Hexacopter Control for Rescue Operations by Detection and Tracking of Visual Features

GRELLA Francesco, MORANDO Luca, SANCHES MOURA Mateus
Supervisors: Prof. Antonio Sgorbissa, Prof. Fulvio Mastrogiovanni

European Master on Advanced Robotics+ (EMARO+) - Robotics Engineering
Dipartimento di Informatica, Bioingegneria, Robotica, Ingegneria Dei Sistemi (DIBRIS)

Università Degli Studi Di Genova, August 2018

Abstract*—Given the importance of deploying autonomous agents in critical situations, such as rescuing in urban scenarios, this work presents a software architecture for hexacopter localization and stabilization with respect to visual features in the surrounding environment. An AscTec Firefly hexacopter equipped with a mvBlueFOX camera runs the proposed architecture on a Robot Operating System (ROS) framework. The target object is tagged with an AprilTags barcode, which is processed by the system and gives the object relative pose to the drone. A fixed Motion Capture system has been used as a backup data source for safety purposes. Preliminary experiments show that, while depending on prior visual object identification, the hexacopter is able to, when in an appropriate camera range, detect the target, compute its pose with respect to the drone frame, and approach a desired configuration with respect to the object. Incorporating computer vision techniques to detect and track generic objects allows applications in more realistic scenarios.

I. Introduction

Unmanned Aerial Vehicles (UAVs), in particular small and medium-sized drones, have been widely used for missions involving rescue and surveillance operations [1], due to their ability of reaching hazardous environments and providing important information to rescuing teams without posing additional risks to humans. Examples of UAV deployment in critical situations include firefighting [2], search and rescue operations following earthquakes [3] and hurricanes [4].

In this context, urban environments pose a significant challenge for rescues due to the complex, dense set of features (e.g.: buildings, cars) that compose the surroundings. In addition, as a drone navigates through such areas, occlusion may affect its localization with loss or degradation of GPS signal. Whether deploying a single agent or a fleet, it is thus convenient to utilize alternatives based on fusion of the internal navigation system data with visual odometry and other information from the available sensors [5].

With the increasing processing power of UAV boards and a wide range of on-market models that come with good quality cameras, the application of online Visual Servoing techniques has been popularized, and a comprehensive description of the recent works on this topic is found on [6]. Within the scope of this work, a Position-Based Visual Servoing (PBVS) approach enables not only robot localization with respect to a given feature, but also its stabilization, complementary to the odometry-based control which tends to degrade over time.

The goal of our work is to analyze and propose a base architecture for the problem

of having an hexacopter being capable to localize and stabilize itself with respect to the environment, determining and fixing its pose with respect to visually distinguishable regions of interest. This corresponds to a necessary initial step for the planning and actions required for the rescue operations in which the robot may be involved, which falls beyond the work scope.

The article structure is as follows: Section I contextualizes and introduces our work, while Section II delimits its scope. Section III presents and details the architecture developed for such problem, and Section IV covers how the experiments were set. Section V presents and discusses the implementation results, and a conclusion is given on Section VI.

II. Scope of Work

In this work, we assumed the hexacopter operation on an indoor, controlled environment with a Motion Capture system, capable of delivering precise position and orientation data to the workstation, which was used solely with the purpose of setting backup procedures to ensure safety. Only one object was present at the environment, which was considered the reference object for localization and stabilization. As a premilinary step on our study, the object was uniquely identified with a barcode.

Figure 1: The AscTec Firefly hexacopter.

The robot used was the Firefly hexacopter from Ascending Technologies [7], seen in Figure 1. It is equipped with a computer board, powered with an Intel Core i7 processor, with Linux and ROS (Indigo) installed, allowing an onboard architecture implementation. The robot's high performance embedded controller, accessible through the ROS interface installed, allows an easy and versatile operation of the hexacopter through different control modes, in this work being a position-based control.

Additionally, two USB mvBlueFOX cameras from Matrix Vision [8] were mounted, one of which was used for providing the image data for feature tracking and detection. Figure 2 shows the camera used, together with lenses adopted for widening the scene view, thereby facilitating the object detection within the workspace.

Figure 2: mvBlueFOX Camera with optical lenses.

III. System Architecture

An overview of the architecture developed is shown on Figure 3. It consists in the acquisition of two streams of data with a Publish/Subscriber pattern, primarily from the camera attached to the hexacopter, and secondarily from the Motion Capture system, and which are processed by the components shown in blue. The yellow box corresponds to the main component developed for this architecture and which integrates the visual feedback with the control interface, represented as the green box in the diagram.

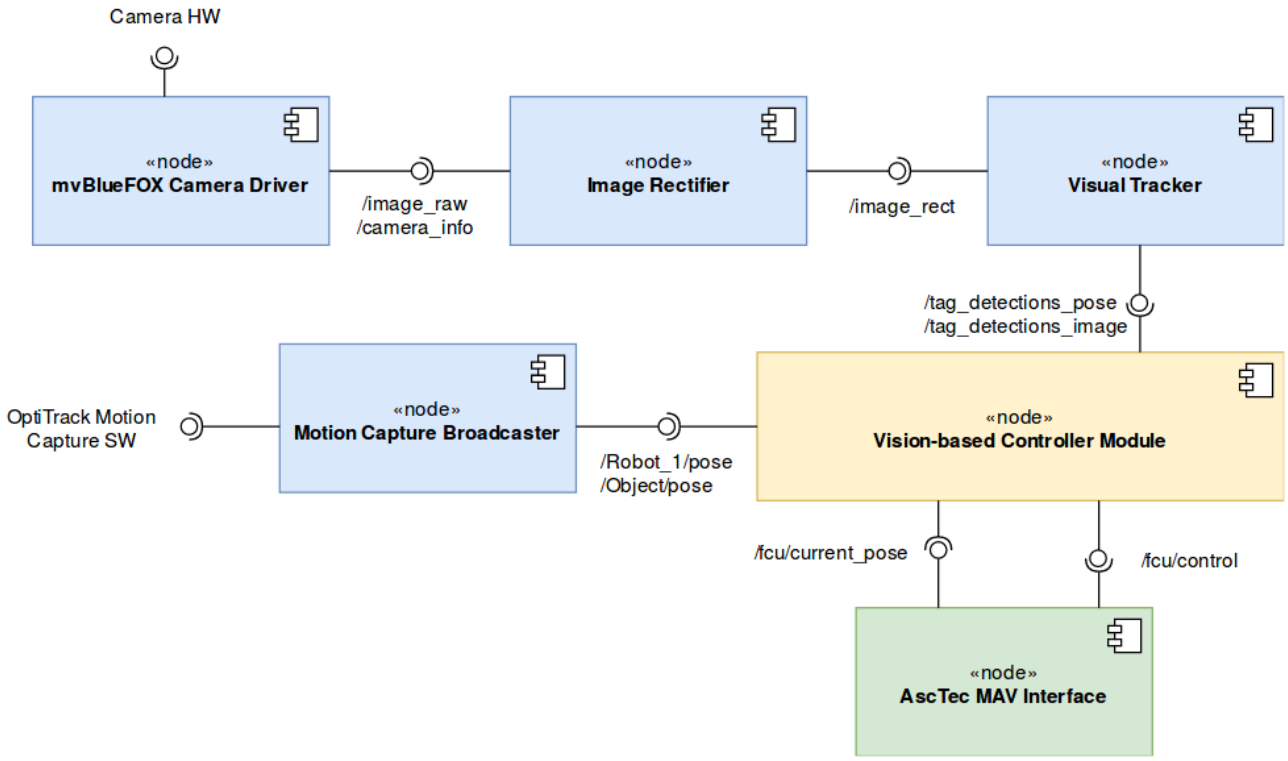

Figure 3: Software Components Diagram of the proposed architecture, displaying the ROS topics published.

To acquire data from the mvBlueFOX camera, the ROS package `bluefox2` from Kumar Robotics [9] is used for retrieving the video frames captured by the camera (on the topic `[device]\image_raw`, along with the informations given by camera calibration information (`[device]\camera_info`), which contain the intrinsic and extrinsic parameters of the camera, as well as its distortion correction parameters.

The calibration was initially performed with the `cameracalibrator` ROS node from the `image_pipeline` framework [10]. The adjustment of camera gains and exposure was done automatically by dynamically setting the `bluefox2` node parameters and saving in a configuration file with the application `dynparam`. Prior to that, it was necessary to manually substitute the camera lenses (seen in Figure 2) to widen its optical field of view. After the calibration and image processing, the resulting image frames are then rectified by the ROS package `image_proc`, also from `image_pipeline`, which outputs the topic `image_rect` mapped onto the camera namespace.

The visual detection and tracking component is based on the package `apriltags_ros` [11], which searches for two-dimensional barcodes from the AprilTags [12] visual fiducial system. The choice of AprilTags as target features is based mainly on the speed and reliability of their tracking, as less information needs to be processed from such simple barcodes. When an AprilTag from a given family is detected in the rectified image, its pose with respect to the camera frame is computed and published by the detector node onto the topic `tag_detections_pose`. Figure 4 shows an example of the detection (shown by the red circle) from a given image frame during experiments.

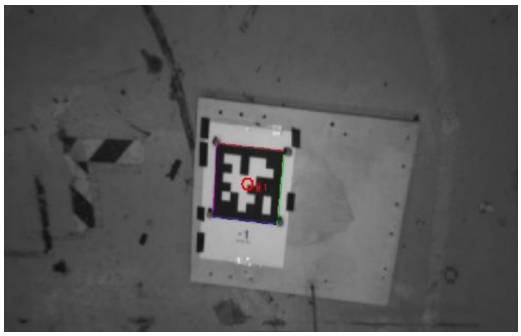

Figure 4: An identified object being detected by the camera after the take-off phase.

Alternatively, the node `mocap_node` from the `mocap_optitrack` package [13] streams the OptiTrack Motive data to `tf` and `geometry_msgs/pose` topics, based on a configuration file that maps the rigid bodies created on the software to the topic identifiers. It is required that the running workstation's IP be sent to Motive and that the option "Data Streaming" be turned on for the data to be published.

The node `Controller Module`, developed by the group for this architecture, integrates the information from the visual tracker, when available, and computes a desired pose with respect to the object frame, allowing the hexacopter to hover the object at a predefined height. The translation between camera and drone center is also considered, such that the control always aims to have the object tag be stabilized at the center of the images from the camera. In case of absence of visual feedback, for example, when the object is out of sight, the motion capture information is taken and a secondary target pose is set according to the backup strategy considered. Nevertheless, the node publishes a control message of type `fcu\control`, compatible with the hexacopter ROS interface, with a rate of 5 Hz. Figure 5 provides an illustration on the frame transformations performed to apply the control.

The aircraft control component, located in the `asctec_hl_interface` package,

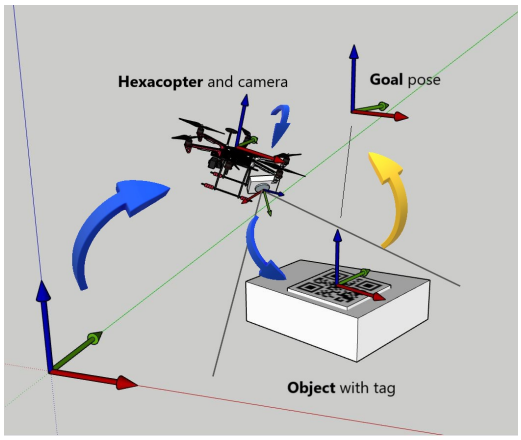

Figure 5: Illustration on the relative pose acquisition and goal pose computation. The absolute frame shown is only used to match the control command sent to the drone internal reference frame.

part of the `asctec_hl_comm` framework [14], receives the desired absolute pose from the visual-based controller node and performs internally a position control based on the odometry information of the hexacopter. It is worthy to state that the initial hexacopter position is set to coincide with its position from the motion capture area, such that a global reference can be established and appealed to when necessary.

IV. Experimental Setting

The experiments were conducted in the Motion Capture area of the EmaroLab, at the University of Genova. To ensure a safe operation, the workspace, fully covered by the Motion Capture cameras, was constrained to the dimensions of $2.14\text{m} \times 2.60\text{m} \times 1.50\text{m}$, even though the motion capture area was larger. Figure 6 contains a snapshot of the workspace and the initial setup.

The experiment procedure consisted in turning on the hexacopter board and controller, connecting it to the local network and having the drone board been accessed by a workstation through the SSH protocol. In sequence, Motion Capture markers were attached to the hexacopter body, and a rigid

body object was created and exported by the OptiTrack software to the hexacopter's IP. A single launch file executed all the required ROS nodes to run the developed architecture.

During the actual experiment, the object was moved around the workspace to verify whether the drone could follow it. The rectified image used for tracking and the current and desired trajectories were recorded using `rosbag`. Another video was recorded from an external device, to analyze the behavior of the robot both from an external view and from the on board camera point of view.

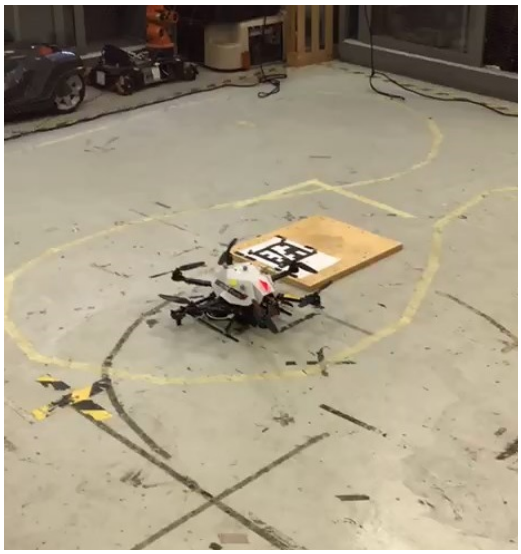

Figure 6: Experimental setup. The Motion Capture cameras that cover the workspace are not visible.

Initially, the hexacopter takes off and rises at a speed of 5cm/s until a height of 1.30m, in order to ensure a comprehensive view of the workspace ground. Then, if an object is detected, the hexacopter enters in a hovering phase and attempts to stabilize itself on a given height relative to the detected tag, while following the tag orientation. If the detection is not successful, or the object tracking is lost, the drone initiates a recovery routine and increases its height until the object tag is detected again, or until the workspace limits are reached. Similarly, in case of object recovery, the height is decreased slowly to ensure a stable tracking.

In any case, should the drone reach outside the designed operative region, a safe backup routine is triggered and a reset pose is sent to the controller, based on the Motion Capture information. The drone must also be hovering the object at a minimum distance of 50cm to prevent collision. While the take-off and hovering phases are executed autonomously, the landing procedure is manual, for safety reasons.

Prior to the actual experiment, initial tests aimed to adjust, refine and complete the architecture, by analyzing `rosbag` data and consequently improving the scripts according to the pursued results.

V. Results and Discussion

Considering the Motion Capture system as the reference source of data and also providing the reference frame, Figure 7 shows the drone height relative to the object in comparison with the desired one, while Figures 8 and 9 display, respectively, the x and y positions of the hexacopter and the object to be tracked during the experiment. From start to roughly 20 seconds, the hexacopter was in its take-off phase, while after 90 seconds, it lost track of the object, and then a landing procedure was taken. Thus, the actual tracking period corresponded to between 20 and 90 seconds.

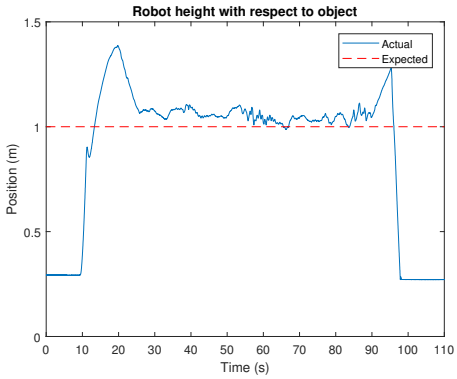

Figure 7: Hexacopter actual height (blue) and desired height (red dashed).

Despite observed fluctuations, most likely due to the low frequency of control commands sent to the interface, the hexacopter behaved well, with an average ground position error of 0.20m during the tracking phase. The system was able to reach and maintain a relative height close to the expected one, even when the object was moving.

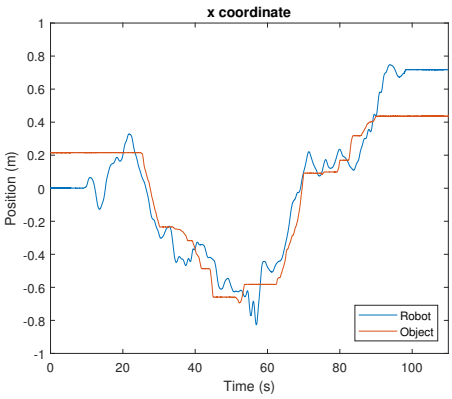

Figure 8: Position (m) in the x coordinates of the hexacopter (blue) and target object (red).

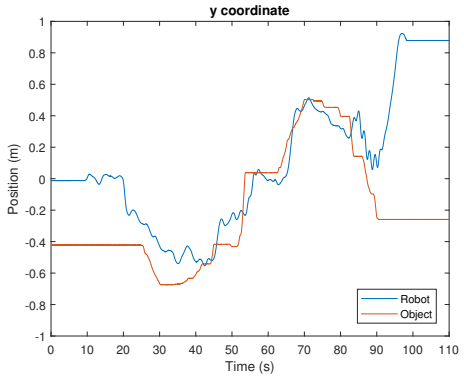

Figure 9: Position (m) in the y coordinates of the hexacopter (blue) and target object (red).

In addition, the angle of the hexacopter and the object frames with respect to the Motion Capture x-axis are displayed in Figure 10. It can be seen that, even after rotating the object in about 270 degrees, the drone reacted fast to orientation changes.

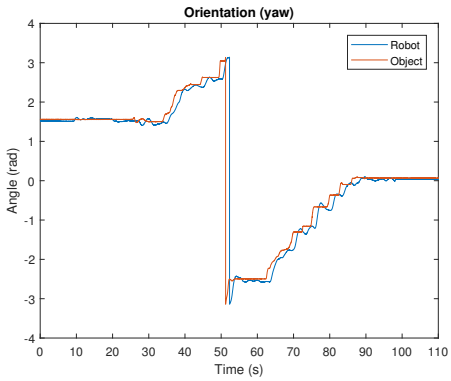

Figure 10: Orientation (yaw - in radians) of the hexacopter (blue) and target object (red).

VI. Conclusions and Further Work

From the results of the preliminary experiments discussed on Section V, the architecture addressed in this work and runned onboard allowed the hexacopter to identify its own pose with respect to a given object within the workspace, and also to stabilize itself while reacting to changes in object position and orientation. This implies that the proposed architecture, within the limitations imposed by the visual tracker and the drone controller, is suitable to operations in dynamic environments.

A possible improvement on the dynamic response of the hexacopter would be changing from position to velocity or acceleration control, preventing fluctuations such as the ones seen in the results and enhancing its stabilization. However, this also requires a careful design and tuning to avoid aggressive maneuvers that could lead to an unsafe state.

Currently, the developed architecture allows solely the detection and tracking of features from the AprilTags gallery, thus limiting its application to identified objects that would mostly be located in well-posed scenarios. Adapting the system to a real-world environment, where objects of different shapes and textures prevail, requires tracker improvement with more elaborated Computer Vision methods, such as template matching or object recognition by learning.

REFERENCES

[1] A. Birk, B. Wiggerich, H. Bülow, M. Pfingsthorn, and S. Schwertfeger, "Safety, security, and rescue missions with an unmanned aerial vehicle (uav)," *Journal of Intelligent & Robotic Systems*, vol. 64, no. 1, pp. 57–76, 2011.

[2] C. Yuan, Z. Liu, and Y. Zhang, "Uav-based forest fire detection and tracking using image processing techniques," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pp. 639–643, IEEE, 2015.

[3] J. Qi, D. Song, H. Shang, N. Wang, C. Hua, C. Wu, X. Qi, and J. Han, "Search and rescue rotary-wing uav and its application to the lushan ms 7.0 earthquake," *Journal of Field Robotics*, vol. 33, no. 3, pp. 290–321, 2016.

[4] J. Yeom, J. Jung, A. Chang, and I. Choi, "Hurricane harvey building damage assessment using uav data," in *AGU Fall Meeting Abstracts*, 2017.

[5] Y. Watanabe, A. Veillard, and C. Chanel, "Navigation and guidance strategy planning for uav urban operation," in *AIAA Infotech@ Aerospace*, p. 0253, 2016.

[6] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for uavs: Current developments and trends," *Journal of Intelligent & Robotic Systems*, vol. 87, no. 1, pp. 141–168, 2017.

[7] A. Technologies, "Asctec firefly." http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-firefly/, 2018 (accessed July 21, 2018).

[8] M. Vision, "mvbluefox usb 2.0 camera." https://www.matrix-vision.com/USB2.0-industrial-camera-mvbluefox.html, 2014 (accessed July 21, 2018).

[9] K. Robotics, "bluefox2 package." https://github.com/KumarRobotics/bluefox2, 2018 (accessed July 21, 2018).

[10] V. Rabaud, "image_pipeline package." http://wiki.ros.org/image_pipeline, 2013 (accessed July 21, 2018).

[11] M. Wills, "apriltags_ros package." http://wiki.ros.org/apriltags_ros, 2016 (accessed July 21, 2018).

[12] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, IEEE, May 2011.

[13] C. Robotics, "mocap_optitrack package." http://wiki.ros.org/mocap_optitrack, 2014 (accessed July 21, 2018).

[14] M. Achtelik, "asctec_hl_comm package." http://wiki.ros.org/asctec_hl_comm, 2016 (accessed July 21, 2018).