A Quadrotor Flight Mode For Tracking Based On Computer Vision Using ROS

Ricardo da Mota Salvador ∗ Leonardo de Assis Silva ∗∗
Vitor Henrique de Moraes Escalfoni ∗
Raquel Frizera Vassallo ∗

∗ Departamento de Engenharia Elétrica, Universidade Federal do Espírito Santo, ES, (e-mails: ricardomsalvador50@gmail.com, vitor.hme@gmail.com e raquel@ele.ufes.br)
∗∗ Coordenadoria de Eletrônica, Instituto Federal do Espírito Santo - Guarapari, ES, (e-mail: leonardo.assis@ifes.edu.br)

Abstract: Unmanned Aerial Vehicles (UAVs) are particularly interesting for covering large areas or observing regions from a privileged angle of view. However, many applications still require direct human control. The addition of visual feedback can enable these vehicles to perform tasks more autonomously. In this context, this work proposes an autonomous flight module, based on computer vision, with embedded processing, to follow a moving target in external environments. Interesting results are obtained through real tests done with a quadcopter that uses Pixhawk for low-level control and a Raspberry Pi for high-level control.

Keywords: Computer Vision, Visual Feedback, ROS, Unmanned Aerial Vehicle, Drone, Pixhawk.

1. INTRODUCTION

Due to its versatility, Unmanned Aerial Vehicles (UAVs) have been used for several applications in which advantages can be obtained when flying over a particular region to visualize or monitor it. Such applications, generally, demand visual information for its realization, thus requiring an embedded camera on the UAV.

Some of the most common applications are: air patrol (e.g., rescue teams support) (Aprville et al., 2014); equipment and structures inspection (e.g., wind generators, transmission lines, oil and gas pipelines inspection) (McAree et al., 2016; Sato and Anezaki, 2017); rural or urban areas monitoring (Reinecke and Prinsloo, 2017); and supporting ground teams or land vehicles on situations in which they have movement restrictions due to obstacles, ground or visual constraints, and risks of getting closer to a place of interest (Sa et al., 2014). The works just mentioned usually require a manual control for handling the UAV. Inserting an autonomous control would probably increase the UAVs functionality and optimize the developed solutions.

In many cases, whenever it’s desired to execute an application autonomously, visual information is commonly very useful. In Zakaria et al. (2016), for example, a visual servoing control was implemented using optical flow, which allows tracking moving target objects. However, the computational processing required for these applications usually demands a ground support base. In a different approach, the detection of markers can be used to indicate tasks to be performed (Carvalho et al., 2016) or attitudes to be taken (Nitschke, 2014). These works use the markers to perform homography calculation, which provides spatial information to plan the vehicle’s movement. The first one presents a simulation for landing according to the mark’s detection, and the second one performs a multi markers control system in an internal environment, with a ground station for processing data.

The possibility of autonomous control makes the equipment more comprehensive and modular, facilitating its insertion in larger projects, such as in the context of digital cities (Barreto and de Souza, 2017) and IoT (Motlagh et al., 2017). The vehicle ceases to require direct or indirect commands from humans operators and becomes a self-sufficient tool to accomplish a given task. In Butzke et al. (2016), it is proposed an autonomous control, using fixed markers and cooperation between aerial and ground robots. The research showed interesting results, however, it contemplates only internal environments, fact emphasized by its authors. This scenario also occurs in Lugo and Zell (2014), which developed a similar system, tested on different types of routes, but always in a controlled internal environment. This highlights the need for specific control approaches for large areas and/or areas under the influence of adverse environmental conditions.

In this context, this work proposes a flight control module capable of following a mobile target, while generates position and orientation commands for the UAV, referenced to a fixed local frame. This feature allows the equipment to be used in various environments and applications, such as cooperation with other robots or its integration into larger systems, such as Smart Spaces or even Smart Cities. Another goal of this work is to make the flight module capable of dealing with imperfections intrinsic to the system, such as the low GPS accuracy in certain situations, and external influences of the environment, like light variation during sunny days and/or the incidence of moderate winds on the
vehicle. Thus, the proposed flight module should allow the platform to chase moving targets in outdoor environments. To validate our work, experiments with several routes were carried out and the results are discussed further in this paper.

Our work is presented in this paper with six sections. Section 2 presents the description of the used quadrotor. The computer vision algorithms and the developed flight module are explained in detail in Sections 3 and 4. The results are presented and discussed in Section 5, and finally the conclusion and future work are discussed in Section 6.

2. SYSTEM OVERVIEW

We used as UAV, a quadrotor controlled by a Pixhawk board (Pixhawk, 2019), which is an open source platform. We also embedded a magnetometer, an accelerometer, a barometer and a GPS module, to enhance the system control.

Additionally, visual feedback is implemented, using an embedded microcomputer (Raspberry Pi 2 Model B+). In order to perform such procedure, and to communicate with the Pixhawk, the system is based on Linux (Ubuntu ARM) and adopts the framework ROS Indigo via MAVROS (MAVLink protocol). Images are acquired by a RaspiCam v1.3, connected to the Raspberry Pi, and a camera stabilizer (Tarot Gimbal) improves image acquisition.

Four 880 $K_v$ brush-less DC motors are powered by a 4 cell battery (5200 mAh), which gives the UAV a 15 minutes flight autonomy. This battery also powers the Pixhawk, the Raspberry and the Gimbal. Figure 1 shows the quadrotor, developed for this project.

![Quadrotor](image)

Figure 1. Quadrotor built for this project.

3. IMAGE ACQUISITION AND PROCESSING

The images are processed by the Raspberry Pi after the RaspiCam acquire them. The result is an attitude command to the UAV. The Pixhawk is responsible for sending such command and ensuring its execution. Therefore, to get a proper visual control, it is necessary to obtain good images for processing. This is often a hard task due to the strong influence of some elements such as vibration and electromagnetic interference caused by the motors.

To minimize such influence, a camera stabilizer (Gimbal) is used, attempting to maintain constant two of three orientation components of the RaspiCam (roll and pitch). However, adding this system changes the relationship between the reference frames of the camera and the UAV. Therefore it also changes the way the obtained data from each image and the sensors are associated. Section 4 fully discuss this topic.

The camera (RaspiCam) performs the acquisition of an image with 480p resolution and 10 fps rate. The 44.0 × 44.0 cm² marker on Figure 2 is adopted as the target for the tracking task. The acquisition is done with the ROS package raspi_cam_node (fpasteaus GitHub repository, 2015).

Once the pattern on the marker is detected, its orientation and position in the camera frame are estimated via homography. This is done by the ROS package ar_track_alvar (ROS.org., 2017) every time the marker is found in an image. If the detection fails, the system does not inform a new orientation and position data to the subsystem described in Section 4. Consequently, the microcomputer continues informing the Pixhawk the last point where the marker was detected as the destination for the UAV.

![Marker](image)

Figure 2. Marker adopted for tracking.

The size of the marker and the camera resolution were defined after analyzing: (i) the processing time for images with different resolutions, (ii) the maximum height from where the UAV is able to detect the marker, (iii) the target speed and (iv) the camera field of view. Such analysis is fully presented on Section 5.

4. DATA PROCESSING

After detecting the marker, the Raspberry Pi should send the Pixhawk a new pose for the UAV, in order to keep it flying above and aligned with the marker.

Because we use a camera stabilizer, the position and orientation estimated and returned by ar_track_alvar should be corrected before it is sent to the Pixhawk.

Figure 3 shows the reason for that: the visual data extracted from the marker $P$ and the UAV $D$ are not in the same reference frame.

While the orientation and position $D$ of the UAV are defined at take-off, in the global reference frame $W$, the visual data $P$ from the marker is represented in the camera frame $C$.

In order to work with those two types of information on a proper manner, both of them need to be compared in the same reference frame ($W$, due to convenience). This can be done by transforming the visual data, known in the camera frame $C$, to be represented in the global frame $W$, using a rotation matrix $R_{WC}$ and a translation matrix $t_{WC}$.

DOI: 10.17648/sbai-2019-111469
We may assume that the origin of camera frame $C$ is coincident to the origin of the UAV frame $D$, since the distance between $C$ and $D$ is insignificant when compared to the one between $D$ and $W$. Thus, we can consider that $t_{WC}$ is the position of the UAV in frame $W$.

However, because we use the Gimbal to not lose sight of the marker when the UAV moves, the orientation of the camera can not be directly considered as the same of the UAV. The camera is always looking down-wise. Therefore, the axis $X$ and $Y$, in frame $C$, are at the same plane $XY$ of the coordinate system $W$, sharing axis $Z$. The difference between these two frames, in terms of rotation, is only at yaw. The UAV yaw angle $\theta_D$ defines such rotation, that is, $R_{WC}$ is given by the following rotation matrix

$$R_{WC} = \begin{bmatrix} \cos(\theta_D) & -\sin(\theta_D) & 0 \\ \sin(\theta_D) & \cos(\theta_D) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

Thus, it is possible to find the marker position $X_W = (x_W, y_W, z_W)^T$ at $W$ from $C$ ($X_C$) using the relation

$$X_W = R_{WC}X_C + t_{WC}. \quad (2)$$

Finally, it is important to remark that the package ar_track_alvar uses a coordinate system $A$, similar to a plane coordinate system, which is different from a quadrotor. To solve that and to represent, in fact, the pose of the marker in $C$, two consecutive rotation must be performed, $180^\circ$ on axis $Y$ and $90^\circ$ on axis $Z$. Figure 4 shows this mechanism.

All these transformations together can be described as

$$X_C = R_Z(90^\circ)R_Y(180^\circ)X_A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} X_A, \quad (3)$$

where $X_A$ is the marker position on the ar_track_alvar frame.

In that way, it is possible to inform the Pixhawk the correct position $Q_W$, to where the UAV should go from the initial position of the mark, with

$$Q_W = X_W + \begin{bmatrix} 0 \\ 0 \\ z_o \end{bmatrix}, \quad (4)$$

with $z_o$ being the height from where the UAV sees the marker while tracking it.

For the orientation control, an important aspect inherent from quadrotors is considered. For the on-board controller, the only rotation that matters is the rotation around axis $Z$ (yaw). As a result, from three obtained orientation components (roll, pitch and yaw) only yaw is directly used.

The great advantage of using the Gimbal, is that, although it prevents the camera from being considered as a fixed part of the UAV, it keeps the position and yaw information identical between the camera and the quadrotor.

Therefore, once we have the UAV/Camera yaw ($\theta_D$ at $W$) and the marker yaw ($\theta_P$ at $C$), we can find the final UAV orientation $\theta_Q$ using the relation

$$\theta_Q = \theta_D + \theta_P, \quad (5)$$

as can be seen in Figure 5.

Figure 3. Scheme illustrating the real scenario and coordinate frames.

Figure 4. Frame changing transformations from coordinate system $A$ to coordinate system $C$.

Figure 5. Scheme illustrating the UAV final position and orientation control.
At the end of this process, the Raspberry Pi sends the position $Q_W$ and orientation $\theta_Q$ to the Pixhawk as the new desired pose, so the UAV can perform the tracking, following the target.

5. EXPERIMENTS AND RESULTS

In order to validate our system, we performed a set of outdoor flights, exposing the UAV to wind and light variations. For that, we attached the marker (Figure 2) to the top of a terrestrial robot (MobileRobots $^{TM}$ Pioneer 3-AT) and set the robot to perform three different trajectories: (i) a circular shaped, with 3.80 meters of radius and linear velocity of 600 mm/s; (ii) a zigzag shaped, with an angular velocity of 30 degree/s and linear velocity of 700 mm/s; (iii) and an eight shaped, with maximum velocity of 600 mm/s. Linear and angular velocities were determined according to the Pioneer’s limitations.

Tracking a moving target depends on variables such as the time needed to process the images, the UAV relative height, the speed of the target in the image and the field of view of the camera. To follow a fast moving target it is important to have a processing time as low as possible. But besides that, the camera field of view also influences the maximum speed that the target can take - the higher the UAV flies, the less chance of missing the moving target.

In this scenario, two crucial factors should be highlighted, the size of the marker and the resolution of the acquired images. For example, a small marker will present small details, which are challenging to detect when the UAV is flying higher. On the other hand, bigger markers can be inconvenient depending on the situation, e.g., moving in narrow passages. Hence, we decided to work with a marker that covers exactly the top of the robot, with $44.0 \times 44.0$ cm$^2$.

To find the most proper image resolution, we placed the UAV above the marker on different heights and tested two resolution values. Table 1 shows the detection rate of the marker in the images (assuming that the marker is always present on the scene) and the average processing time for both resolutions.

### Table 1. Detection.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Height (m)</th>
<th>Detection (%)</th>
<th>Processing Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>480p (640 × 480)</td>
<td>10m</td>
<td>100</td>
<td>104.4</td>
</tr>
<tr>
<td></td>
<td>15m</td>
<td>91.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16m</td>
<td>52.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20m</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24m</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>720p (1024 × 720)</td>
<td>10m</td>
<td>100</td>
<td>382.2</td>
</tr>
<tr>
<td></td>
<td>15m</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16m</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20m</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24m</td>
<td>82.7</td>
<td></td>
</tr>
</tbody>
</table>

Based on the results shown in Table 1, we have adopted a resolution of 480p, since the relative height of the experiments described in this paper is 10m. This value of resolution allowed better performance regarding the processing time, maintaining the same detection rate when compared to the higher resolution.

Once the marker parameters are defined, we can establish the experiment protocol. First of all, on manual mode, we take off the UAV and flight it until the target is detected. Afterwards, the visual feedback module is activated and it takes over the flight control. Our goal is to keep the horizontal distance between the UAV and the marker close to zero, and the height close to 10m ($z_0$), which is set by the user.

Figure 6 shows the terrestrial robot odometry, the UAV position and the expected trajectory for the circular shaped route. The position data are colored according to the control mode of the flight: manual mode, in red; visual mode (controlled by visual feedback), in blue. It can be notice that the UAV followed properly the marker, even in the presence of wind and light variation.

We can also note that the trajectory presents imperfections (noise), when compared to the expected one. Table 2 shows the horizontal and vertical errors quantitatively.

### Table 2. Trajectory error from GPS data, for the circular shaped route experiment.

<table>
<thead>
<tr>
<th>Error</th>
<th>Mean Error(cm)</th>
<th>Standard Deviation(cm)</th>
<th>Max Error(cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal</td>
<td>1.03</td>
<td>14.48</td>
<td>34.49</td>
</tr>
<tr>
<td>Vertical</td>
<td>42.44</td>
<td>38.37</td>
<td>117.11</td>
</tr>
</tbody>
</table>

However, it is worth mentioning that the UAV movement was reconstructed based on GPS data, which has intrinsic measuring errors, since there is an incertitude associated to the sensor - 2m, horizontally (Modules., 2011). Such phenomenon influences the results, which does not correspond to the real performance of the system.

---

1 This experiment was recorded and can be viewed on: [https://youtu.be/hB3IHAB07I0](https://youtu.be/hB3IHAB07I0)
To confirm that hypothesis, we recorded the experiment with a camera on the ground, at the center of the terrestrial robot trajectory. Figure 7 shows the expected route, in green; the UAV route, given by GPS data, in red; and the route calculated from the visual data extracted from the camera, in blue.

![Figure 7. Expected route, in green; UAV route, given by GPS data, in red; and route estimated from the camera images, in blue.](image)

As we can see on Figure 7, the real trajectory of the UAV presents less noise when compared to the one given by the GPS data. Even though there is a GPS error, the UAV follows the marker successfully. We may explain that by the following: when the GPS misses the UAV position at W (global coordinate frame), it equally misses the marker position. That is, the control is performed relatively between the UAV and the marker, reducing the interference caused by inaccurate sensor measurements.

Furthermore, we have analyzed the visual feedback controlling system, in which it is possible to obtain the difference between the marker and the camera position from ar_track_alvar (tracking error). From this data we have calculated the Euclidean distance between the current position and the one expected, the same coordinated X and Y, with 10 cm high (axis Z). The Euclidean error is calculated point-wise and shown in Table 3.

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>Mean Distance (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ZigZag</td>
<td>43.12</td>
</tr>
<tr>
<td>Circle</td>
<td>61.43</td>
</tr>
<tr>
<td>Eight</td>
<td>55.74</td>
</tr>
</tbody>
</table>

Table 3 shows that, as well as the trajectory error, the tracking error is also small. The distance from the UAV to the marker was, on average, 0.61 cm in X, 0.11 cm in Y and 10.007 cm in Z, which is near the expected. It can also be noticed that axis X presents the largest distance, due to the fact that the largest velocity component of the moving target, from the camera perspective, is on axis X in all of the experiments. That happens because the UAV tends to align itself to the orientation of the marker, adjusting the axis X of the camera and the robot moving direction. In the zigzag shaped experiment, the tracking errors are due to translational movement errors, because when the robot turns, there is only rotational movement, which maintains the marker in the image’s center, reducing the tracking to only the minimization of the difference between the vehicles angles.

Besides, we can imply that these obtained errors arise from the embedded vision system limitations, causing an estimated delay of 104.4 ms between the sample acquisition and the command given by the Raspberry (Table 1).

The behavior of the yaw angle given by UAV data in zigzag and circle shaped experiments is shown in Figure 8. Such data are colored according to the control mode: manual, in green, and visual feedback, in lilac.

![Figure 8. Behavior of the yaw angle given by UAV data in circle and zigzag shaped experiments.](image)

We can notice, from Figure 9(a), that the values of the yaw angle regarding the control mode, based on visual feedback, presents a linear increase, except on 0:17 min and 1:00 min. At such instants there is a step from +180° to −180°, due to the way the Pixhawk deals with the incoming angles, as described on Section 4. This linear behavior is a reflection of the circular movement performed by the robot, that is, the UAV does not only follow the position of the marker, but also its orientation.

Finally, on Figure 9(b), it is possible to note a similar phenomenon for the zigzag shaped experiment. The particularity of this trajectory relies on the fact that the UAV executes rectilinear and rotation movements separately. We can see that at instants 0:36 min and 0:51 min the UAV turns positively (to the right) and negatively (to the left), respectively. Between such instants, the yaw angle is almost unaltered, in other words, the UAV follows a
straight line. The small variations on this rectilinear sections (e.g., from 0:40 min to 0:51 min) are justified by external factors.

6. CONCLUSION AND FUTURE WORKS

In this article, an autonomous flight module for outdoor target tracking was proposed. The goal is to keep an UAV flying above a moving target marker regardless of the route it makes, through a controller based on visual feedback. The develop system should control the position and orientation of the UAV relative to the tracked marker. The proposed module was tested on external flights and presented satisfactory results, even in situations of wind and light variation. The visual control stands out for reaching an average horizontal error of 53.66cm and vertical error of 3.53cm (trajectory error), for all experiments, in addition to maintaining an approximate height of 10m from the target marker. Such a result can be considered promising and suggests that the proposed system provides the vehicle with the ability to track a moving target.

As this project works with a local reference for position control and final orientation, it allows the UAV to be easily included in larger projects, on which several robots share the same local reference, assisting, for example, applications of cooperation between robots or assuming the UAV as a component of a digital city, capable of meeting generic demands that share the same service needs provided by this type of agent.

In addition, mobile target tracking using computational vision demands a lot of processing power, especially when it comes to agile targets or that have a trajectory with abrupt variations of direction and speed. In this context, it is possible to identify a future need for the project, since the current processing power presents itself as a performance constraint.

One possible solution would be to transfer the embedded processing to a ground base, to which only the compressed image is sent and, in response, the UAV would obtain the desired position and orientation. With this optimization, the system should maintain its ability to cover large areas. However, it would need a wide-ranging data exchange technology such as Wi-Fi, WiMax or even GSM.

Another intended future work is the implementation of a search routine for the marker, both for mission start situations and for cases where the quadrotor loses sight of the pattern. A feasible approach for this search mode is to ascend the vehicle, in a simple or helical route, in order to gradually increase the field of view of the camera and also increase the chances of locating the marker.

ACKNOWLEDGMENT

The authors would like to thank Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES) for the financial support through Projeto Universal No. 504/2015.

REFERENCES


DOI: 10.17648/sbai-2019-111469