the control node over TCP (port 8060). The dispatcher maintains a TCP server that accepts connections from the control node and forwards video data in 8192-byte chunks. This architecture allows both live monitoring and post-processing from recorded data.

3) Additional Pod Components: Pod devices can be powered by either a portable power station or a wall outlet. For larger pods, a modular three-tier housing is used; smaller pods are designed as a single tier with all devices.

B. Wireless Mesh Network

The pods and control node communicate via a wireless mesh network. Consumer-grade ASUS routers form the mesh, with routers deployed on a subset of pods. Edge devices connect to the mesh using their built-in WiFi antennas. The control node also connects to the mesh, to receive video streams. A Real-Time Clock (RTC) module is connected to each Raspberry Pi via GPIO pins, for time synchronization across cameras/pods.

C. Control Node

The control node aggregates incoming sensor data from the pods, executes track-based algorithms, and either displays or pipes information about airspace objects to users.

1) Hardware: Either a multi-core CPU-only processor or a processor with CPUs + GPUs can be used for the control node; customized algorithms have been developed for both cases. In our instantiation, several CPU-only laptops (a MacBook Pro (M1) with 10 CPU, and two Dell Inspiron models) have been used. An NVIDIA Jetson Orin Nano device with GPU processing (6-core ARM CPU, 1024 CUDA cores, 8GB RAM) has also been used.

2) Algorithms and Software: The control node uses a multi-threaded architecture to receive and ingest concurrent TCP video streams from multiple pods, with one concurrent thread per sensor video stream. OpenCV libraries are used to perform basic image processing operations. Specifically, a standard Gaussian-mixture-based background subtraction algorithm (MOG2) is used together with morphological filtering to detect and shape foreground motion, with parameters chosen specially for airspace-object tracking. Customized multi-track extraction algorithms are then applied to the output of the background subtraction to form and manage multiple object tracks, allowing capture and distinction of tracks for multiple objects within and across camera feeds in real time. Track filtering is applied to exclude terrain- and environment-related disturbances. Then, advanced track-based algorithms are developed and used for initial object detection, object classification (bird vs UAS vs crewed aircraft), and 3D geolocation via multi-view triangulation. A visualization interface

displays tracked objects, classifications, and aviation-relevant alerts in real-time. This data can also alternatively be piped to remote stakeholders (Figure 3). Control-node algorithms are mostly instantiated in Python, with C++ used to accelerate key routines (nearest-neighbor distance calculations, track-to-detection association, track history maintenance, and periodic track pruning). Multiple CPU cores are also used to speed up distance calculations. For the case where GPU computing is available, specialized instantiations of the standard OpenCV algorithms and the track-extraction algorithms are used (e.g., CUDA-accelerated MOG2 background subtraction).

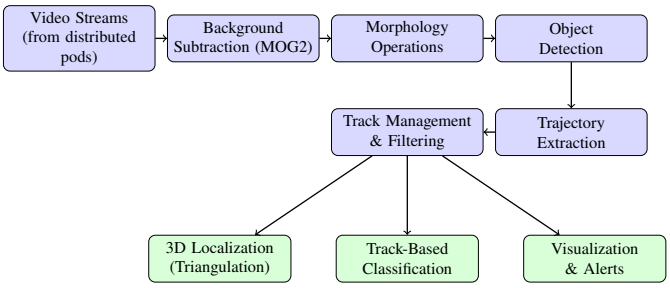

Fig. 3: *TRAMIS* processing pipeline.

III. Customizations for *TRAMIS* Implementation

TRAMIS is customized for real-time surveillance of multiple small, fast-moving aerial objects in airspace volumes. These customizations span algorithmic, computing, and hardware/networking dimensions. Customizations exploit the physics-governed (inertial) motion characteristics of tracked objects, and the specific goals of airspace monitoring, to reduce computation and communication burden while improving monitoring performance. Here, we describe several customizations used in *TRAMIS*. We also summarize some more run-of-the-mill design choices (e.g., edge vs centralized track computation, camera-interface selection) that impact *TRAMIS* performance. In this work, we focus specifically on describing the implementation of these customizations or choices, and how they improve computation/networking. The reader is referred to earlier and concurrent studies [3], [15], [23] for algorithm details and aviation performance benefits of the solution.

A. Customizations in Track Extraction and Management

TRAMIS uses standard video processing algorithms for foreground motion detection and object shaping, but the following multi-track extraction algorithms are customized. Specifically, aerial objects' tracks are governed by inertial physics, which impose temporal/spatial continuity and smoothness constraints. In addition, airspace monitoring requires simultaneous tracking of many small-sized, variable-speed objects. These properties are exploited to reduce computational complexity in multi-track extraction and management. In particular, track forming/extraction requires matching foreground-object motion detections in each frame with existing tracks' states. Rather than performing exhaustive frame-by-frame matching across all detected objects, we use distance- and direction-based association. New detections within a spatial and temporal threshold (selected as 10-20 pixels for standard camera resolutions, and a 3s time window, for typical UAS-traffic management settings) of existing track endpoints (center of mass of the object) are assigned to those tracks. In cases where multiple existing tracks are close by, the directionality of object motion as specified by a cone around the track direction is also used. Further customizations based on object size, estimated distance/speed, and other parameters are also possible. A statistical analysis based on an inertial model can be developed to optimally define the search area, and adaptive search is also possible; details are omitted.

Track state management operates on a per-camera basis, maintaining concurrent trajectories for each video stream. Tracks persist through brief detection gaps, enabling recovery from temporary occlusions or detection failures. The system returns the recent track history (nominally the last 50-100 points per track) for visualization and downstream analytics. Cross-camera track correlation is performed separately using temporal synchronization from RTC timestamps together with heading-closeness measures, enabling multi-view triangulation and track stitching without requiring continuous coordination across cameras during the extraction phase. This approach reduces computation and eliminates the need for bi-directional communications.

This physics-informed approach, which constrains the search space based on expected inertial motion rather than appearance features, enables efficient multi-object tracking with reduced computational complexity compared to exhaustive matching methods.

B. Customized Track Quality Filtering

Raw track extraction produces tracks not only for airspace objects of interest but other moving features in the outdoor environment (vegetation, cloud features, shadows). Spurious tracks due to camera jitter may also be produced. To focus computational resources on object tracks of interest, we apply a per-track quality gate that retains only plausible airspace objects. Specifically, at each time step, *TRAMIS* maintains a set of current tracks, each of which is specified by a sequence of time-stamped object-centroid locations in an image frame. The quality gate computes five metrics for each track: 1) the current number of samples in the track; 2) the current duration of the track (the difference between the current time and the initial detection time of the track); 3) the completeness of the track (the ratio of the number of samples in the track to the number of samples possible over the track duration), 4) the density of the track (average number of track samples per second), and 5) a linearity score computed as a short-duration correlation coefficient averaged over the current track. The quality gate uses thresholds on these metrics to distinguish tracks of interest from others, with thresholds being optimized from labelled field-test data. Specifically, airspace objects of interest are found to have: 1) sufficient number of samples and durations, 2) high completeness and density, and 3) high linearity compared to other tracks.

Tracks that meet the quality threshold are considered for downstream processing, while other tracks are suppressed. This makes downstream algorithms more computationally efficient and higher confidence. As an illustration of the scale of advantage, over three days of field testing (see Section IV below), 8 million object motion detections were reduced to approximately 1 million raw tracks, and only approximately 8000 tracks of interest.

C. Customized Advanced Algorithms: Brief Summary

Track-based algorithms have been developed for a number of advanced monitoring functions, including initial object detection, classification of object types (bird vs. fixed-wing UAS vs. rotor-based UAS vs. crewed aircraft), 3D geolocation, track smoothing, and forecasting.

The development of these algorithms is the focus of other work [1], [23],but we give brief summaries of selected algorithms to give an idea of the computational/networking benefit. The classification algorithm differentiates aerial object types (birds vs. fixed-wing UAS vs. rotorcraft vs. crewed aircraft) based on trajectory features that reflect differences in inertial dynamics and reference-signal structures. The primary metric uses correlation coefficients computed over short track intervals (2-8 seconds) to capture closeness to linearity. Aircraft (both crewed and UAS) exhibit persistently high correlation coefficients (> 0.95) due to their inertial nature and predominantly straight-line flight, while bird tracks show lower correlations (typically 0.6-0.85) due to more erratic motion patterns. A secondary metric based on trajectory deviation amplitudes and temporal autocorrelation over very short intervals (0.2-1 second) further distinguishes cloud tracks and large birds from UAS. Classification confidence increases with observation duration, with reliable discrimination typically achieved within 7 seconds of track initiation

The geolocation algorithm performs multi-view triangulation to estimate object positions in geographical coordinates (latitude, longitude, altitude). Given paired tracks from two cameras, the algorithm computes rays from each camera to the object and determines the closest point of approach between the rays. For multiple objects in the field of view, automatic track pairing is achieved by minimizing the mismatch distance between candidate ray intersections. The system uses one-shot calibration based on a known reference point (such as a surveyed ground location or UAS GPS log data). This approach enables real-time localization of far-field objects without requiring dense 3D scene reconstruction or feature-based object identification across images.

We stress that these algorithms are customized to exploit the dynamic characteristics of airspace objects, including inertial behaviors and also reference-signal structures/modes. Since these algorithms are entirely track-based rather than image-based, they require only limited computation when applied to the gated tracks, and no additional communication. As an example, the classification algorithms require only simple autocorrelation and linear-regression computations for short track segments, which can be done with very limited computational effort even for 10's or 100's of tracks. Similarly, the geolocation algorithm exploits the airspace geometry to automatically pair tracks simultaneously with geolocation.

D. Customization for GPU-based Computation

The full control-node functionality (background subtraction-based motion detection, track extraction, track quality gating, advanced track analyses, display, and piping) is also being implemented using GPU-CPU heterogeneous computing. Overall, the system selectively applies GPU acceleration and CPU-based parallel processing to improve real-time performance. The GPU-based solution is being developed for and tested on a microcontroller — specifically, a Jetson Nano Orin — thus potentially allowing for its flexible use as a control node or directly as an edge device on the pods.

Several functions have been implemented using GPUs. First, a hardware-based video decoder (e.g., nvv4l2decoder) is used to replace a CPU-based H.264 encoding, accompanied by an upgrade to H.265, in order to achieve higher video resolution and frame rates while lowering latency and reducing per-frame processing time. Secondly, for the video processing module, GPU acceleration via CUDA-OpenCV is used, focusing on the computationally intensive components such as MOG and morphological operations. Optimizations, including kernel design and downsampling, have been undertaken, providing faster per-frame processing speed. These GPU-based functions are merged with a multi-thread architecture, allowing multiple frames to be processed in parallel synchronously. Initial tests show that the Jetson Nano achieves accuracy comparable to a CPU-based control center deployed on an Apple M1 MacBook Pro using multi-threaded CPU processing, given proper kernel design and sampling strategy. Meanwhile, the per-frame processing time robustly remains under 0.04 seconds, ensuring stable target point input for track construction. We are working on integrating track extraction, quality filtering, and advanced algorithms for the CPU-GPU architecture.

E. Additional Design Choices

At the system level, one key design choice was to use edge devices only to store and relay video data, while doing all track-based computations (extraction, filtering, advanced monitoring) at the central control node. This architectural choice reduces computational burden at the edge; however, it does so at the cost of much higher-bandwidth computation. We chose this architecture for two reasons:

1) The need for cost-efficient deployment and operation in outdoor environments. This need motivated the use of low-cost, low-power edge devices (e.g., Raspberry Pis), which lack sufficient computational capability to support the full processing pipeline, particularly video processing algorithms such as MOG and morphological operations. Although GPU–CPU-based edge devices can afford the full trajectory extraction pipeline, reducing the cost and consumption of such devices is difficult. For instance, we found that an NVIDIA Jetson Nano was capable of processing the entire pipeline and transmitting trajectory data only from the edge. However, GPU and CPU utilization generally exceeded 90%, leading to significant

thermal buildup. Additionally, the higher cost and power budget for these devices complicate adoption and operation.

2) The need to coordinate tracks across multiple camera views and pods, to support track stitching, classification, and 3D-geolocation. Such coordination is simplified by centralizing track-related computations. In our solution, the control node maintains a unified track database that coordinates information from all camera feeds, enabling multi-view analytics and hence obviating further edge-to-edge communications.

Beyond the architecture selection, several specific design choices impacted performance. First, we found that the choice of camera interface has a significant impact on edge-device real-time performance and data-stream robustness. We evaluated USB and Camera Serial Interface (CSI) options on the Raspberry Pi platform under identical conditions (30 minutes continuous 1080p@25fps capture), see. Table I.

TABLE I: *Performance Comparison: CSI vs USB Camera Interfaces*

Metric	CSI Camera	USB Camera
CPU Usage	10.5%	78.1%
Memory Usage	1353 MB	2020 MB
Temperature	52.1°C	80.1°C

The CSI interface substantially outperforms the USB interface across CPU usage, memory usage, and thermal metrics. The low CPU utilization is supportive of long-time battery-based operation in the outdoor environment, and provides headroom for future edge processing enhancements if needed in future deployments. On the other hand, the USB interface's operation approaches thermal throttling thresholds, risking hardware degradation during extended outdoor deployments. The CSI interface was also observed to be more robust compared to the USB, in the sense of having many fewer (almost nil) broken-pipe errors and color space conversion failures. One cause of the performance difference is that, in a wireless video transmission deployment, USB cameras require additional software layers to emulate a network (web) camera, which introduces additional computational overhead.

Another design choice was to use hybrid software mixing Python and C++ (see Section II). On one hand, Python provides a rich ecosystem of libraries for data analysis and a concise syntax, which makes the development of modular video and track analysis functions convenient. It also offers a flexible foundation for building track databases and new data-informed track-analysis modules into the system. For example, based on Pandas and SQLite, we have developed a dataset with over 10,000 tracks of fixed-wing and rotor-based UASs, birds, and crewed light aircraft, see [22]. On the other hand, C++ is well-suited for efficient implementation of compute-intensive algorithmic components. For example, after migrating the real-time track update module from Python to C++, the per-frame excess processing time was reduced from over 0.01 seconds to below 0.003 seconds, ensuring stable expected frame rates during video stream transmission. The hybrid strategy, therefore, compensates for Python's limitations in execution speed and parallel computing, especially under the constraints of the Global Interpreter Lock (GIL), while preserving Python's rich ecosystem and strong extensibility.

IV. Field Testing and Preliminary Evaluation

TRAMIS has been field tested in multiple settings. These include: two three-day field tests during flight campaigns in Alaska, where multiple UASs in close proximity (as well as birds and crewed aircraft) were monitored for UAS traffic control/management applications; a full-day field test with multi-UAS beyond-visual-line-of-sight flights at NASA's Langley Research Center, also involving many large birds; and multiple shorter field tests in College Station, Texas. The Alaska flight campaigns were focused on three different structured crossing-traffic patterns, with 3-5 variations of each pattern, and 12-50 repetitions for each variation. The patterns included intentional near-conflicts (25-250 feet separation) to stress-test monitoring capabilities.

TABLE II: *TRAMIS System Configuration and Deployment*

Component	Parameter	Specification
Sensing Pods	Camera Resolution	1920 × 1080 @ 25 fps
	Hardware	Raspberry Pi 4B, 8GB RAM
	Network Interface	Wi-Fi (ASUS quad-band gaming router)
Control Node	Processing Hardware	MacBook Pro (M1) 10 CPU Core
	Network	Local wireless mesh
	Software	Python + C++ (track extraction)
Deployment	Test Locations	Alaska, Texas, Virginia
	Network Range	Up to 300m between pods
	Concurrent Streams	4

Table II summarizes the system configuration used for the field tests. For the test, 2-5 pods containing 1-4 cameras each, depending on coverage requirements, were used. The control node was used to process four concurrent video streams simultaneously, with stream selection based on operational priorities. Concurrent streams could originate from different pods or from multiple cameras on the same pod, and were modified based on coverage needs and bandwidth constraints.

A few highlights of the field tests are shown in Figures (4, 5, 6). Figure 4 is a photograph of one pod during a summer field test at Nenana Airport, Alaska. Five pods with 11 cameras were deployed during the field test, with separations ranging from 20-90m. Figures 5 and 6 show sample outputs of the *TRAMIS's* processing pipeline. Specifically, figure 5 shows 3D geolocation of three UASs during a crossing-streams flight pattern. Figure 6 shows real-time classification of three UASs (green) and two birds (purple), as displayed at the control node.

Initial evaluation of TRAMIS has been undertaken in several ways. First, toward understanding operational robustness, the system has been deployed and tested under widely varied temperatures ($0^o - 95^o$ Farenheit, weather conditions (clear-mostly cloudy+light precipitation, varying winds), varied lighting conditions, and time frames (15min-10 hours), varied terrains (e.g., mountainous terrain vs semi-urban). TRAMIS operated correctly in these varied environments, indicating robust operation.

Performance evaluation of TRAMIS functions was undertaken through post-processing of recorded video data

Fig. 4: Sensing pod with passive sensors, edge devices, router, and battery bank.

and comparison with "truth" data from GPS flight logs and/or manual inspection. Meanwhile, network and processing performance were measured directly during live operations and/or laboratory experiments. Holistically, the performance evaluation was focused on three questions: (1) Can the distributed architecture reliably stream and process multiple concurrent video feeds? (2) Does the track-based approach achieve real-time performance on commodity hardware? (3) How accurate are the advanced monitoring functions (e.g., initial detection, classification, 3D geolocation) provided by *TRAMIS*? In the interest of space, the methodology for evaluation is not presented in detail: we refer the reader to ([1], [23]) for further details. In brief, statistical analyses based on repeated trials were used for evaluation. The algorithmic results reported here are based on analysis of one hour of field data during the second flight campaign in Alaska, which contained approximately 220 object tracks of interest; the networking and computing analyses were based on repeated performance readings during the field tests and/or laboratory tests.

TABLE III: *TRAMIS performance metrics during field tests.*

Metric	Measured Value
Network & Streaming	
Per-Stream Video Transmission Rate	2.55 Mbps
Wireless Range Tested	300m
Communication Delay	$< 0.5s$
Processing Performance	
Frame Processing Time	0.08 sec/frame
Control Node CPU Usage	80% (avg)
Concurrent Streams	4
Airspace Monitoring Performance	
Classification Accuracy (Drone vs Birds/Other)	95% - 98%
Object First-Detection Time	< 0.25s for 100% of UASs/birds
Track Fidelity (in frame)	Tracks maintained for 94% of time points
3D Localization Error	8.1m - 39m @ distance between 300m - 800m

Table III presents mean values of several performance metrics, based on the preliminary performance evaluation undertaken. Briefly, the table indicates that networking/computational and functional performance are adequate for most airspace operations, where high fidelity tracking/detection, classification, and geolocation are needed in time frames of a few seconds, for the purpose of conflict avoidance and other operational goals. We point out, particularly, that the fidelity of tracks for small-sized objects (1-10ft in diameter) is quite high compared to existing, radar- or light-spectrum camera systems. We also stress that high-fidelity automated classification is achieved within a few seconds; to our knowledge, automated classification has not been achieved by any other system. As a baseline comparison, standard deep-learning detection methods (e.g., YOLO) were also attempted for classification. Our preliminary testing with YOLOv11 showed minimal detection capability for objects smaller than 20 pixels at distances beyond 200m, while the objects that we were tracking had diameters of 1-4 pixels. This speaks to the value of the track-based approach. It is important to caution that the evaluation of *TRAMIS* is preliminary, as it is based on relatively limited data (observations of network/computing over periods of minutes to an hour, analysis of monitoring performance based on about 200 tracks). More extensive statistical analysis from ongoing longer-duration field campaigns will be reported in future work [2].

An initial qualitative error analysis indicates that a few confounding factors can modulate computational and functional performance: (1) vegetation movement in wind creating spurious tracks near the airspace boundary; (2) cloud edge movements creating linear tracks that can be confused with distant aircraft; and (3) brief track fragmentations during rapid maneuvers or temporary occlusions. The track quality filtering described in Section III-B successfully removes approximately 99% of spurious tracks (reducing ≈1 million raw tracks to ≈8,000 quality tracks during a 3-day campaign) in a computationally friendly way, though refinements to distinguish cloud movements and boundary vegetation are ongoing.

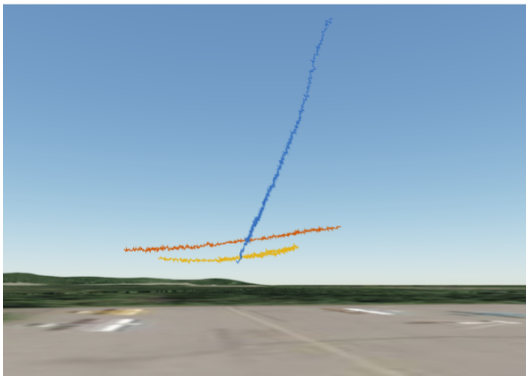

Fig. 5: Sample output of the *TRAMIS* track-based computation pipeline: 3D-Geolocation of multiple objects (UASs) in the airspace volume.

Fig. 6: Sample output of the *TRAMIS* track-based computation pipeline: object classifications. Here, the classification of multiple UAS and bird tracks within an airspace volume is shown.

References

[1] Distributed multi-hazard monitoring system for high-density BVLOS operation. Final report for a federal aviation administration UAS Integration Office project, Texas A&M Engineering Experiment Station, September 2025. Developed by S. Roy with assistance from C. Zhu and O. Ajeigbe. in preparation.

[2] Oluwafemi Ajeigbe, Chenyan Zhu, and Sandip Roy. Field evaluation of multi-object tracking and classification algorithms for high-density BVLOS UAS operations. Submitted to AIAA AVIATION Forum and Exposition, 2026.

[3] Logan Dihel, Chester Dolph, Henry T Holbrook, and Sandip Roy. Classifying aircraft using velocity data with support vector machines and likelihood ratio tests. In *AIAA SCITECH 2023 Forum*, page 0898, 2023.

[4] Ali O Ercan, Abbas El Gamal, and Leonidas J Guibas. Object tracking in the presence of occlusions using multiple cameras: A sensor network approach. *ACM Transactions on Sensor Networks (TOSN)*, 9(2):1–36, 2013.

[5] Federal Aviation Administration. Unmanned aircraft system (UAS) traffic management (UTM) concept of operations v2.0. FAA NextGen Office, Aug 2020. Accessed: Jul. 15, 2025. [Online]. Available: `https://www.faa.gov/sites/faa.gov/files/2022-08/UTM_ConOps_v2.pdf`.

[6] Xiantao Jiang, F Richard Yu, Tian Song, and Victor CM Leung. A survey on multi-access edge computing applied to video streaming: Some research issues and challenges. *IEEE Communications Surveys & Tutorials*, 23(2):871–903, 2021.

[7] Pu Jin, Lichao Mou, Yuansheng Hua, Gui-Song Xia, and Xiao Xiang Zhu. Futh-net: Fusing temporal relations and holistic features for aerial video classification. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13, 2022.

[8] Parimal Kopardekar, Joseph Rios, Thomas Prevot, Marcus Johnson, Jaewoo Jung, and John E Robinson. Unmanned aircraft system traffic management (utm) concept of operations. In *AIAA Aviation Forum and Exposition*, 2016.

[9] Jia Liu, Qun Yu Xu, and Wei Shi Chen. Classification of bird and drone targets based on motion characteristics and random forest model using surveillance radar data. *IEEE Access*, 9:160135–160144, 2021.

[10] Liang Liu, Xi Zhang, and Huadong Ma. Optimal node selection for target localization in wireless camera sensor networks. *IEEE Transactions on vehicular technology*, 59(7):3562–3576, 2009.

[11] Murari Mandal, Lav Kush Kumar, and Santosh Kumar Vipparthi. Mor-uav: A benchmark dataset and baselines for moving object recognition in uav videos. In *Proceedings of the 28th ACM international conference on multimedia*, pages 2626–2635, 2020.

[12] Ya Nan, Shiqi Jiang, and Mo Li. Large-scale video analytics with cloud–edge collaborative continuous learning. *ACM Transactions on Sensor Networks*, 20(1):1–23, 2023.

[13] Charikleia Papatsimpa and Jean-Paul Linnartz. Distributed fusion of sensor data in a constrained wireless network. *Sensors*, 19(5):1006, 2019.

[14] Arun A Ravindran. Internet-of-things edge computing systems for streaming video analytics: Trails behind and the paths ahead. *IoT*, 4(4):486–513, 2023.

[15] Sandip Roy, David N Petrizze, Mengran Xue, Chester Dolph, and Henry T Holbrook. Using trajectory smoothness metrics to identify drones in radar track data. In *AIAA AVIATION 2022 Forum*, page 4004, 2022.

[16] Aswin C Sankaranarayanan, Ashok Veeraraghavan, and Rama Chellappa. Object detection, tracking and recognition for multiple smart cameras. *Proceedings of the IEEE*, 96(10):1606–1624, 2008.

[17] Benjamin Scott. Army Counter-UAS 2021–2028. *Military Review*, pages 65–80, March-April 2021. Accessed: Jul. 15, 2025. [Online]. Available: `https://www.armyupress.army.mil/Portals/7/military-review/Archives/English/MA-21/Scott-Counter-UAS-1.pdf`.

[18] Sutthiphong Srigrarom, Kim Hoe Chew, Denzel Meng Da Lee, and Photchara Ratsamee. Drone versus bird flights: Classification by trajectories characterization. In *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 343–348. IEEE, 2020.

[19] B Suganya, R Gopi, A Ranjith Kumar, and Gavendra Singh. Dynamic task offloading edge-aware optimization framework for enhanced uav operations on edge computing platform. *Scientific Reports*, 14(1):16383, 2024.

[20] Yuting Yan, Sheng Zhang, Xiaokun Wang, Ning Chen, Yu Chen, Yu Liang, Mingjun Xiao, and Sanglu Lu. Visflow: Adaptive content-aware video analytics on collaborative cameras. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 2019–2028. IEEE, 2024.

[21] Shuai Zhang, Chong Wang, Shing-Chow Chan, Xiguang Wei, and Check-Hei Ho. New object detection, tracking, and recognition approaches for video surveillance over camera network. *IEEE sensors journal*, 15(5):2679–2691, 2014.

[22] Chenyan Zhu, Oluwafemi Ajeigbe, and Sandip Roy. Field surveillance data set for high-density UAS traffic management concepts of operation. in preparation, 2025.

[23] Chenyan Zhu, Oluwafemi Ajeigbe, and Sandip Roy. Simultaneous track monitoring of multiple mobile airspace objects using non-collocated video sensors. In *AIAA SciTech Forum*. AIAA, 2026. To appear. Session IS-34 Distributed Data Acquisition and Processing for Advanced Air Mobility III, scheduled Jan 16, 2026. Presentation Control ID 4354005.