VOL.39 Nº1 2022
https://doi.org/10.22491/IMECTA.10848.en

# Remotely piloted aircraft system for autonomous detection of mobile ground units in restricted areas

ricardo Maroquio Bernardo[a], Luis Claudio Batista da Silva[b], Paulo Fernando Ferreira Rosa[c]
[a]Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo,
ricardo.maroquio@ifes.edu.br
[b]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca,
luis.silva@cefet-rj.br
[c]Instituto Militar de Engenharia,
rpaulo@ime.eb.br

*ABSTRACT: Surveillance of restricted areas and the respective maintenance of a security perimeter for installations sensitive to the presence of unauthorized persons is a necessary task in both civilian and military environments and can be conducted in different ways. This study presents a surveillance method of restricted areas in open environments via a remotely piloted aircraft system to autonomously detect and track mobile ground units entering a previously established perimeter to support a decision-making process regarding a possible intrusion event. The proposed solution has low acquisition and maintenance costs, easy deployment logistics, reduced risk to human lives, and decreased possible invasion detection time, in addition to being portable and ready to use.*

*KEYWORDS: RPAS. Robotics. Surveillance.*

*RESUMO: A vigilância de áreas restritas e a respectiva manutenção de um perímetro de segurança para instalações sensíveis à presença de pessoas não autorizadas é uma tarefa necessária tanto no meio civil quanto no meio militar e pode ser realizada de diversas formas. Este artigo apresenta um método de vigilância de áreas restritas em ambientes abertos que faz uso de um sistema de aeronaves remotamente pilotadas para detectar e rastrear, de forma autônoma, entidades terrestres móveis que adentrem um perímetro previamente estabelecido, de forma a apoiar um processo de tomada de decisão em relação a um possível evento de invasão. A solução proposta possui baixo custo aquisitivo e de manutenção, logística de implantação facilitada, redução do risco a vidas humanas e redução do tempo de percepção de uma possível invasão, além de ser portátil e de pronto emprego.*

*PALAVRAS-CHAVE: SARP. Robótica. Vigilância.*

## 1. Introduction

Surveillance of restricted areas consists of controlling access to areas sensitive to the presence of humans, animals, vehicles or other mobile ground units (MGUs). External areas without proper closure make this task even more challenging. A classic solution to this problem is human patrolling with or without the aid of vehicles, enabling the detection of attackers as soon as they enter a patrolman's view.

However, the number of patrol officers required to shorten the detection time during persistent surveillance makes this type of solution impossible. Moreover, this approach can generate conflicts between invaders and patrol officers, putting human lives at risk. An alternative to human patrolling are closed-circuit television (CCTV), enabling surveillance by monitors.

CCTV reduces risks to lives and requires fewer human resources; however, the need for operators' uninterrupted attention can lead to fatigue and to a subsequent failure [1]. Moreover, CCTV deployment requires adequate infrastructure and considerable installation time, inadequate for open areas and locations requiring temporary surveillance.

Mobile robots with sensory capability enhanced by cameras and computer vision algorithms can have relevant advantages to tackle invasive detection. However, the effectiveness of this method requires subproblems to be solved which constitute in itself a challenging subject, especially if the task is to be performed autonomously.

Good quality is one of the challenges inherent in the use of images captured from cameras embedded in mobile robots, especially in attempting to independently detect moving objects in image sequences obtained from mobile cameras since both the background and the moving object move between frames. Studies [2, 3, 4] show different ways of dealing with this problem.

Another important issue is that image processing and computer vision require significant processing power and directly influence the response time of the whole system. The computational resources typically available on computers embedded in mobile robots are quite limited, most often demanding the transfer of processing to an external computer.

Computational resources tend to become even scarcer (or even non-existent) with small aerial mobile robots. Their load capacity is even more limited and may even require the transmission of images for processing on an external computer.

Despite the challenges mentioned, unmanned aerial vehicles (UAVs) are a type of mobile robot whose use is becoming increasingly common both in civilian and military applications. Military use can be interesting in tasks such as surveillance and patrolling [5], search and rescue [6], mapping [7], supply [8], and communication [9]. The most common civil applications include package transportation [10], image acquisition [11], entertainment [12], and precision agriculture [13].

Recently, the rapidly growing use of UAVs in various applications can be explained by the ease of access to previously inaccessible equipment, whether for geographical, technical or economic reasons. Research has improved the accuracy, miniaturization, response time, and robustness of UAV building components, in addition to their high market availability and increasingly reduced cost, fostering their use and construction.

Thus, this study shows a relevant method of detecting and tracking mobile ground units in a restricted area previously defined by a human operator. As a differential, our method proposes the use of a remotely piloted aircraft of the quadcopter type specially designed for this purpose, which acts autonomously with a fully embedded processing software inherent to the fulfillment of its mission.

Following this introduction, Section 2 shows a brief theoretical foundation of the issues related to the intrinsic subproblems of the general solution. Section 3 covers some of the research related to this. Next, Section 4 exposes the formalization of the problem and the proposed solution, whereas Section 5 details the UAV prototype built for the actual experiments. Section 6 shows the diagrams and algorithms of the embedded software. Section 7 discusses some experimental results and Section 8 concludes with some considerations.

## 2. Theoretical framework

This section shows key concepts in mobile robotics, remotely piloted aircraft systems, and computer vision. The following subsections individually approach these matters.

### 2.1 Mobile robotics

Most techniques used in the proposed solution are relevant to mobile robotics. [14] synthesized the concepts we show. Mobile robots can be terrestrial, aquatic, and aerial. We adopted the latter as a platform to develop the proposed solution. Aerial robots can be fixed-wing, rotary-wing, or lighter than air. The solution proposed uses quadcopters, which are multi-rotor rotary aerial robots.

The degree of autonomy a mobile robot has to perform a task depends on its embedded computational, sensory, and actuation equipment. Environment perception is carried out by processing the data from sensors such as cameras and sonars. Some sensors may also be required for specific applications, such as $CO_2$ sensors or the camera itself.

Sensors can be classified under various criteria. **Field of view** corresponds to the coverage width of the sensor, usually expressed in degrees. **Range** is the maximum distance for reliable measurement. **Accuracy** indicates how correct the reading provided is against an exact reference. **Repeatability**, also treated as sensor accuracy, refers to the supply of the same measure under a given condition.

**Resolution** corresponds to the smallest possible difference between two sensor values.

**Energy consumption** concerns the current and voltage required by the sensor, also an important attribute. **Reliability** refers to its independence from external factors such as voltage variation. **Computational complexity** corresponds to the amount of computational processing required to process the data. Finally, **physical dimensions**, namely size and weight, are very relevant attributes for systems embedded in robots with space and payload restrictions.

Mobile robots can use multiple sensors to estimate their location relative to a fixed reference. Sensing errors and noise are common and can be treated with stochastic methods and representations based on belief levels. The Kalman filter is widely used to merge sensor data, promoting normalization or weighted combination [15]. Discretization of a continuous workspace can reduce the dimensionality of the problem, facilitating computation [16].

## 2.2 Remotely piloted aircraft systems

This subsection presents operational and constructive concepts of quadcopter-type UAVs, which this study treats as a type of aerial robot.

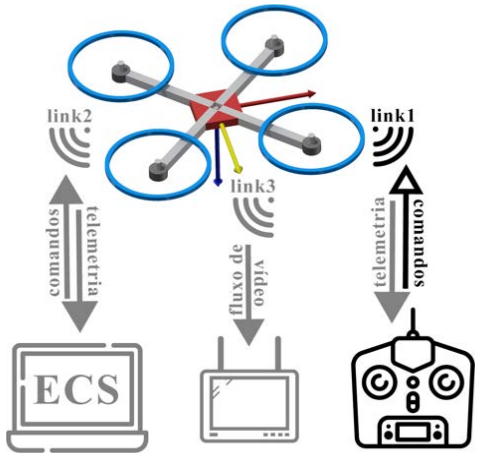


**Fig. 1** - Basic operation of a quadcopter-type UAV.

**Figure 1** shows a layout with the devices used during UAV operation and the types of messages exchanged between devices. The UAV must have a linked transmitter radio control (RCT) to enable its manual control. Optionally, the RCT can receive telemetry data such as battery voltage, altitude, and more. The center of the device is a real-time video stream preview screen transmitted by an embedded camera. ECS is the ground control station corresponding to the computer from which the mission operator can set high-level and emergency operational commands such as landing or route resetting.

**Figure 2** shows a schema with the main UAV components. Light gray elements are exterior to the vehicle. Yellow elements are not mandatory, but desirable. Dashed rectangles represent logical element groups, which do not necessarily mean they belong to the same printed circuit board. Arrows indicate the direction of data flow between the components.

The flight controller processing unit, represented by the central dashed rectangle, receives sensor data to compute the position and orientation of the UAV relative to a fixed reference. The processing unit is dedicated to running the aircraft control software. The right dotted rectangle represents the components used for specific applications, whereas the left one corresponds to the components used in first-person piloting mode, in which the operator views, in real time, images captured by an embedded camera.

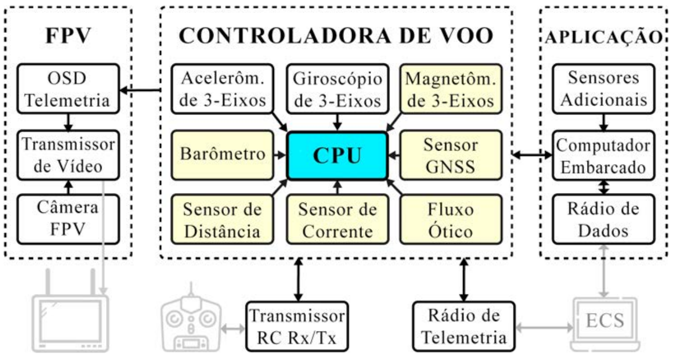


**Fig. 2** - Basic organization of the internal and external components of a quadcopter.

The propulsion system of a quadcopter consists of four engines equipped with propellers and controlled by electronic speed controllers (ESC). The correct combination of engine and propeller and ESC and battery is critical to defining flight quality.

The rotation differential between the four propellers defines the dynamic behavior of the aircraft. **Figure 3** shows some maneuvers according to the propeller rotation differential.

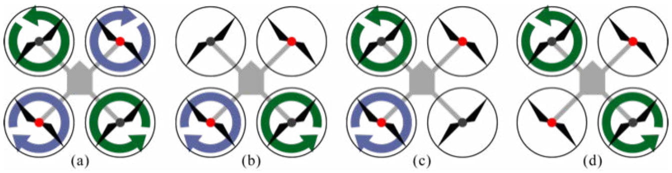


**Fig. 3** - Basic quadcopter flight dynamics depending on the rotation of each propeller. a) ascent; b) front pitch; c) right roll; and d) right yaw.

## 2.3 Computational vision

According to [17], computer vision is an area of study aiming to understand a 3D scene from 2D images, imitating human vision via software and hardware. Image quality is fundamental for this. It is common to implement vision systems into UAVs for tasks such as independently moving object (IMO) detection and tracking, position estimation, navigation, obstacle detection, autonomous landing, stabilized flight, attitude determination, 3D reconstruction, among others [18-25].

The extraction of image characteristics is essential to create descriptors representing specific image points. The atomicity of a descriptor, as well as its robustness to lighting and rotation variations, for example, enable operators to track it over multiple frames and get information about moving image parts.

There are different algorithms for computing point descriptors, and descriptor robustness usually incurs a longer processing time. Each algorithm has its advantages and disadvantages, and its performance is better or worse according to the situation.

IMO detection consists of segmenting images from a sequence in the background and mobile plane. IMOs stay on the mobile plane, whereas the action scenario is in the background. The main techniques for detecting IMOs are frame differentiation, dense or sparse optical flow, and background subtraction [26].

Frame differentiation considers the subtraction of two consecutive frames in a sequence. It has the advantage of being simple and fast, but the IOM is not completely detected as only part of it moves between two contiguous frames. In optical flow, vectors are plotted between corresponding descriptors from two subsequent frames and, according to the magnitude of these vectors, it is possible to have a sense of the regions of the image which are in motion, but without obtaining a complete IMO outline.

In background subtraction, an initial frame is maintained as a reference and is updated with each new frame, maintaining a viable background representation. With each subsequent frame, a subtraction is made, and the difference is considered an IOM. This method identifies IMOs clearly but requires proper updating of the reference frame. Tracing an IOM consists of recording its position over time and continuously identifying the corresponding objects between the frames in a sequence. Temporary problems, such as object occlusion, can be circumvented using prediction techniques.

## 3. Literature review

These studies [27-29], as this one, are some of the several cases in which the authors propose their own UAVs during the development of their research. **Table 1** shows the main characteristics of the aircraft built in the referenced studies. The available flight mode column refers to the most sophisticated flight mode offered by the UAV.

**Tab. 1** - Comparison of the main characteristics of aerial vehicle used in the referenced studies.

| Ref. | Total Weight (g) | Flight Time (min) | Available Flight Mode | Total Cost (US$) |
|---|---|---|---|---|
| [26] | 4000 | 11 | Stabilized | NA |
| [27] | 1400 | 5-10 | Stabilized | 300.00 |
| [28] | 2250 | 30 | Automatic | 1500.00 |

The Introduction mentioned several UAV applications, some of which may require IMO detection and/or screening. The following paragraphs briefly describe some studies which used UAVs to detect and/or track MGUs for some purpose. In this study, the term MGU is used in place of IOM to better characterize the target object.

Article [30], sharing three authors with this study, proposed a system to control several small UAVs to survey a predefined area. Its proposed solution aims to control the positioning and displacement of an aircraft within the target surveillance area, which is decomposed into hexagonal cells with different coverage priorities. A priority-based UAV distribution policy defines the visitation frequency of each cell. This solution enables a single operator to command a fleet of autonomous UAVs using a high-level user interface, dispensing the need for their individual manual control.

More recent studies tackle the problem of the persistent surveillance of restricted areas using UAV swarms. Article [31] is a theoretical study which mathematically analyzes the data collection capacity of a UAV swarm flying in circular, straight, and diagonal formations over an area decomposed into square cells, maintaining a uniform update rate of each cell assessed by the UAV.

The authors of study [32] combine the use of UAVs with unmanned land vehicles to cover urban areas, meeting the demands some locations have of detailed visual sensing. The proposed solution discretizes the UAV operation space into cubes and the land vehicle one into squares projected on the ground from these cubes. A path for cooperative vehicle action to solve the problem of coverage with restrictions is modeled as an optimization problem. Their solution uses a hybrid approach to genetic and estimation of distribution algorithms.

In [33], the authors show a solution using a swarm of commercial UAVs for semi-autonomous aerial surveillance. An operator manually defines UAV circular trajectories according to a method preventing the overlap of images captured by embedded cameras. These images are then continuously sent to a ground station which processes these images to detect objects of interest (which may not necessarily move). UAV amount and coverage frequency may vary depending on the size of the area.

Border surveillance via drones and a stationary ground sensor network is the subject of study [34]. A ground sensor network composed of probes and infrared sensors is used to detect movements in the environment. When it is detected, a UAV is triggered to go to the location and capture images. Such images are transmitted to a ground station that processes the images to find people.

For some applications, studies have addressed the establishment of UAV positioning policies. The authors of [35] propose a framework to position UAVs equipped with the proper devices to maximize the connectivity of backhaul networks, connecting backbones to peripheral networks. [36] establishes a method for positioning UAVs to provide a wireless network for users in a previously defined square ground area, seeking to maximize the number of users covered by the network, with the possibility of clustering user groups and employing multiple UAVs. The study by [37] is similar to [36] in its use of UAVs to provide connectivity and communication services for ground users. The authors of [37] focus on establishing an optimal operating altitude, considering the physics of signal propagation and soil roughness.

## 4. Problem formalization

**Figure 4** shows a high-level 3D representation of a simplified surveillance scenario in which a car enters a restricted area which includes a building. The walls around it define the restricted area. At the top is a surveillance UAV with a camera pointed down, continuously processing captured images. The camera field of view is represented by the semitransparent pyramidal volume between the UAV and the ground. Note that the field of view covers the entire circle representing the restricted area. On the road which gives access to the building, there is a vehicle within the restricted area, representing an MGU.

### 4.1 Preliminary considerations

In short, the elements relevant to the problem are the restricted area to be monitored, mobile ground units, UAVs, and their workspace. Thus, be $W$ an external environment comprising a set of georeferenced coordinates in which each $w_i \in W$ is a point on the ground given by [*latitude*, *longitude*, *altitude*]. The **restricted area** $A$ to be guarded is a manually defined subset of $W$ whose outline $C$ is a closed polygon of coordinates $c_i \in A$.

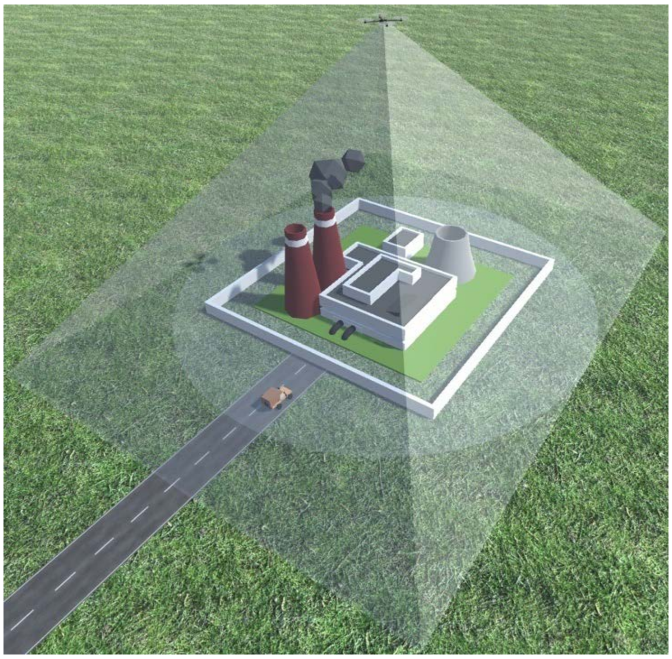


**Fig. 4** - 3D representation of a surveillance scenario.

The restricted area $A$ is decomposed into $n_d$ cells and represented by the set $D = \{d_i \mid 0 \leq i < n_d\}$ whose elements assume the coordinate values of cell centroids. The environment $W$ can contain a set of **mobile ground units** (MGUs), which can enter the restricted area at any time. Thus, $E = \{e_i \mid 0 \leq i < n_e\}$, whose elements correspond to centroid coordinates.

Initially, all MGUs are outside the restricted area, i.e., $\forall e_i \in E, e_i \in (W - A)$. When in this condition, MGUs are represented only to formulate the problem and have no corresponding elements in the concrete solution. The moment the MGU enters the restricted area, it becomes part of a subset of $E$ known as $E^t_{det}$, corresponding to detected MGUs.

If *UVA* is a VTOL-type (vertical takeoff and landing) **stationary UVA** with a starting position $P^0 \in D$, i.e., the *UVA* is within a cell of $D$ but not necessarily in its centroid, the *UVA* must be able to hover in a previously calculated position $P^{hover}$ whose ground projection belongs to A and, from that position, be able to perform a full visual coverage of $A$ via a camera.

Image $img^t$, of dimensions [$w$, $h$], obtained from the camera embedded in the *UVA* at a time $t$, is represented by $img^t = \{p^t(x,y) \in \Re^3 \mid 0 \leq x < w, 0 \leq y < h\}$, and each element $p^t(x, y)$ of the image stores the RGB values of the pixel in coordinate $(x, y)$ of the image captured at time $t$. The set formed by the sequence of images captured by the camera in the *UVA* is represented by *Img*.

## 4.2 Detection formalization

Detecting invasive MGUs requires individual processing of the frames captured by the embedded camera in the *UVA*. According to the formalization in the previous subsection, the set of images captured by the embedded camera in the *UVA* is defined by *Img*. With each image $img^t \in Img$, at $t = 1,..,t_{curr}$, in which $t$ is a temporal sequential index and $t_{curr}$ is the most current index of this sequence, the detection function $f_{det}$ must return, as output, a vector of $n^t_{det}$ pairs $(x^t_{\det^i}, y^t_{\det^i})$ corresponding to the coordinates of the pixels $p^t_{est}(x^t_{\det^i}, y^t_{\det^i})$ in the centroids $n^t_{det}$ of the IMOs detected in $img^t$, at $i = 1,\ldots,n^t_{det}$. Thus, the problem of detection can be represented by Equation 1 below:

$$FO = \sum_{1 \leq t \leq t_{curr}}^{NSites} PSI(Xd(i) = 1, i), \qquad (1)$$

In which $E^t$ is the set of MGUs detected in time $t$, $n^t_{det}$ is the number of IMOs detected at instant $t$, and $t_{curr}$ is the temporal sequential index of the most current image of sequence *Img*. Since the result of the detection function is a vector of centroids based on pixel coordinates, an additional function is necessary to convert these coordinates into georeferenced values, given that the navigation mechanism of the *UAV* is based on a satellite location sensor.

Thus, $f_{gps}$($x$, $y$, *lat*, *long*, *alt*, *cam*, *heading*) is the function converting a coordinate $(x, y)$ of a pixel from an image *Img* into a georeferenced coordinate, in which (*lat*, *long*, *alt*) are the georeferenced coordinates of the *UVA* at the time of image capture, *cam* corresponds to the intrinsic parameters of the camera, and *heading* is the orientation of the vehicle relative to the north. Note that IOM is the name given to independently moving objects detected in the image (which could be later characterized as MGUs).

### 4.3 Tracking formalization

To track invasive MGUs, it is necessary to find the correspondence between detected MGUs over time. Thus, let $f_{track}$ be the function capable of determining the matches between $E^{t}_{det}$ and $E^{t-1}_{det}$. For this, as input, $f_{track}$ continuously receives the result of $f^{t}_{\det}(img^{t}) = E^{t}_{det}$ at $t > 1$, the entities detected at the previous instant $E^{t-1}_{det}$, and the set of traces of each entity in $E^{t-1}_{det}$, named $R^{t-1}$.

As output, the function must return a set $E^{t}_{det}$ containing the updated set of invasive MGUs and the updated trace data set $R^{t}$ for each MGU set $E^{t}_{det}$. The tracking problem, thus, can be represented by Equation 2 below:

$$\sum_{i=1}^{NSites} Xd(i) = NAAA, \tag{2}$$

$$\left(E^{t}_{det}, R^{t}\right) \leftarrow f_{track}(E^{t}_{det}, E^{t-1}_{det}, R^{t-1}) \,|\, 1 \le i \le n^{t}_{det}, \quad 2 \le t \le t_{curr}$$

In which $E^{t}_{det}$ corresponds to the coordinates of all MGUs belonging to set $E_{det}$ at instant $R^{t}$; , the vectors of position coordinates over time for the whole $e^{i}_{det} \in E^{t}_{det}; (x^{t}_{e^{i}_{det}}, y^{t}_{e^{i}_{det}})$ at position $e^{i}_{det}$ at instant $t$; and $R^{t}_{e^{i}_{det}}$, the set $e^{i}_{det}$ of positions from instant 0 to instant $t$.

Note that maintaining the set $E^{t}_{det}$ updated corresponds to keeping only the MGUs present in restricted area $A$ at instant , i.e., $f_{track}$ must maintain a policy of including and excluding MGUs on $E^{t}_{det}$ since some MGUs $E^{t-1}_{det}$ may be absent from $A$ at moment $t$ and some MGUs $E^{t}_{det}$ may have just entered the restricted area and will be unmatched in $E^{t-1}_{det}$.

### 4.4 Restrictions, assumptions, and additional requirements

We considered that *UAV* had an automatic navigation capability by georeferenced waypoints. Because the system relies on RGB-type cameras, it is critical that the operation scenario has adequate lighting. We also considered that the terrain is flat and that the restricted area  is stationary, i.e., its location and perimeter remain constant throughout the mission. The trajectory between the *UAV* take-off point and surveillance position ignores obstacles with height higher than the operating altitude of the aircraft. The *UAV* is of type VTOL and able to communicate with a ground control station, which, in turn, is the interface with the mission operator.

## 5. Proposed Solution

**Figure 5** shows a simplified view of the Unmanned Aerial Vehicle System (UAVS) architecture used in the proposed solution. It consists of a stationary UAV and a ground control station communicating via a two-way channel. The UAV detects and tracks invasive MGUs using only the devices embedded in it, continuously transmitting MGU (position, speed, and size) and telemetry data (battery, position, tilt, and sensor status) to ECS.

Complementing **Figure 5**, **Figure 6** shows UAVS behavior throughout a mission, highlighting the high-level tasks involved in solving the problem and the sequence in which such tasks are performed. The dashed rectangle corresponds to the coordinates of the vertices which define a polygon corresponding to the perimeter of the restricted area provided by the mission operator as input data. The system then starts by computing the optimal position and orientation (pose) of the UAV to maintain full visual coverage of the restricted area.

With the computed pose, the UAV takes off and navigates until reaching it. Then, the MGU detection algorithm starts obtaining a frame of reference and processing the subsequent frames in search of changes which characterize the presence of IMOs in the image. With the detection results, the tracking algorithm maintains a temporal record of the positioning of the detected MGUs. Concluding a cycle, MGU static and dynamic information is transmitted to the ECS.

If the mission is to proceed, a new detection-trace-transmission cycle begins. The mission can be deliberately completed by the operator or automatically upon the UAV state. The following subsections provide details of the high-level tasks in this subsection.

## 5.1 UAV positioning

For the greater usefulness of the captured images, the *UAV* should hover over the restricted area *A* to adjust its field of view so the restricted area includes as many pixels as possible and maximize its pixel/area ratio. Thus, MGUs entering *A* will occupy more pixels, making detection easier. Since the camera sensor is rectangular to maximize the pixel/area ratio, a rectangle *Rect* is computed, whose minimum area surrounds the entire restricted area *A*.

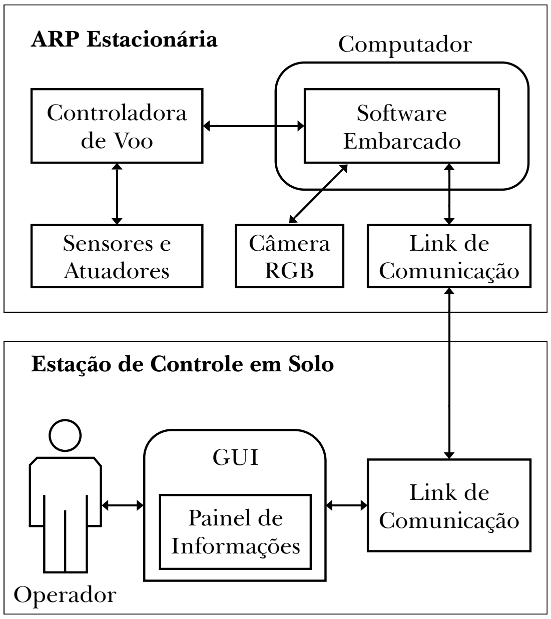


**Fig. 5** - Simplified UAVS architecture applied in the solution.

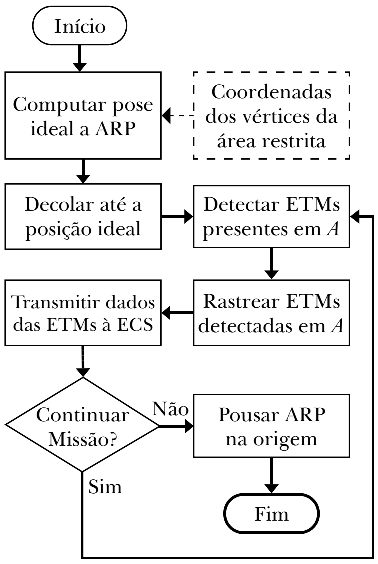

Início
Computar pose ideal a ARP
Coordenadas dos vértices da área restrita
Decolar até a posição ideal
Detectar ETMs presentes em A
Transmitir dados das ETMs à ECS
Rastrear ETMs detectadas em A
Continuar Missão?
Não
Pousar ARP na origem
Sim
Fim

**Fig. 6** - High-level view of the tasks performed by the UAVS.

Then, the *UAV* must match the latitude and longitude of the *UAV* center and the orientation must be defined so both the longest side of the computed rectangle and the captured image are parallel and their centers coincide. After computing the *UAV* pose, it is necessary to estimate the altitude at which the aircraft must hover, so its camera has full coverage of the restricted area. In this case, we considered that the parameters related to the camera lens are previously known.

According to Equation 3 [38], the dimensions $(W, L)$ of the coverage area of a camera gliding to the ground at an altitude $h$ can be estimated as follows:

$$\rho_f c_p \left( \frac{\partial T_f(r,t)}{\partial t} \right) = K_f \nabla^2 T_f(r,t) + Q(t)$$

(3)

$$L = 2h * \tan\left(\frac{\beta}{2}\right)$$

In which, α corresponds to the vertical opening angle of the camera and β, to its horizontal opening. Thus, the minimum altitude $h_{min}$ allowed for the *UAV* camera to fully cover *A* can be derived from Equation 3 as follows:

$$T_f(t) = e^{at} \left\{ T(0) + b\int_0^t e^{at'} Q\left(t'\right) dt' \right\}$$

$$T(0) = 1, a = \frac{Q_0}{\rho_f c_p \langle f \rangle}; b = \frac{1}{\rho_f c_p \langle f \rangle};$$

(4)

$$\langle f \rangle = T_f - \frac{Q_0}{} R_f{}^2;$$

These calculations disregard horizontal and vertical displacements caused by environmental weather and errors in the *UAV* sensors. Thus, we recommend a safety margin to avoid the possible lack of coverage of regions near the edges of *A*. **Figure 7** shows an image of a fictitious restricted area to be guarded.

In **Figure 7**, the polygon with dark green solid edges corresponds to restricted area *A*. The rectangle with yellow solid edges corresponds to the minimum rectangle surrounding the restricted area. The rectangle with blue dotted edges corresponds to the area covered by the *UAV* camera when the aircraft is in the previously computed pose, represented by the white circle with a red border. For comparison, the larger rectangle with yellow dotted edges corresponds to the image captured by the *UAV* camera when the aircraft is hovering in the horizontal position of the computed pose, but at a higher altitude and with a different orientation.

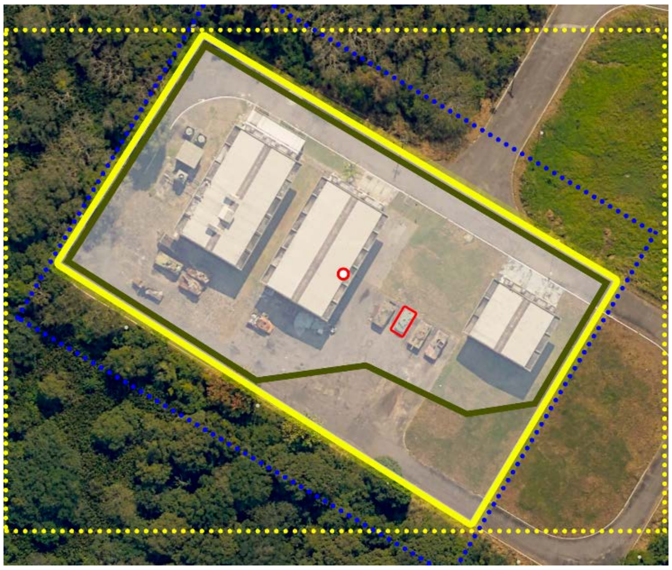


**Fig. 7** - Fictitious restricted area to be monitored and representations of the scope of the camera in different situations.

The rectangle with red edges within the restricted area corresponds to a random object of interest. This object occupies about 2.3% of the blue dotted rectangle and 1.4% of the yellow dotted rectangle, i.e., in the image captured in the ideal pose, the object would have approximately 64% more I, making it more "visible" in the detection phase of MGUs.

## 5.2 Image stabilization

Vibrations caused by engines and movement caused by weather or sensor inaccuracy are among the causes of problems in images captured using UAVs. Maintaining the alignment of frames captured by the camera is crucial for IMO detection algorithms. This study uses a hybrid solution based on mechanical stabilization with a motorized gimbal and software stabilization. The gimbal is triggered for angular scrolling and pitching movements. In [39], the same authors of this article developed this technique.

**Figure 8** shows an example of alignment that creates invalid edges: in *(a)*, the frame of reference, *(b)*, the frame to be aligned, and *(c),* the frame corrected by geometric transformation and the black edges, which emerged as a side effect. We should mention that the size of the black edges increases proportionally to the movements of the camera. Thus, maintaining UAV stability is of fundamental importance.

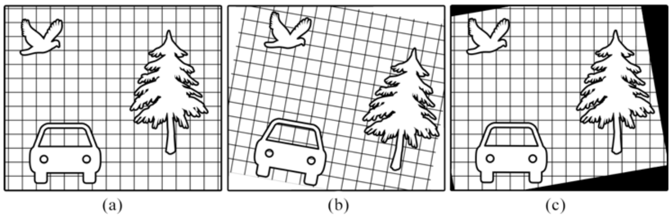


**Fig. 8** - Generation of invalid edges caused by geometric transformations applied to correct unwanted camera movements. (a) frame of reference; (b) frame to be aligned; (c) corrected table.

## 5.3 MGU detection and tracking

MGU detection and tracking considers that the optical axis of the embedded camera has the same direction and sense as the gravity vector, thanks to its image stabilization feature. Thus, the images captured from the ground are coplanar, facilitating MGU detection. The diagram in **Figure 10** provides an overview of the MGU detection and tracking algorithm.

A detection and tracking cycle starts from the **video stream** of the embedded camera. This cycle repeats for each processed frame. First, a frame is captured to serve as a **frame of reference**. Each new frame, represented in the diagram by the block with the **current frame** label, is first aligned so the image of the current frame coincides with the reference image. Then, the current frame is subtracted from the frame of reference and the result is submitted to a threshold which will determine whether each resulting pixel should be disregarded (0) or considered (1), generating a binary image.

Contiguous binary pixel groupings form binary large objects (blobs). Blobs larger than a second threshold are maintained, whereas smaller ones are considered outliers and are removed from the image. Finally, blobs are properly segmented from the background and stored in the current blob **collection**.

These blobs are the candidates for detected MGUs, beginning the tracking phase.

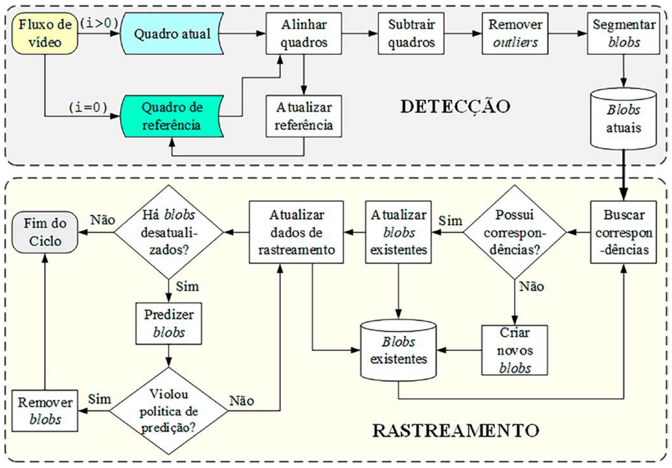


**Fig. 9** - Overview of the MGU detection and tracking algorithm.

In the first cycle, blobs will automatically constitute the collection of **existing blobs**. In the next cycle, current blobs are compared with those computed in the previous cycle to **search for matches** between blob collections. Thus, operators can maintain a history of blob positions over time.

In view of UAV frame processing rate, camera resolution, and altitude, blobs move by a few pixels between one frame and another. Thus, searching for a match consists of finding a neighboring blob with the same physical (aspect and area ratio) and behavioral (direction and speed) characteristics as the current frame blob. Thus, the search radius is a parameter influencing performance.

A new blob is any current blob which does not have a corresponding existing blob, i.e., one which has just entered the restricted area. Current blobs with matching existing blobs will have their tracking information and data updated. If any existing blob has no match, it may have left the restricted area or is experiencing occlusion.

Occlusion is handled via a prediction policy in which the next position of the occluded blob is computed according to its dynamic behavior, based on its position history. The prediction policy can delete a blob or update its tracking data with the new estimated position. Deletion can occur if the blob position indicates it has left the restricted area or if it is unmatched for a while above an adjustable threshold.

## 5.4 The built UAV

To conduct experiments on the intrinsic parts of the proposed UAVS, an UAV was built, whose characteristics meet all the premises of our problem. Our methodology evolved from a previous project, published in [40]. The UAV uses highly commercially available shelf components, which were carefully selected from empirical, individual, and integrated computerized tests.

The UAV built has a 330mm wingspan, a hovering range of more than 40 minutes, and a final weight of 1080g, whereas its predecessor had a 450mm wingspan, flew up to 23 minutes, and weighed 1.5kg. Evolutions in its sensors were also relevant. The autonomy gain is mainly due to the lithium-ion cell battery built specifically for this vehicle. We should also mention that the gimbal used for mechanical stabilization was built using shelf components. **Figure 9** shows a real photo of the UAV built for this study.

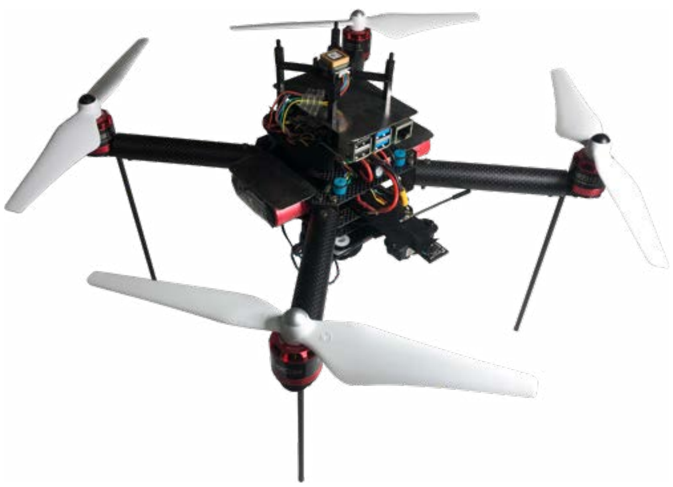
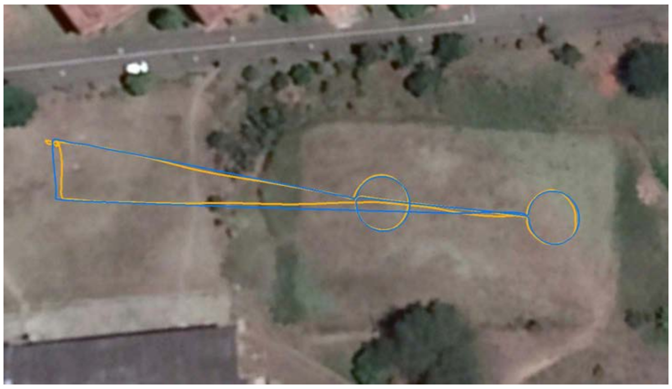


**Fig. 10** – 330mm UAV built to play the role of stationary UAV.

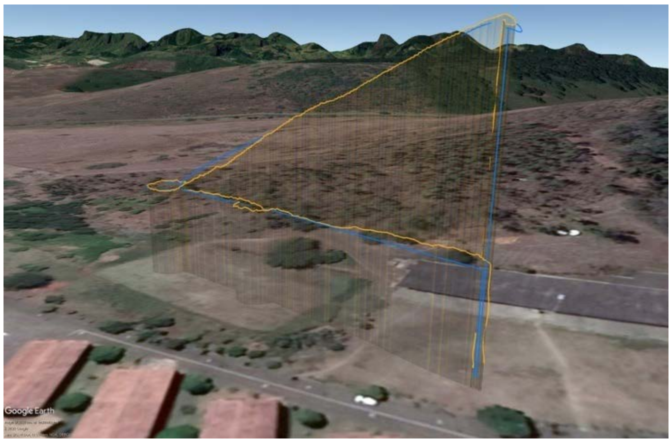
For navigation and control, a library has been developed with high-level functions for UAV movement, which is available in [41], based on another, of lower abstraction, called DroneKit [42], which uses a specific protocol, called MavLink [43], to communicate with small UAVs.

## 6. Experimental data

Due to the complexity of creating a real-world scenario to fully validate UAVS in a surveillance mission, the intrinsic parts of the system were individually validated. Real and computer-simulated flight tests were performed to validate the built UAV. Computer simulation used a Software in the Loop (SITL) architecture made available by the team developing ArduPilot [44], the flight controller firmware. **Figure 11** shows the results.



(a)



(b)

**Fig. 11** – Trajectories performed by the SITL-simulated UAV (blue dash) and the UAV in real flight (orange dash). (a) 2D visualization; (b) 3D display.

The graphs in **Figure 12** show the difference in latitude, longitude, and altitude between the actual flight experiments and SITL. The difference between the results is negligible for this type of application and clearly shows that UAV navigation accuracy enables SITL simulation for the initial validation of navigation algorithms.

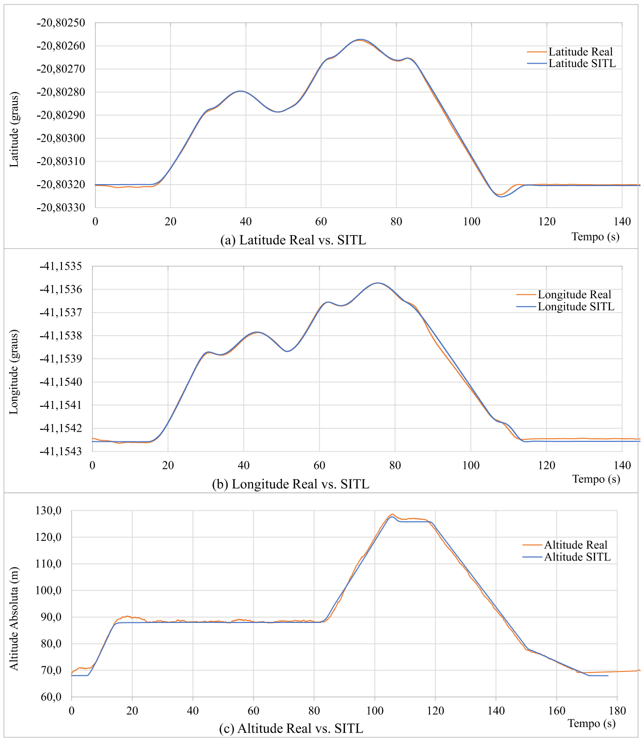


**Fig. 12** - Latitude (a), longitude (b), and altitude (c) of the trajectories generated by SITL simulation (blue line) and by the UAV in real flight (orange line).

To validate the detection and tracking algorithm, a computer simulator was designed to generate MGU animations entering a restricted area, experiencing occlusions, and overlapping other MGUs. **Figure 13** shows screenshots of the detection and tracking algorithm in action. The first capture (a) shows a green region representing the terrain; brown rectangles, occluded regions; and a white rectangle, an MGU. The second capture (b) shows a binary image with an MGU highlighted from the background. The third capture (c) shows the MGUs trails over time, including predicted positions.

The simulator also generates groundtruth data. The graph in **Figure 14** shows some qualitative detection and tracking results by comparing tracking data generated by the algorithm with groundtruth data generated by the simulator.

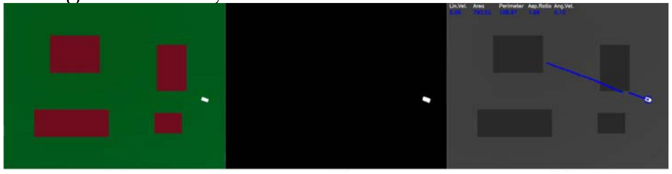


**Fig. 13** - Result of the tracking algorithm applied to the animation generated by the simulator. (a) original image; (b) image subtracted from the background; (c) computed tracking.

In the caption shown at the top of the graph in **Figure 14**, "GT" means "groundtruth;" "tra," "tracking;" "N.O.," "no occlusion;" and "W.O.," "with occlusion." The graph has eight lines representing trajectories traveled by the respective MGUs along the image, four generated by groundtruth data (thick lines) and four computed by the detection and tracing algorithm (narrow lines). Each generating point of such lines refers to the MGU centroid which traveled the trajectory represented by the line. The groundtruth-related line has been widened to facilitate comparison, as the chart has many overlaps.

The first experiment, shown in **Figure 14** by gray and yellow lines, corresponds to an MGU moving in a straight line without undergoing occlusion. The yellow line suffers small variations but fails to leave the central region of the groundtruth line. The second experiment, shown in **Figure 14** by the black and light blue lines, corresponds to an MGU moving in a straight line and suffering some occlusions. Computed tracking was also close to the groundtruth one, except for the highlighted region in which an occlusion occurred, which required a more intense use of the prediction mechanism.

The third experiment, shown in **Figure 14** by the red and white lines, simulated an MGU on a path with curves without suffering occlusion. Tracking was also very close to groundtruth. The fourth and final experiment, shown in **Figure 14** by the blue and green lines, simulated an MGU on a trajectory with curves and suffering occlusions at some points. In this fourth experiment, the computed trail was close to groundtruth, but it highlights a passage in which an occlusion caused a slight deviation in the trajectory estimated by the prediction policy.

To validate runtime, 16 animations were generated simulating groups of 20 to 500 MGUs simultaneously travelling in the scene. The animations were captured at three different video resolutions (1024×768, 800×600, and 640×480), representing different camera resolutions. **Figure 15** shows the relation between the number of MGUs and the frame rate per second of each tested resolution. Dotted lines are just to show a linear trend in the complexity of the algorithm.

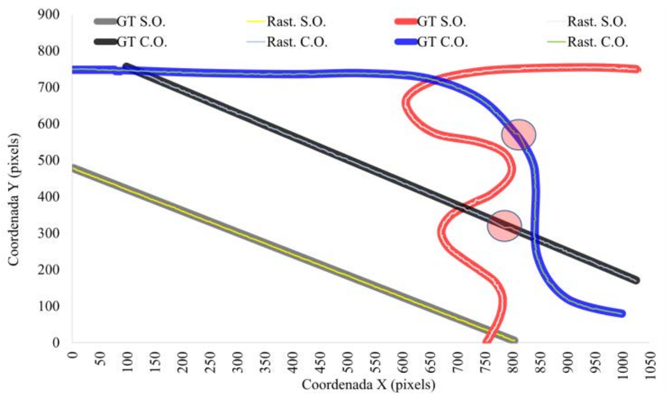


**Fig. 14** - Comparison between groundtruth trajectories and tracks computed by the detection and tracking algorithm.

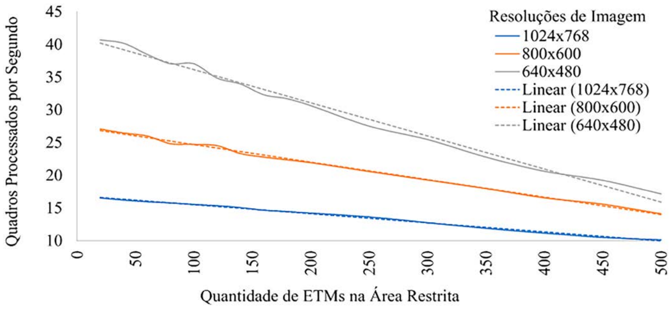


**Fig. 15** – Influence of the number of MGUs and video resolution on the rate of frames processed per second.

The experiments were conducted on a Raspberry Pi 4B with 4GB of RAM, whose temperature was maintained between 41 and 45 degrees Celsius. An expected observation is that the best case occurs with 20 MGUs, whereas the worst case, with 500 MGUs. For the 640×480 video resolution, the frame rate per second ranged from 17.1 (worst case) to 40.7

(best case), intaking a processing time per frame of 58ms and 24ms, respectively. At 800×600 video resolution, the frame rate per second ranged from 14.1 to 27.1, demanding 71ms and 37ms per frame, respectively. With the 1024×768 resolution, the frame rate per second varied between 10.1 and 16.5, demanding 99ms and 61ms per frame, respectively.

Note that the higher the video resolution, the lower the impact of the number of MGUs on processing time, indicating that the algorithm spends most of its time performing image processing and less time managing the MGUs in the scene. The linear trend and number of MGUs in the scene impacts runtime less than the other involved variables contribute to facilitate the dimensioning of the computational resources necessary for some specific application requiring, for example, surveillance of extensive areas and the use of higher resolution cameras.

Finally, we should mention that, in qualitative terms, the minimum size for detectable MGUs varies according to the number of pixels they occupy in the image and their contrast in relation to the background (soil). The proposed detection and tracking algorithm proved capable of tracking MGUs occupying at least four pixels in simulation-generated videos. However, in visually noisy real environments, the minimum size of detectable MGUs tends to increase as does noise intensity, and the establishment of this trend requires more specific studies.

## 7. Final remarks

This study showed a UAVS to autonomously detect MGUs in restricted areas in open environments, designed for situations in which conventional surveillance means, such as CCTV and human patrolling, are unfeasible due to establishment time, available structure, cost, and operational coordination. The proposed solution can bring relevant advantages to these aspects and reduce the risk to human lives by requiring only one remote operator.

Regarding the UAV, the experimentally obtained results show the feasibility of the prototype built to meet the established premises. It is a low-cost portable platform whose operation and maintenance is easy and thus useful for various applications. As for the developed software, performance results show that algorithmic complexity enables fully embedded real-time running without the need for transmitting images for external processing.

We should mention that the reduced flight time of portable quadcopter-type UAVs is an important restraint which would limit the useful time of the MGU detection system and, consequently, its feasibility. Some solutions can reduce this limitation. One possibility is using multiple UAVs to automatically replace stationary UAVs. Another is using power-tethered UAVs, as in [45]. A suggestion for future studies is the creation of an algorithm to determine MGU threat levels according to physical and behavioral characteristics such as its size, speed, movement pattern, and direction.

## Acknowledgements

This study was partially funded by the Coordination for the Improvement of Higher Education Personnel (CAPES).

## References

[1] HAERING, N.; VENETIANER, P. L.; LIPTON, A. The evolution of video surveillance: an overview. **Machine Vision and Applications**, v. 19, n. 5-6, p. 279–290, 2008.

[2] NAYAR, S. K.; NARASIMHAN, S. G. Vision in bad weather. In The Proceedings of the **Seventh IEEE International Conference on Computer Vision**. USA: IEEE, 1999. p. 820–827.

[3] WANG, H.; LI, S. Z.; WANG, Y. Face recognition under varying lighting conditions using self quotient image. In Proceedings of the **Sixth IEEE Int. Conference on Automatic Face and Gesture Recognition**. USA: IEEE, 2004. p. 819–824.

[4] HAUTIÈRE, N.; TAREL, J.P.; AUBERT, D. Towards fog-free in-vehicle vision systems through contrast restoration. In **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Piscataway, NJ, USA: IEEE, 2007. p. 1–8.

[5] JAKOB, E. S.M.; PAVLICEK, D.; PECHOUCEK, M. Autonomous UAV surveillance in complex urban environments. **IEEE Int. Joint Conferences on Web Intelligence and Intelligent Agent Technologies**, v. 2, n. 1, p. 82–85, 2009.

[6] NAIDOO, Y.; STOPFORTH, R.; BRIGHT, G. Development of an UAV for search & rescue applications. **IEEE AFRICON**, v. 1, n. 1, p. 1–6, 2011.

[7] GOTOVAC, D.; GOTOVAC, S.; PAPI, V. Mapping aerial images from UAV. In **IEEE International Multidisciplinary Conference on Computer and Energy Science (SpliTech)**, v. 1, n. 1, p. 1–6, 2016.

[8] WU, C.; QI, J.; SONG, D.; QI, X.; LIN, T.; HAN, J. Development of an unmanned helicopter automatic barrels transportation system. In **IEEE International Conference on Robotics and Automation (ICRA)**, v. 1, n. 1, p. 4686–4691, 2015.

[9] MERWADAY, A.; GUVENC, I. UAV assisted heterogeneous networks for public safety communications. In **IEEE Wireless Communications and Networking Conference Workshops (WCNCW)**, v. 1, n. 1, p. 329–334, 2015.

[10] GRIPPA, P. Decision making in a UAV-based delivery system with impatient customers. **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, v. 1, n. 1, p. 5034–5039, 2016.

[11] BESTDRONEFORTHEJOB**. Best Drone for The Job**: Pro Photography. 2021. Acesso: 28 ago. 2021. Disponível em: http://bestdroneforthejob.com/product-category/work-drones/pro-photography-drones/.

[12] LEAGUE, D. R**. The Drone Racing League**. 2021. 28 ago. de 2021. Disponível em: https://thedroneracingleague.com/.

[13] GEORGE, E. A.; TIWARI, G.; YADAV, R.; PETERS, E.; SADANA, S. UAV systems for parameter identification in agriculture. In **IEEE Global Humanitarian Technology Conference**: South Asia Satellite (GHTC-SAS), v. 1, n. 1, p. 270–273, 2013.

[14] SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to Autonomous Mobile Robots**. 2.ed. Cambridge, Massachusetts, USA: MIT Press, 2011. 472 p.

[15] SABATELLI, S. et al. A sensor fusion algorithm for an integrated angular position estimation with inertial measurement units. In: **Design, Automation & Test In Europe Conference & Exhibition (DATE)**, 2011, Grenoble, France. USA: IEEE, 2011. p. 1-4.

[16] FRIEDRICHS, Stephan et al. **The continuous 1.5 D terrain guarding problem**: Discretization, optimal solutions, and PTAS. 2015, Cornell University, Ithaca, New York, USA: arXiv preprint arXiv:1509.08285.

[17] SZELISKI, R. **Computer vision**: algorithms and applications. [S.l.]: Springer Science & Business Media, 2010.

[18] AJMERA, J.; SIDDHARTHAN, P.; RAMARAVIND, K.; VASAN, G.; BALAJI, N.; SANKARANARAYANAN, V. Autonomous visual tracking and landing of a quadrotor on a moving platform. In 2015 **Third Int. Conference on Image Information Processing (ICIIP)**. Piscataway, NJ, USA: IEEE, 2015. p. 342–347.

[19] KAKVAND, P.; JABERZADEH, M.; INALLOU, M. M.; ALBORZ, Y. Smart onboard UAV system: using computer vision system to find a movable and stationary target. In **2nd International Conference on Knowledge--Based Engineering and Innovation (KBEI)**. Piscataway, NJ, USA: IEEE, 2015. p. 694–699.

[20] ZHANG, C.; CHEN, J.; SONG, C.; XU, J. An UAV navigation aided with computer vision. In **26th Chinese Control and Decision Conference (CCDC)**. Piscataway, NJ, USA: IEEE, 2014. p. 5297–5301.

[21] GUPTE, S.; MOHANDAS, P. I. T.; CONRAD, J. A Survey of Quadrotor Unmanned Aerial Vehicles. In **2012 Proceedings of IEEE Southeastcon**. Piscataway, NJ, USA: IEEE, 2012. p. 1–6.

[22] PAZ, C.; PAINA, G. P.; CANALI, L. Visual homography-based pose estimation of a quadrotor using spectral features. **In 2015 Latin America Congress on Computational Intelligence (LA-CCI)**. USA: IEEE, 2015. p. 1–6.

[23] FOWERS, S.; LEE, D.J.; TIPPETTS, B.; LILLYWHITE, K.; DENNIS, A.; ARCHIBALD, J. Vision aided stabilization and the development of a quad-rotor micro UAV. In **2007 International Symposium on Computational Intelligence in Robotics and Automation (CIRA)**. USA: IEEE, 2007. p. 143–148.

[24] TANG, D.; LI, F.; SHEN, N.; GUO, S. UAV attitude and position estimation for vision-based landing. In **2011 International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT)**. USA: IEEE, 2011. p. 4446–4450.

[25] JIN, Z.; WANG, X.; PAN, Q.; MORAN, B. Optimal UAV localisation in vision based navigation systems. In **2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. USA: IEEE, 2016. p. 1021–1025.

[26] SUKANYA, C. M.; GOKUL, R.; PAUL, V. A survey on object recognition methods. **International Journal of Science, Engineering and Computer Technology**, v. 6, p. 48–52, 2016.

[27] POUNDS, P.; MAHONY, R.; CORKE, P. Modelling and control of a quad-rotor robot. In Australian Robotics and Automation Association Inc. In **Proceedings Australasian Conference on Robotics and Automation 2006**. [S.l.], 2006.

[28] NAVAJAS, G. T.; PRADA, S. R. Building your own quadrotor: A mechatronics system design case study. In **IEEE 2014 III International Congress of Engineering Mechatronics and Automation (CIIMA)**. [S.l.], 2014. p. 1–5.

[29] TARIQ, Abdullah; OSAMA, Syed M.; GILLANI, A. Development of a low cost and light weight UAV for photogrammetry and precision land mapping using aerial imagery. In **IEEE. 2016 International Conference on Frontiers of Information Technology (FIT)**. [S.l.], 2016. p.360–364.

[30] SILVA, L. C. B.; BERNARDO, R. M.; OLIVEIRA, H. A.; ROSA, P. F. F. Multi-UAV agent-based coordination for persistent surveillance with dynamic priorities.In **2017 International Conference on Military Technologies (ICMT)**, 2017, p. 765-771, doi: 10.1109/MILTECHS.2017.7988859.

[31] CHO, J.; SUNG, J.; YOON, J.; LEE, H. Towards Persistent Surveillance and Reconnaissance Using a Connected Swarm of Multiple UAVs. In **IEEE Access**, vol. 8, p. 157906-157917, 2020, doi: 10.1109/ACCESS.2020.3019963.

[32] WU, Y.; WU, S.; HU, X. Cooperative Path Planning of UAVs & UGVs for a Persistent Surveillance Task in Urban Environments. In **IEEE Internet of Things Journal**, vol. 8, no. 6, p. 4906-4919, 2021, doi: 10.1109/JIOT.2020.3030240.

[33] BANDARUPALLI, A.; SWARUP, D.; WESTON, N.; CHATERJI, S. Persistent Airborne Surveillance using Semi-Autonomous Drone swarms. In **Proceedings of the 7th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications**, 2021, p. 19–24, doi:10.1145/3469259.3470487.

[34] SHARMA, M. K.; SINGAL, G.;GUPTA, S. K.; CHANDRANEIL, B.;AGARWAL, S.; GARG, D.; MUKHOPADHYAY, D. INTERVENOR: Intelligent Border Surveillance using Sensors and Drones. In **6th International Conference for Convergence in Technology (I2CT)**. IEEE, 2021, p. 1-7.

[35] ABDEL-MALEK, M. A.; IBRAHIM, A. S.; MOKHTAR, M. Optimum UAV positioning for better coverage-connectivity tradeoff. 2017 **IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)**, 2017, p. 1-5, doi: 10.1109/PIMRC.2017.8292633.

[36] SUN, J.; MASOUROS, C. Drone Positioning for User Coverage Maximization. In **2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)**, 2018, p. 318-322, doi: 10.1109/PIMRC.2018.8580746.

[37] NASRAOUI, L.; ROY, S. Optimal UAV Positioning for Terrestrial Users. In **91st Vehicular Technology Conference (VTC2020-Spring)**, IEEE, 2020, p. 1-5, doi: 10.1109/VTC2020-Spring48590.2020.9128870.

[38] NAM, L. H.; HUANG, L.; LI, X.J.; XU, J.F. An approach for coverage path planningfor UAVs. In **IEEE 14th International Workshop on Advanced Motion Control (AMC)**. USA: IEEE, 2016. p. 411–416.

[39] BERNARDO, R. M.; SILVA, L. C. B.;ROSA, P. F. F. Onboard Video Stabilization for Low Cost Small RPAS Surveillance Applications. In **7th Brazilian Conference on Intelligent Systems (BRACIS)**, 2018, p. 450-455, doi: 10.1109/BRACIS.2018.00084.

[40] BERNARDO, R. M.; SILVA, L. C. B.; ROSA, P. F. F. Concepção de Uma Plataforma de VANT de Baixo Custo para Uso em Pesquisas.**V Workshop de Comunicação em Sistemas Embarcados Críticos (WoCCES)**. SBC, 2017.

[41] BERNARDO, R. Maroquio. **The DroneAPI Python Library**. Disponível em: https://github.com/maroquio/DroneAPI. Acesso em: 20 de janeiro de 2021.

[42] DRONEKIT. **Developer Tools for Drones**. 2021. 20 out. de 2021. Disponível em: https://dronekit.io/.

[43] MAVLINK. **Micro Air Vehicle Communication Protocol**. 2021. 20out. de 2021. Disponível em: https://mavlink.io/en/.

[44] ARDUPILOT**. Open-Source Autopilot Software System**. 2021. 20 out. de 2021. Disponível em: https://ardupilot.org/.

[45] PAPACHRISTOS,C.; TZES, A. The power-tethered UAV-UGV team: A collaborative strategy for navigation in partially-mapped environments. In **22nd Mediterranean Conference on Control and Automation**, 2014, p. 1153-1158, doi: 10.1109/MED.2014.6961531.