Design and Implementation of a UAV based System for Context-Aware Aerial Surveillance of Outdoor Critical Infrastructures

1st Bernardo Albano
INESC INOV and INESC-ID
Instituto Superior Técnico, University of Lisbon
Lisbon, Portugal
bernardoaalbano@gmail.com

2nd António Grilo
INESC INOV and INESC-ID
Instituto Superior Técnico, University of Lisbon
Lisbon, Portugal
antonio.grilo@inov.pt

3nd João Calvário
Advisory Services
KPMG Portugal
Lisbon, Portugal
jcalvario@kpmg.com

***Abstract*—This work delivers a hybrid-VTOL UAV surveillance architecture and prototype implementation, which fuses real-time edge computer vision with cooperative geolocation beacons. Lightweight LTE-M/NB-IoT beacons transmit GNSS data via mutual-TLS to a cloud server; these positions are merged with drone pose, gimbal attitude and zoom level to project beacon locations into the video frame. Persons detected by the onboard GPU are labelled "authorized" or "intruder" using a GNSS-modelled distance threshold. The system also includes IP-based remote control, WebSocket gimbal commands and adaptive WebRTC streaming. Ground tests establish the boundaries of accurate classification and low-latency communications, offering a framework for Beyond Visual Line of Sight surveillance.**

***Index Terms*—UAV-based Surveillance, Beyond Visual Line of Sight, Internet of Drones, Internet of Things, Edge Computing, Target Classification.**

I. Introduction

Unmanned Aerial Vehicle (UAV)-based surveillance systems are increasingly employed to monitor critical outdoor infrastructure in remote or restricted-access environments. The flexibility of UAVs to operate in Three-Dimensional (3D) space, combined with advances in propulsion efficiency, hybrid Vertical Take-Off and Landing (VTOL) architectures, and broadband mobile communication technologies, have expanded the feasibility of both commercial and academic UAV applications, particularly in missions that extend beyond traditional radio-frequency Line of Sight (LOS) constraints [1]. Cellular networks, in particular 4G and 5G, have further enabled reliable connectivity for UAVs in diverse contexts, including environmental monitoring, industrial surveillance, and public safety [2]–[4]. These capabilities are strengthened by the integration of Internet Protocol (IP)-based communication layers that support the Internet-of-Drones (IoD) paradigm and are essential for Beyond Visual Line of Sight (BVLOS) operation scenarios.

Within this ecosystem, Internet of Things (IoT)-focused technologies such as LTE for Machines (LTE-M) and Narrowband-IoT (NB-IoT), classified under Machine Type Communications (MTC) use cases, have gained attention for their scalability, energy efficiency, and suitability for infrastructure-free deployments [5], [6]. Their growing maturity highlights the opportunity to employ cellular MTC networks as the basis for distributed sensing architectures that integrate seamlessly with UAV-based systems.

A persistent limitation of visual-cue-only UAV surveillance lies in its inability to achieve reliable individual-level classification. While State-Of-The-Art (SOTA) vision models can detect object classes such as "person" or "vehicle" with high confidence, the ability to classify entities based on operational context (e.g., distinguishing between "authorized personnel" and "intruders") remains constrained. Prior works have explored cooperative tagging through Ultra Wideband (UWB) anchors or Wi-Fi channel-state information (CSI) [7]–[9], however, these approaches depend on short-range static setups and specialized supporting infrastructure, making them undesirable for outdoor UAV missions.

This work addresses these limitations by proposing an edge-enabled, human-supervised, UAV surveillance framework that augments real-time onboard inference with cooperative, portable geolocation beacons carried by authorized personnel on the ground. The UAV performs local AI-based detection and transmits annotated results to the Ground Control Station (GCS), where an operator supervises classification outcomes in real time. This human-in-the-loop design improves operational reliability while preserving low-latency decision support. Processing visual data at the edge also minimizes bandwidth requirements and reduces dependence on cloud post-processing, which is critical for scalable BVLOS operations.

Firstly, the authors would like to thank the other members of the capstone team, namely Tiago Vera-Cruz, Francisco Alves, Sasha Mot, Prof. Fernando Lau, and Prof. Frederico Afonso. The authors would also like to thank Portuguese mobile operators NOS and Vodafone for providing the 4G and 5G SIM cards used in this project. In particular, they would like to thank Ricardo Dinis (NOS), Raul Almeida (Vodafone), Mónica Gomes (NOS), and Luís Santo (NOS) for all the technical help and invaluable insights regarding scenario design. This work was also partially supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project UIDB/50021/2020.

Unlike indoor or infrastructure-bound localization systems, the proposed framework employs cellular MTC connectivity to realize a distributed outdoor sensing architecture, enabling cooperative position tagging without dedicated ranging infrastructure, aligning with commercial integration goals.

The main contributions of this work are the following:

- Design and integration of a UAV payload featuring cellular connectivity and cloud integration capabilities, developed to enable BVLOS perimeter surveillance applications.
- Implementation and validation of a complete surveillance prototype for outdoor critical infrastructures, combining UAV-based visual analytics with NB-IoT/LTE-M-enabled geolocated tags for real-time intruder detection.

The rest of this document is structured as follows. Section II provides an overview on the project in whose context this work was developed. Section III explains the theoretical models used to calculate the coordinates of detected targets. Section IV describes the implementation of the UAV based surveillance system. Section V presents the results of the validation tests. Finally, Section VI concludes the paper.

II. Project Overview

This work is part of a multidisciplinary engineering project at Instituto Superior Técnico (IST), funded by KPMG, and was developed in collaboration with NOS, Vodafone. The project team consisted of four graduate students. The project aimed to integrate cellular communication technologies in a UAV platform designed for aerial surveillance applications, supporting BVLOS operations. Within the scope of the project, a UAV platform was designed and manufactured [10], [11] with the purpose of carrying the payload of the system described in this paper.

A. Quadplane Platform

Among multiple UAV configurations the hybrid quadplane VTOL emerged as the most versatile baseline platform across varied scenarios [11]. Its adaptability, coupled with academic interest in its aerodynamic and structural complexity, justified its selection and manufacture. The quadplane is shown in Figure 1.

Fig. 1. Hybrid VTOL Quadplane developed.

When fully assembled, the quadplane has a wingspan of approximately 2.5 m and a length of 1.3 m from nose to tail. The total mass is around 9.2 kg. Key avionics include the Pixhawk 6X Flight-Controller (FC) and the M9N Global Navigation Satellite System (GNSS) receiver.

B. Payload Selection

The team acquired the following systems:

- **On-board Computer:** Following the conclusions established in [12], which evaluated SOTA challenges similar to our own, the NVIDIA Jetson Orin Nano [13] was selected. Its advantages are rooted in: I) hardware characteristics, notably high AI performance enabled by the integrated GPU; II) software maturity, particularly the *CUDA*, *TensorRT*, and *DeepStream* frameworks, which offer robust support and *Python* integration.
- **4G/5G User Equipment:** Cellular connectivity is established using a combination of a carrier board and a modem. The system consists of the following items:
  - *5G M.2-to-Gigabit Ethernet* carrier board by Waveshare: a robust, fully enclosed platform compatible with M.2 (NGFF) PCIe modems.
  - *SIM8262E-M2* module by SIMCom: a cellular modem compatible with 3G, 4G, and 5G frequency bands used in Europe.
- **Gimbal Camera:** The SIYI ZR10, chosen based on a comparative study conducted in [12]. The SIYI ZR10 is a 3-axis stabilized gimbal camera. It supports video recording at 1K Full HD (1920×1080) resolution and offers an optical zoom range from 1× to 10× [14].

The Jetson interfaces with the modem and avionics (through the FC) via USB. The ZR10 is connected via Ethernet.

C. Operational Concept and Application Scenario

Figure 2 illustrates the operational concept of the proposed UAV-based surveillance architecture developed in this study.

Based on this operational concept, the surveillance system was defined to meet the following functional requirements:

- Implement a classification framework distinguishing **authorized personnel** from **intruders**[1] by fusing computer-vision inference with geolocated beacon data.
- Establish a secure cloud-based communication framework to interconnect the UAV, the GCS, and other IP-enabled components
- Enable real-time streaming of processed video with visual overlays of classification results to the GCS.
- Support remote control of on-board systems through IP-based protocols, forming the basis for BVLOS operation.

The system fuses the location telemetry of authorized personnel within the UAV's visual frame. The classification algorithm combines the UAV's pose (attitude and position) with the broadcasted beacon positions to determine individual identities. The next section presents the theoretical model used to compute these coordinates and to characterize the relevant subsystems.

[1] An intruder is here defined as someone not carrying a beacon who is also not close to authorized personnel.

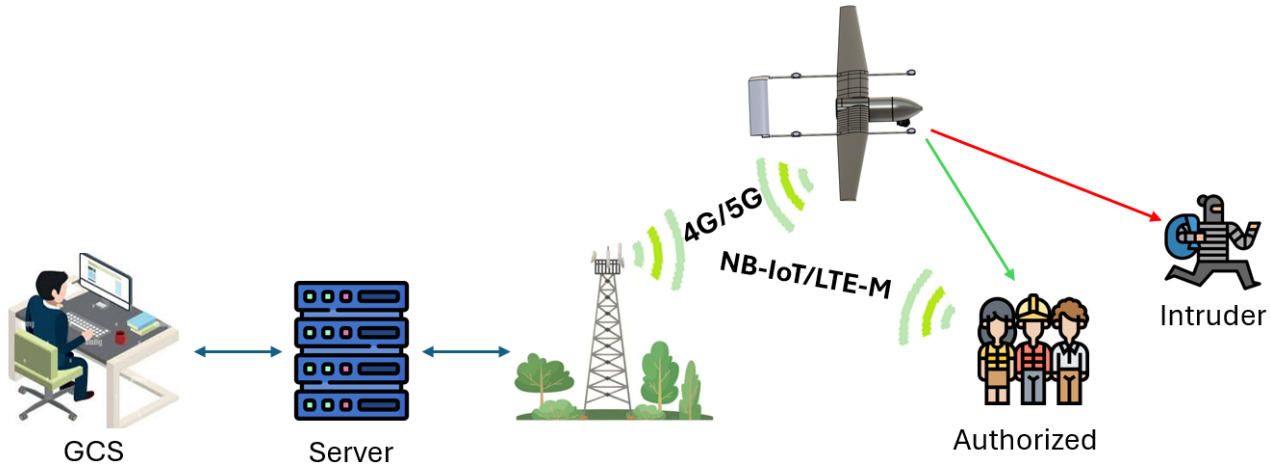

Fig. 2. Aerial surveillance and target classification operational concept.

III. Theoretical Models of Target Coordinates

Both the UAV (denoted in this section by the subscript $_u$) and the authorized location beacon (denoted by $_b$) use the GNSS system to determine their position in the global reference frame. However, the computer vision algorithms used to detect people or other targets operate in the image pixel frame. It is therefore necessary to establish the theoretical framework that enables the projection of personnel coordinates onto the image plane, and conversely, the projection of image coordinates into the global reference frame.

A. Transformation between Bases

The coordinates of a point expressed in Cartesian frame *A* can be converted to the Cartesian frame *B* in two steps: I) *Translate:* subtract the origin of B expressed in A, ${}^{A}_{B}\vec{\mathbf{o}}$, to the point vector ${}^{A}\vec{\mathbf{v}}$; and II) *Rotate:* premultiply by the transformation matrix ${}^{B}_{A}\mathbf{T}$. The conversion is computed as follows [15]:

$$ {}^{B}\vec{\mathbf{v}} = {}^{B}_{A}\mathbf{T} \cdot \left({}^{A}\vec{\mathbf{v}} - {}^{A}_{B}\vec{\mathbf{o}} \right). \tag{1} $$

The orientation of the rotation follows the right-hand convention for positive angles of rotation, θ, α and β, over the respective basis vectors, $\vec{\mathbf{i}}$, $\vec{\mathbf{j}}$ and $\vec{\mathbf{k}}$. The respective transformations matrices are [15]:

$$ {}^{B}_{A}\mathbf{T}_{\mathbf{i}}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) \\ 0 & -sin(\theta) & cos(\theta) \end{bmatrix}; \tag{2} $$

$$ {}^{B}_{A}\mathbf{T}_{\mathbf{j}}(\alpha) = \begin{bmatrix} cos(\alpha) & 0 & -sin(\alpha) \\ 0 & 1 & 0 \\ sin(\alpha) & 0 & cos(\alpha) \end{bmatrix}; \tag{3} $$

$$ {}^{B}_{A}\mathbf{T}_{\mathbf{k}}(\beta) = \begin{bmatrix} cos(\beta) & sin(\beta) & 0 \\ -sin(\beta) & cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4} $$

B. Frames of Reference

1) Geodetic Coordinates: The World Geodetic System of 1984 (WGS 84) is the standard Earth reference model used in GNSSs, including systems such as Global Positioning System (GPS) and Galileo. It models the Earth as an oblate ellipsoid of revolution, which more accurately approximates the Earth's shape than a perfect sphere [16]. Coordinates are reported with latitude ϕ, longitude λ, and height above or below the ellipsoid, h.

2) Earth-Centered Earth-Fixed (ECEF) Frame: A 3D Cartesian coordinate system fixed to the rotating Earth. Its origin is Earth's center of mass, and its axes rotate with the Earth, therefore the ECEF system is not inertial. ECEF coordinates can be calculated from WGS 84 with [17]:

$$ \begin{cases} x = (R_N + h)\, cos(\phi) cos(\lambda); \\ y = (R_N + h)\, cos(\phi) sin(\lambda); \\ z = \left[(1-f)^2 R_N + h \right] sin(\phi); \\ R_N = a / \sqrt{1 - f(2-f) sin^2(\phi)}; \end{cases} \tag{5} $$

where $a = 6378137.0\,m$ and $f = 298.257223563^{-1}$ [18].

3) North-East-Down (NED) Frame: A Cartesian coordinate system whose origin is arbitrary, with $\vec{\mathbf{i}}$ pointing toward true North, $\vec{\mathbf{j}}$ pointing East, and $\vec{\mathbf{k}}$ pointing downward.

By establishing the UAV's NED frame through a sequential set of rotations from the ECEF frame, we obtain:

$$ {}^{N}_{E}\mathbf{T}_{u} = \mathbf{T}_{\mathbf{j}}(-\phi)\ \ \mathbf{T}_{\mathbf{i}}(\lambda)\ \ \mathbf{T}_{\mathbf{j}}(-90°). \tag{6} $$

The coordinates of a beacon in the NED frame, with the origin located at the UAV, are therefore:

$$ {}^{N}\vec{v_b} = {}^{N}_{E}\mathbf{T}_{u} \left(\vec{p_b} - \vec{p_u} \right), \tag{7} $$

where $\vec{p_b}$ and $\vec{p_u}$ are the ECEF coordinates computed using Equation (5).

4) UAV Body Frame: A Cartesian frame B with origin at the UAV's center of mass, where $\vec{\mathbf{i}}$ points toward the nose, $\vec{\mathbf{j}}$ points to the right wing, and $\vec{\mathbf{k}}$ points downward.

The transformation matrix from the NED frame to the Body frame is:

$$ {}^{B}_{N}\mathbf{T} = \mathbf{T}_{\mathbf{i}}(\phi)\,\mathbf{T}_{\mathbf{j}}(\theta)\,\mathbf{T}_{\mathbf{k}}(\psi). \tag{8} $$

Here, ψ, θ, and φ represent the UAV's yaw, pitch, and roll angles, respectively.

From this frame onward, differences in frame origin will be neglected, as the distances between successive origins are negligible compared to the UAV's distance to any point of interest in this application.

5) Gimbal Frame: The Gimbal frame G has its origin at the camera module, with $\vec{\mathbf{i}}$ pointing forward along the optical axis, $\vec{\mathbf{j}}$ pointing to the right (aligned with the horizontal axis of the Field-of-View (FOV)), and $\vec{\mathbf{k}}$ pointing downward (aligned with the vertical axis of the FOV). The transformation matrix from the Body frame to the Gimbal frame is:

$$ {}^{G}_{B}\mathbf{T} = \mathbf{T}_{\mathbf{i}}(\delta)\,\mathbf{T}_{\mathbf{j}}(\beta)\,\mathbf{T}_{\mathbf{k}}(\alpha). \tag{9} $$

Here, α, β, and δ represent the gimbal's yaw (pan), roll, and pitch (tilt), respectively.

6) Camera Frame: The Camera frame C has its origin at the camera's focal point and is defined by a rotation of the Gimbal frame to match the coordinate system used in the pinhole projection model, detailed next.

$$ {}^{C}_{G}\mathbf{T} = \mathbf{T}_{\mathbf{k}}\,(-90°)\mathbf{T}_{\mathbf{j}}(90°). \tag{10} $$

C. Pinhole Model

The pinhole camera model [19] is an idealized geometric representation of how a 3D scene is projected onto a Two-Dimensional (2D) image plane.

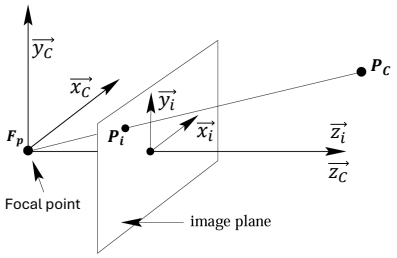

Fig. 3. Pinhole model geometry [19].

On the image plane, the optical sensor is modeled with the same shape and dimensions as the physical camera sensor, a rectangular bounding box. Any point P_i projected inside this box corresponds to a ray of light intersecting the optical sensor.

Assuming zero skew, the transformation from the Camera frame $(x_C,\, y_C,\, z_C)$ to the Pixel frame $(x_p,\, y_p)$ can be written using the camera intrinsic matrix K [19]:

$$ \begin{bmatrix} z_C x_p \\ z_C y_p \\ z_C \end{bmatrix} = K \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix}, \text{ where } \quad K = \begin{vmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{vmatrix}; \tag{11} $$

$$ \alpha_x = \frac{p_x f}{s_w}; \quad \alpha_y = \frac{p_y f}{s_h}; \quad x_0 = \frac{p_x}{2}; \text{ and } \quad y_0 = \frac{p_y}{2}. \tag{12} $$

The focal length f of the camera depends on the zoom level. However, neither the mapping of f across zoom levels nor the sensor width s_w and height s_h were provided by the manufacturer. Only the pixel resolution, p_x and p_y, is known.

Instead of estimating the sensor dimensions and modeling the zoom-to-f mapping directly, the ratios $\frac{f}{s_w}$ and $\frac{f}{s_h}$ were modeled. This is done using the formulation in (11), considering the application to a point with pixel coordinates $(x_p = y_p = 0)$, which corresponds in the camera frame to the point $(x_{\text{edge}},\, y_{\text{edge}},\, z_{\text{edge}})$ as geometrically illustrated in Figure 4.

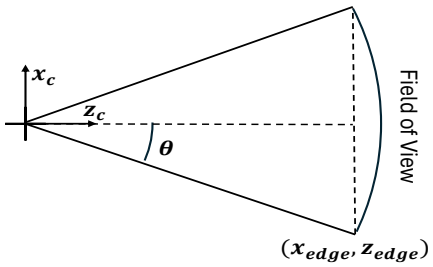

Fig. 4. Camera horizontal FOV.

From this, we obtain:

$$ \begin{cases} -x_{\text{edge}}/z_{\text{edge}} = \tan(\theta) \Longrightarrow f/s_w = (2\tan(\theta))^{-1}; \\ -y_{\text{edge}}/z_{\text{edge}} = \tan(\alpha) \Longrightarrow f/s_h = (2\tan(\alpha))^{-1}. \end{cases} \tag{13} $$

Where θ and α represent, respectively, half the horizontal and vertical FOV.

For our ZR10, experiments were conducted to determine θ and α at different zoom levels, from which the horizontal and vertical ratios were modeled as linear regressions:

$$ \begin{cases} \frac{f}{s_w}(z) = 0.920z - 0.020; \\ \frac{f}{s_h}(z) = 1.55z - 0.041. \end{cases} \tag{14} $$

D. Classification Rule

The classification of targets identified by the computer vision inference pipeline is achieved by projecting the positions of nearby beacons into the UAV's camera pixel frame.

Combining the previously discussed transformations, the projection of a beacon's 3D ECEF position into the camera's 2D pixel frame is expressed as follows:

$$ z_C \begin{bmatrix} x_b & y_b & 1 \end{bmatrix}^T = \mathbf{K}\ {}^{C}_{G}\mathbf{T}\ {}^{G}_{B}\mathbf{T}\ {}^{B}_{N}\mathbf{T}\ {}^{N}_{E}\mathbf{T}\,(\vec{p_b} - \vec{p_u}). \tag{15} $$

In an ideal scenario, the projected pixel coordinates would precisely coincide with the matching person. However, due to sensor measurement errors and sampling time, this alignment is not exact. Therefore, the classification of a detected target as *authorized* is based on the Euclidean distance between the detected person's pixel coordinates $(x_s,\, y_s)$ and the closest projected beacon coordinates $(x_b,\, y_b)$:

$$ \sqrt{(x_s - x_b)^2 + (y_s - y_b)^2} < d_{\text{thre}}. \tag{16} $$

The threshold d_{thre} is selected based on the expected error in the tracking vector $\vec{v_T} = \vec{p_b} - \vec{p_u}$.

E. GNSS Measurement Error Model

The GNSS measurements were modeled as stochastic processes in the ECEF reference frame (since this frame is Earth-fixed), as follows:

$$\vec{p} = \vec{\mu} + \vec{\epsilon}, \tag{17}$$

where:

- $\vec{\mu}$ is the true (but unknown) position in the ECEF frame; and
- $\vec{\epsilon}$ is a measurement error vector that captures all sources of uncertainty.

We assume that the error $\vec{\epsilon}$ is a zero-mean Gaussian random vector, a reasonable approximation under the Central Limit Theorem, given that many independent noise sources are involved [20]:

$$\vec{\epsilon} \sim \mathcal{N}(0, \Sigma), \tag{18}$$

where Σ is the 3×3 covariance matrix summarizing the uncertainties in the x, y, and z directions in the ECEF reference frame.

To characterize the distribution of $\vec{\epsilon}$, we use the Mahalanobis (squared) distance [21]:

$$d_p^2 = \left(\vec{p} - \vec{\mu}\right)^{\mathsf{T}} \Sigma^{-1} \left(\vec{p} - \vec{\mu}\right) = \vec{\epsilon}^{T} \Sigma^{-1} \vec{\epsilon}. \tag{19}$$

Under the assumption of a Gaussian error model, this distance follows a chi-squared distribution with 3 degrees of freedom [22].

The confidence ellipsoid that models the α-quantile of the error distribution can therefore be expressed using the eigenbasis of Σ and the tabulated value $\chi^2_{3,\alpha}$, the α-quantile of the chi-squared distribution with 3 degrees of freedom.

Let $\lambda_1, \lambda_2, \lambda_3$ be the eigenvalues and $\vec{v_1}, \vec{v_2}, \vec{v_3}$ the corresponding orthonormal eigenvectors of Σ. Then, the lengths of the semi-axes of the ellipsoid are [23]:

$$l_{i,\alpha} = \sqrt{\chi^2_{3,\alpha} \cdot \lambda_i}, \qquad i = 1, 2, 3. \tag{20}$$

These semi-axes define the ellipsoid that contains a proportion α of the measurement error $\vec{\epsilon}$.

Assuming the UAV and beacon GNSS measurements, $\vec{p_u} = \vec{\mu_u} + \vec{\epsilon_u}$ and $\vec{p_b} = \vec{\mu_b} + \vec{\epsilon_b}$, are independent, the tracking vector's stochastic model is characterized by:

$$\vec{\mu_T} = \vec{\mu_b} - \vec{\mu_u}; \qquad \Sigma_T = \Sigma_b + \Sigma_u. \tag{21}$$

The confidence ellipsoid of the tracking vector is then used to define the threshold d_{thre}.

IV. Surveillance System Implementation

This section presents the implementation of the functional requirements for the application scenario discussed.

A. Geolocation Beacon

The geolocation beacon prototype is implemented using the LilyGo *T-SIM7080G S3* development board, priced at approximately €28. The board integrates an ESP32-S3 microcontroller, a SIMCom SIM7080G modem, and an AXP2101 Power Management Unit (PMU). This platform was selected primarily for its low cost and integration of key capabilities: the SIM7080G modem provides LTE-M and NB-IoT connectivity, supports secure communication over HTTP-Secure (HTTPS) (Transport Layer Security (TLS) 1.1/1.2), and includes a built-in GNSS receiver.

The assembled beacon weighs 115 g and is powered by a 46 g lithium-ion cell with a capacity of 3250 mAh, providing up to 26 hours of autonomy and requiring approximately 8 hours for a full recharge. A critical hardware constraint is that the SIM7080G modem cannot operate its GNSS and cellular subsystems simultaneously, requiring serialized operation of these modules.

1) Firmware Architecture: The firmware is developed in C/C++ using the Arduino framework. It is structured into two main phases: initialization and the main execution loop.

- **Initialization:** Performed once on startup, this phase:
  - Configures the PMU, including power rail behavior, system thresholds, and battery parameters.
  - Sets network parameters on the modem, including radio technology and TLS configuration (version and cipher suite).
  - Loads x.509 certificates onto the modem to enable mutual-TLS (mTLS) over HTTPS.
- **Execution Loop:** Repeats continuously after initialization, following this sequence:
  - Deregister and power down the cellular subsystem.
  - Power up the GNSS receiver and acquire a valid position. A valid position requires a NMEA fix status and a number of satellite locks $\geq m_{sat}$.
  - Power down the GNSS receiver, re-enable the cellular subsystem, and reattach to the network.
  - Establish an HTTPS session, encode telemetry in JSON format, and transmit via a POST request.

Since GNSS and cellular subsystems must operate sequentially, the sampling frequency is limited by the time required to reacquire satellite lock after each GNSS power cycle.

2) Cloud Integration for Beacon Telemetry: The geolocation IP beacons and the UAV communicate with a Flask-based web application deployed in the central server. This application interfaces with a back-end *PostgreSQL* database, which stores and manages beacon location data. The server exposes two REST endpoints:

- **POST** `/add_location`: Accepts a JSON payload containing the beacon's reported telemetry.
- **GET** `/get_locations`: Returns a JSON-formatted object containing the most recent coordinates of all valid beacons stored in the database.

All traffic is secured using mTLS, which authenticates both the beacons and the UAV. Separate x.509 root certificates are

issued for beacons and UAVs, preventing an attacker with stolen beacon credentials from accessing the `GET` endpoint.

B. GCS Remote Control

The GCS interfaces with external systems via Secure Shell (SSH), using the *OpenSSH* tool. To simplify and streamline these interactions, various shell scripts are used for automating routine tasks:

- **Central Server:** The GCS accesses the server through a standard SSH connection, using either the server's fixed IP address or domain name. From this connection, the operator can remotely launch or shut down critical services, including: I) the Flask-based beacon telemetry server; II) JavaScript services (Gimbal Proxy, WebRTC cloud services); and III) the Coturn TURN server.
- **UAV On-board Computer:** The primary challenge in connecting to the UAV's onboard computer is the lack of a public IP address. The 5G SIM cards used in the UAV provide basic, unlimited data service but operate under a dynamic IP model managed by the Access Point Name (APN) of the 4G/5G operator, which does not support inbound connections.
  To overcome this, the UAV establishes a reverse SSH tunnel to the server upon boot and successful network connection. The GCS then logs into the server via a regular SSH session and uses a second, forwarded SSH connection to tunnel into the UAV. This grants the operator shell access to the onboard system, from which UAV-side services can be managed, including: I) the gimbal control client; and II) the Python-based multi-streaming computer vision pipeline.

C. Gimbal IP Control

The UAV employs the SIYI ZR10 gimbal. Traditional integration methods for this gimbal rely on the FC to issue commands and forward video through a proprietary transceiver. However, to support BVLOS operations and reduce system costs, an IP-based remote control strategy was adopted.

This control framework involves three components:

1) **UAV-side Agent:** A Python script wraps a open-source gimbal driver within a WebSocket (WS) client that connects to the proxy server. When JSON-formatted commands are received from the WS server, the script issues them to the ZR10 via a socket. This client implements: I) pitch and yaw control via angular velocity adjustment; II) focus; III) gimbal mode selection (FPV, Follow, and Lock); IV) zoom in/out functionality; V) gimbal centering. Additionally, this agent runs a local WS server that relays gimbal telemetry (attitude and zoom level) to the classification pipeline.
2) **GCS-side Agent:** Implements a simple Graphical User Interface (GUI) and keyboard event-driven framework, which sends commands when triggered by the operator. Like the UAV-side agent, it connects to the central proxy server to establish the communication channel.
3) **Centralized Proxy Server:** A lightweight Node.js WS proxy running on the central server, which facilitates real-time communication between the operator interface and the UAV-side agent.

The Round-Trip Time (RTT) averaged 147 ms. 95 % of the RTT values fall within the range of 24 to 368 ms. The time to relay commands can be estimated to be half of the RTT.

D. Individual Classification and Video Pipeline

The integration of the media pipeline with the classification-augmentation sensor suite was implemented in a unified Python script that coordinates real-time telemetry acquisition, data fusion, GStreamer pipeline management, WebRTC negotiation and adaptive bitrate control.

Target classification is done by computing the pixel coordinates of all known beacons using Equation (15) and comparing them against the inference metadata. Bounding boxes are colored according to the results defined in Equation (16).

The video pipeline was implemented using GStreamer. This is a modular, event-driven multimedia framework that leverages hardware-accelerated plugins to offload video processing tasks to the GPU.

Although multipeer streaming was not a core requirement of the project, the pipeline was designed with support for simultaneous video streaming to multiple remote clients.

The full GStreamer pipeline, represented in Figure 5, and explored in detail in [15], integrates both inference and streaming in a unified structure, which can be conceptually divided into two major segments: a static segment and a dynamic segment.

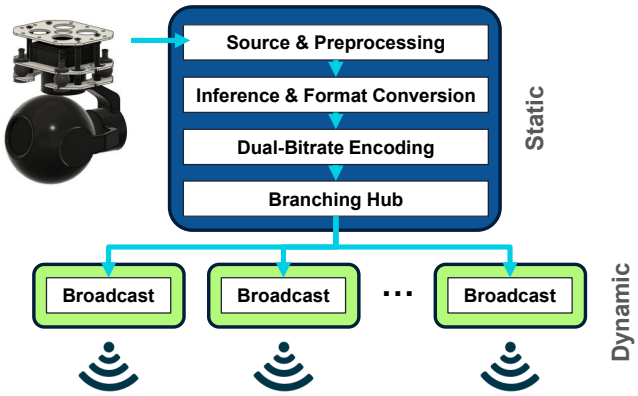

Fig. 5. Representation of the Gstreamer video pipeline.

1) Static Pipeline Segment: This segment remains active from pipeline initialization to shutdown and handles media acquisition, processing, encoding, and distribution. For clarity, it is subdivided into four functional blocks:

a) RTSP Video Source and Preprocessing: Acquires Real-time Transport Protocol (RTP) packets from the Real-Time Streaming Protocol (RTSP) source (ZR10). It then retrieves, parses and decodes the H.264 stream into raw frames.

b) Image Processing, Inference, and Format Conversion: This block performs frame rate control, color conversion, GPU memory allocation, inference execution, and annotation. The used inference model was the `resnet18_trafficcamnet_pruned` [24], which, among other classes, supports the classification of people.

c) Dual-Bitrate Encoding and Stream Splitting: Encodes the processed video into two VP8 streams, one high-quality and one bandwidth-optimized, to support adaptive streaming based on network conditions. While using separate encoders for each client could provide finer bitrate control, each encoder introduces significant CPU utilization on the Jetson platform.

d) Terminal Distribution Hub: This final block acts as the output distribution point for the static pipeline, enabling new publisher branches to be spawned and connected to on demand.

2) Dynamic Pipeline Segment: A new broadcast branch is instantiated dynamically per client. WebRTC was selected for its sub-500 ms latency, built-in NAT traversal via ICE/TURN, and native browser support. Compared to alternatives such as RTSP or SRT, WebRTC better suits interactive, peer-to-peer streaming over mobile networks in BVLOS scenarios. Quality of Service (QoS) feedback is used to dynamically select the encoded stream according to network conditions.

The measured WebRTC mean RTT was ≈90 ms in both the peer-to-peer and TURN-relayed cases, with a 97.5% of the samples under 260 ms. The measurements show that, for our deployment, the additional hop through the TURN server does not impose a measurable latency penalty.

E. WebRTC Cloud Services

To support WebRTC video broadcasting, several cloud services were deployed on the central server to handle signaling, stream delivery, and client interface hosting:

- *TURN Server:* A Coturn-based TURN server relays traffic when direct peer-to-peer connections are not possible due to NAT restrictions.
- *WebSocket Servers:* Node.js was used to host the necessary WS services used to connected the UAV publisher to the ground viewers and redirect the WebRTC signalling messages.
- *HTTP Server:* A lightweight Express.js server serves the `viewer.html` page, which acts as the user interface for video streaming.
- *Viewer Frontend:* The `viewer.html` file includes HTML and JavaScript that: I) establish WS connections with the signaling servers; II) performs SDP/ICE-based WebRTC negotiation; and III) displays the video stream in an HTML `<video>` element.

V. Experimental Results

This section describes the prototype performance and validation tests.

A. Tracking Vector Error Characterization

To characterize the tracking vector error, both the beacon and the Pixhawk GNSS receiver were placed stationary on an open-sky platform, and their telemetry was logged continuously for 24 hours. Using Equation (20) and the recorded samples, the semi-axis lengths of the 95% confidence ellipsoids were obtained for both receivers.

The ellipsoid corresponding to the beacon measured (19.2, 9.2, 5.8) m along its principal axes, substantially larger than that of the Pixhawk receiver, which measured (4.4, 2.1, 1.1) m. These results indicate that the beacon is the primary contributor to the overall tracking vector uncertainty. Due to the hardware constraints described earlier, the beacon transmits a new position approximately every 32 s.

The resulting 95% confidence ellipsoid for the tracking vector, $\vec{v_T}$, exhibits semi-axis lengths of (19.6, 9.4, 6.2) m, consistent with the dominant influence of the beacon's estimation error. Based on this characterization, the Euclidean distance threshold d_{thre} used in the classification rule (Equation (16)) was set to 20 meters.

B. Classification Algorithm Results

The methodology of the test consisted in having the operator to manually control the ZR10 gimbal's attitude and zoom to capture the video feed, while people acting as targets walked along a designated path.

The beacon's tracker is shown with the label `tracker_B`, and detected people are outlined in color-coded boxes: I) Green outlines indicate classification as "authorized" personnel, in close proximity to a beacon; and II) Red outlines indicate a "intruder" located beyond d_{thre}, and thus not associated with any nearby beacon.

Figure 6 shows the beacon tag, the beacon carrier outlined with a green bounding box labeled "authorized", and a second person (without beacon) is outlined in red as an "intruder".

Fig. 6. "Intruder" beyond Beacon Range.

Fig. 7. Reclassification to "Authorized".

As shown in Figure 7, when the person initially far from the beacon approaches the beacon and its carrier, their bounding box changes from red to green, indicating correct reclassification as ”authorized” personnel, which is the expected behavior.

The effect of the beacon's update rate (approximately 32 s) can be clearly seen in Figures 8 and 9.

During the experiment, the beacon carrier walked from left to right across the scene. In Figure 8, the tracker is still projected near the left edge of the image, representing the last broadcasted position. By this point, however, the carrier had already moved about 30 m forward, causing the system to misclassify the beacon carrier as an “intruder.”

Fig. 8. Misclassification due to outdated Beacon Position.

In Figure 9, the updated beacon position is finally received. As a result, the system correctly reclassifies the carrier as ”authorized”, aligning their current position with the tracker.

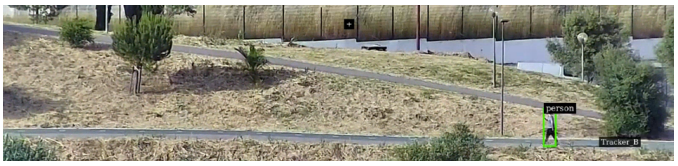

Fig. 9. Correct classification after beacon position update.

These results highlight the limitation in the classification methodology introduced by the constrains in the currently selected beacon hardware. The low position sampling degrades the classification algorithm accuracy for moving targets.

VI. Conclusions

This paper presented the design, implementation, and validation of a cellular-enabled UAV-based surveillance system that augments standard computer vision pipelines through the integration of cooperative geolocation beacons. The proposed approach is particularly suited to outdoor surveillance missions and aligns with BVLOS and IoD paradigms.

By fusing the UAV's pose, gimbal orientation, and zoom level with beacon positions transmitted over LTE-M/NB-IoT, the system extends inference capabilities from class-level to identity-aware classification. Although performance for moving targets was affected by hardware-imposed sampling limitations, the experimental results demonstrate the feasibility of the proposed approach and suggest significant improvements with higher beacon position sampling rates.

A control framework was developed to enable remote access and real-time gimbal operation, while a multi-stream WebRTC pipeline provided low-latency video transmission through standard web interfaces. Ground-based validation confirmed the effectiveness of both the cooperative classification logic and the communication architecture.

Future work will focus on extended in-flight testing, further latency reduction via optimized WebRTC-based control, improved autonomy of remote services, and upgraded beacon hardware.

References

[1] H. Ongkowijoyo et al. Effect of uav flight characteristics to mobile network quality for uav bvlos operations. In *2022 IEEE 4th Eurasia Conference on IOT, Communication and Engineering (ECICE)*, pages 5–8, 2022.

[2] Sarun Duangsuwan and Phoowadon Prapruetdee. Drone-enabled ai edge computing and 5g communication network for real-time coastal litter detection. *Drones*, 8(12), 2024.

[3] Minh Long Hoang. Smart drone surveillance system based on ai and on iot communication in case of intrusion and fire accident. *Drones*, 7(12), 2023.

[4] Thapelo Makhalany Ane, Lusani Mamushiane, Hlabishi Kobo, and Albert Lysko. Towards a 5g equipped video surveillance uav for public safety. In *2024 IST-Africa Conference (IST-Africa)*, pages 1–11, 2024.

[5] Benjamin Finley and Alexandr Vesselkov. Cellular iot traffic characterization and evolution. *CoRR*, abs/1911.02877, 2019.

[6] Jie Ding, Mahyar Nemati, Chathurika Ranaweera, and Jinho Choi. Iot connectivity technologies and applications: A survey. *IEEE Access*, 8:67646–67673, 2020.

[7] F. Liu et al. An uwb/vision fusion scheme for determining pedestrians' indoor location. *Sensors*, 20(4):1139, 2020.

[8] H. Chen et al. Rfcam: Uncertainty-aware fusion of camera and wi-fi for real-time human identification with mobile devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(2):1–29, 2022.

[9] M. Ishige et al. Opt-in camera: Person identification in video via uwb localization and its application to opt-in systems. *arXiv preprint arXiv:2409.19891*, 2024.

[10] S. Mot. Aerodynamic and propulsion system design for a vtol fixed wing drone for multi-functional missions. M.sc. thesis, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal, Aug 2024.

[11] F. Alves. Structural analysis and prototyping in the development of a vtol fixed-wing uav. M.sc. thesis, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal, Dec 2024.

[12] T. Abreu. User interface for uav object detection and geolocation. M.sc. thesis, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal, Oct 2024.

[13] NVIDIA Corporation. Jetson orin nano super developer kit, 2025. Accessed: 2025-05-19.

[14] Reebot Robotics. Siyi zr10 gimbal camara, 2025. Accessed: 15 March 2025.

[15] B. Albano. An internet-of-drones architecture for context-aware aerial surveillance. M.sc. thesis, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal, Jul 2025.

[16] U.S. Department of Defence. Department of defense world geodetic system 1984: Its definition and relationships with local geodetic systems. second edition, 1992. Accessed: 2025-04-17.

[17] D. Vallado. *Fundamentals of astrodynamics and applications*. Microcosm Press, 2013.

[18] National Geospatial-Intelligence Agency. World geodetic system 1984 (wgs 84), 2012. Published by the United Nations Office for Outer Space Affairs.

[19] R. Hartley. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[20] E. Kaplan et al. *Understanding GPS: principles and applications*. ARTECH HOUSE, 2006.

[21] R. Maesschalck et al. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, 2000.

[22] A. Afonso et al. *Probabilidades e Estatística. Aplicações e soluções em SPSS. Versão revista e aumentada.* 10 2019.

[23] Pennsylvania State University. Lesson 4: Multivariate normal distribution. https://online.stat.psu.edu/stat505/lesson/4, 2025. Accessed: 2025-05-12.

[24] NVIDIA. Trafficcamnet, 2025. Accessed: 2025-05-22.