UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

GROUND TARGET CHASING WITH A UNMANNED AERIAL VEHICLE

A Degree Thesis

Submitted to the Faculty of the

ETSETB, Universitat Politècnica de Catalunya

And

Purdue University

by

Daniel Mas Montserrat

In partial fulfilment

of the requirements for the degree in

AUDIOVISUAL SYSTEMS ON TELECOMMUNICATIONS ENGINEERING

Advisor: Edward J. Delp

Co-advisor: Luis Torres Urgell

September 2015

Abstract

Ground target chasing systems using Unmanned Aerial Vehicles are increasingly used in aerial photography and film recording. In most of the cases GPS-based system is not useful and the use of computer vision is required. In this work an alternative to common GPS-based systems is presented using the open-source ArduPilot technology. In this work, the video is transmitted by radio signal and is processed in an external computer that also controls the movements of the vehicle. The system is compound by a tracker based on the mean-shift algorithm that combines information of the color, information from visual features (ORB) and motion information of the video. The output of the tracker is used by a set of PID controllers that control the movements of the vehicle. This work presents good results and it can be a good start point for a commercial product.

Resum

Els sistemes de seguiment de blancs terrestres mitjançant vehicles aeris no tripulats són cada cop més utilitzats en fotografia aèria i en rodatge de pel·lícules. En molts casos l'ús de sistemes basats en GPS no son útils i s'utilitza la visió per computador. En aquest projecte, una alternativa als sistemes basats en GPS es presentada. La tecnologia usada, de codi lliure, és ArduPilot. La senyal de vídeo es transmesa per radio i es processada en un ordinador extern que també s'ocupa de controlar els moviments del vehicle. El sistema esta format per un *tracker* basat en l'algoritme *mean-shift* que combina la informació del color, de *visual features* (ORB) i informació del moviment. La sortida del *tracker* es utilitzada per un conjunt de controladors PID que s'ocupen de moure el vehicle. Aquest treball presenta bons resultats i es un bon punt de partida per a un producte comercial.

Resumen

Los sistemas de seguimiento de blancos terrestres mediante vehículos aéreos no tripulados son cada vez más utilizados en fotografía aérea o en rodaje de películas. En muchos casos los sistemas basados en GPS no son útiles y se utiliza la visión por computador. En este proyecto, una alternativa a los sistemas basados en GPS es presentada. La tecnología usada, de código libre, es ArduPilot. La señal del video es transmitida por radio y procesada en un ordenador externo que también se ocupa de controlar los movimientos del vehículo. El sistema está formado por un *tracker* basado en el algoritmo *mean-shift* que combina la información del color, de *visual features* (ORB) y del movimiento de las imágenes. La salida del *tracker* es utilizada por un conjunto de controladores PID que se ocupan de mover el vehículo. Este trabajo presenta buenos resultados y es un buen punto de partida para un producto comercial.

Acknowledgements

A special thanks to Professor Edward J. Delp who has given me the opportunity to work in his laboratory these months in this project. I am also grateful to Prof. Luis Torres, for his assistance and supervision over all the development of the project. A truly thanks to all the VIPERs, for always been willing to help me at any moment. Finally a heartfelt thanks to my family for support me all this months.

Revision history and approval record

Revision	Date	Purpose
0	10/06/2015	Document creation
1	24/07/2015	Document revision
2	13/08/2015	Document completion
3	20/08/2015	Document approval

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Daniel Mas Montserrat	dmas93@gmail.com
Edward J. Delp	ace@ecn.purdue.edu
Luis Torres Urgell	luis.torres@upc.edu

Written by: Daniel Mas Montserrat		Reviewed and approved by: Edward J. Delp	
Date	13/08/2015	Date	20/08/2015
Name	Daniel Mas Montserrat	Name	Edward J. Delp
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract 1
Resum 2
Resumen 3
Acknowledgements 4
Revision history and approval record 5
Table of contents 6
List of Figures 8
List of Tables: 9
1. Introduction 10
1.1. Formulation of the problem 10
1.2. Statement of purpose 10
1.3. Project background 11
1.4. Project requirements and specifications 11
1.5. Project plan 11
2. State of the art of the technology used: 12
2.1. Introduction to quadcopters 12
2.2. Introduction to ArduPilot 14
2.2.1. Flight modes 15
2.2.2. Radio Control (RC) Channels 16
2.2.3. Ground Control Station (GCS) 17
2.3. State of the art on ground target chasing 18
3. Methodology / project development: 19
3.1. Development environment 19
3.2. System design 20
3.3. Tracker development 22
3.3.1. Color Based 23
3.3.2. Feature Based 24
3.3.3. Motion Based 26
3.3.4. Tracker combination 27
3.4. IBVS development 29
4. Experimental testing and Results 35
4.1. Tracker testing 35
4.2. IBVS testing 39

5. Budget 40
6. Conclusions and future development: 42
Bibliography: 43
Appendix I: Hardware specifications 44
Appendix II: Project Plan 46
Glossary 49

List of Figures

Figure 1 Work packages 11
Figure 2 Examples of quadcopter movements 12
Figure 3 Types of vehicle supported by ArduPilot 14
Figure 4 Radio controller used in this project 16
Figure 5 Mission Planer GUI 17
Figure 6 Ground target chasing system description 21
Figure 7 Block diagram of the ground target chasing system 21
Figure 8 Color-Based tracker block diagram 24
Figure 9 Feature-based tracker block diagram 26
Figure 10 Motion-based tracker block diagram 27
Figure 11 PID controller block diagram 31
Figure 12 GUI for tuning the PID values 33
Figure 13 Target appearance and selection 35
Figure 14 Moving target being tracked 36
Figure 15 Combination of the three trackers 36
Figure 16 Combination of the three trackers 37
Figure 17 Target is lost when disappearing of the scene 37
Figure 18 Target appearance and selection 37
Figure 19 Combination of the three trackers 38
Figure 20 Combination of the three trackers 38
Figure 21 Camera moving while following the target 39

List of Tables:

Table 1 Main flight modes.....15
Table 2 Radio Channels of ArduCopter 16
Table 3 Main ground control station softwares.....17
Table 4 Weights of the different trackers.....28
Table 5 Initial, minimum and maximum value of the different parameters of the PID 33
Table 6 Components, hardware and software budget.....40
Table 7 Human resources design cost.....40
Table 8 Design cost.....41

1. Introduction

1.1. Formulation of the problem

The use of drones or Unmanned Aerial Vehicle (UAV), and specially quadcopters, has abruptly increased over the last decade. They have been widely used on military and security operations as it provides safety to the pilot.

Nowadays they are not only used in military or security applications, but also in aerial photography and movie filming. Drones are an affordable alternative to manned helicopters for recording extreme athletes, as it can be professional snowboarders or surfers, or for recording aerial shots for a movies. A big community of hobbyist has originated due to the availability of low-price quadcopters.

While recording the video, usually is required to follow the athlete, actor or target that is being recorded. This is usually made by placing a radio transmitter in the desired target. This transmitter constantly communicates its position to the drone. This solution has two main drawbacks: first, it doesn't work in zones where GPS signal is restricted, and secondly, the target being followed need to wear a radio transmitter. Wearing a radio transmitter can be a problem in some situations, i.e. for water-sports athletes, security or surveillance applications, and actors. To overcome this problems a new approach is presented in this project, changing the use of the GPS signal by the use of a camera and video processing.

1.2. Statement of purpose

The purpose of this project is give autonomy to an UAV using computer vision. The UAV should be able to flight without constant human interaction and detect and chase a target as it can be pedestrians or vehicles. To achieve this purpose, a commercial UAV and an embedded camera are used, in our case, IRIS by 3DR and GoPro Hero 3+ are our main devices. The software developed is used with an open-source library.

The project main goals are:

- Understand the main aspects of the behavior of UAV, especially of quadcopters
- Develop an autonomous system with a UAV, a camera, a radio frequency communication and a computer.
- Develop algorithms to control and move the UAV without the need of human pilot.
- Develop a tracking system to follow the ground targets.

1.3. Project background

This project starts from scratch and it's independent from the other works realized in VIPER lab. The main ideas for the project are provided and supervised by the supervisor Professor Edward J. Delp.

1.4. Project requirements and specifications

The project requirements are:

- Maintain the UAV (the IRIS) completely functional and updated.
- Provide pedestrian, vehicle or any ground target chasing using video signal from the GoPro camera.
- Adapt the system to the limitations and specifications of the hardware used (Appendix 1)

The project specifications are:

- The UAV should be able to autonomously follow targets by using the aerial images.
- A human interaction is initially required to select the target to be followed.
- The system is designed to work with quadcopters based on Autopilot, especially on IRIS.
- The software is designed to run on Ubuntu 14.02 with OpenCV 2.4.11, Python 2.7, Mavproxy and DroneKit API.

1.5. Project plan

This project was structured by 3 main work packages: the first work package consist in the understanding and maintenance of the UAV. It includes all the modifications applied to the drone and its respective calibration. The second package consist in the developing of a tracking algorithm. Finally, the third work package consist in the implementation of an Image-Based Visual Servoing (IBVS) Controller.

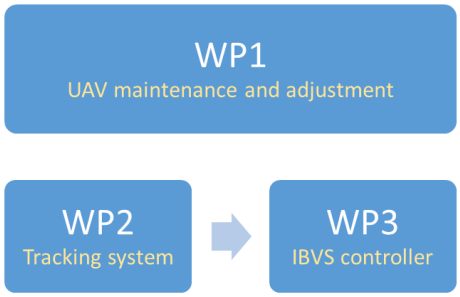


Figure 1 Work packages

A more detailed content of the work packages and grant diagram can be found at appendix 2.

2. State of the art of the technology used:

A brief review of the behavior and state of the art of the quadcopters is presented. This is followed by a presentation of Ardupilot and it ends with a summary of the works previously done related with ground target chasing

2.1. Introduction to quadcopters

An unmanned aerial vehicle (UAV), also known as drone, is an aircraft without a human pilot aboard. They are usually deployed for military and security applications, but also used in a growing number of civil applications such as policing, firefighting, and non-military work, such as aerial photography.

UAVs typically can be classified depending its functionality:

- Military and special operation
- Logistics
- Research and development
- Civil and Commercial UAVs

This work is focused in commercial and research UAV. The emergence of low-cost drones in the last decade has led to an increase of commercial UAV used for aerial photography, filmmaking, extreme sports video recording and other applications based on computer vision. Commercial UAV are usually multirotor helicopters such as quadcopters, hexacopters and octocopters, with 4-, 6- and 8-rotor helicopters, respectively.

Quadcopters are based on two sets of identical fixed pitched propellers; two clockwise and two counter-clockwise. The control of vehicle motion is achieved by altering the rotation rate of one or more rotor discs, thereby changing its tilt and/or rotation.

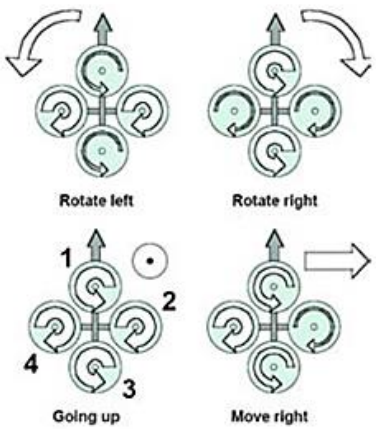


Figure 2 Examples of quadcopter movements

This vehicles use an electronic control system and sensors, such as gyroscopes, to stabilize the aircraft. With their small size and agile maneuverability, the quadcopters can be flight indoors as well as outdoors. The set of hardware and software that ensembles the signals from the sensors and stabilize and pilot the drone is usually called autopilot.

Quadcopters are a useful tool for researchers to test and evaluate new ideas in a number of different fields, including flight control theory, navigation, real time systems, robotics, and computer vision. Nowadays, swarms of quadcopters can autonomously perform complex flying routines such as flips or darting through hula-hops.

There are numerous advantages in using quadcopters as versatile test platforms. They are relatively cheap, available in a variety of sizes and their simple mechanical design means that they can be built and maintained by amateurs.

There is a lot of projects and companies focused in quadcopters, most of them with content open source, open hardware, or both. Some of them are listed here:

- OpenPilot: Flying robot framework with open hardware and open software based on STM32 microcontroller
- ArduPilot: Flying robot framework with open hardware and open software based on Arduino.
- ARdrone: Quadcopter technology with public API. It has become a popular tool in research and education, especially in visual-based autonomous navigation, autonomous surveillance and human-machine interaction. Mainly is used in in-door flights.
- DJI: Quadcopter with public API for third party developers.

There are several companies developing technologies for quadcopters, some of them for recreational applications, as AR, 3DR, DJI, Nixie (a wearable drone), or for industrial applications such as PrecisionHawk, DragonFly or Aeryon.

2.2. Introduction to ArduPilot

ArduPilot/APM is an open source autopilot system supporting multi-copters, traditional helicopters, fixed wing aircraft and rovers. The source code is developed by a large community of enthusiasts.

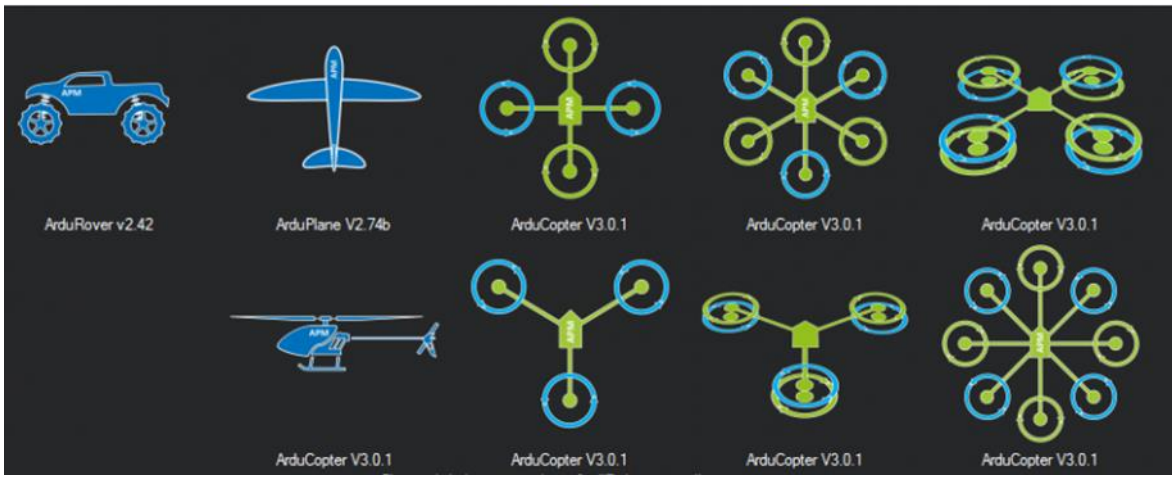


Figure 3 Types of vehicle supported by ArduPilot

Currently ArduPilot supports the following autopilot boards: Pixhawk, PX4 FMU, VRBrain, FlyMaple, BeagleBone Black, APM1 and APM2.

The main flight code for ArduPilot is written in C++. Support tools, such as APIs or Ground Station software, are written in a variety of languages, most commonly in python.

ArduPilot is released as free software under the GNU General Public License version 3 (or later) and is supported by the company 3DR, how also builds different UAV such as IRIS, the one used in this project.

The main features that ArduPilot can provide are:

- Multiple flight modes.
- Failsafe programming options when lost control signal or low battery conditions.
- Three Axis camera control and stabilization, shutter control and live video link with programmable on-screen-display.
- Data transceivers allow real-time telemetry and control between your ground station device and APM.
- Full data logging provides comprehensive post mission analysis, with graphing and Google Earth mapping tools.

There are two main ways to control the ArduPilot devices: with a RC controller or with a ground station (laptop, tablet, smartphone…). Ground station and the radio controler send RC signals to the device using a protocol called MAVLink that allows fully control of the drone and also the transmission of information like the state of the battery, relative altitude, GPS location, velocity, gyroscope information (tilt, roll and pitch) or flight mode. The communication goes in both ways, the quadcopter sends and receives information from/to the RC Controller or ground station.

2.2.1. Flight modes

Depending of the flight mode, the drone will react differently to the same RC signals. There are several flight modes:

Main flight modes	
Stabilize	Stabilize mode allows you to fly your vehicle manually, but self-levels the roll and pitch axis.
ALT_Hold	In altitude hold mode, the copter maintains a consistent altitude while allowing roll, pitch, and yaw to be controlled normally.
Loiter	Loiter automatically attempts to maintain the current location, heading and altitude. The pilot may fly the copter in Loiter mode as if it were in manual. Releasing the sticks will continue to hold position.
RTL	In return to launch (RTL) mode, the copter navigates from its current position to hover above the home position.
Auto	In Auto mode the copter will follow a pre-programmed mission script stored in the autopilot

Table 1 Main flight modes

Besides this 5 main flight modes there are some secondary modes: Acro, Sport, Drift, Guided, Circle, Position, Land, Follow Me, Simple and Super Simple.

2.2.2. Radio Control (RC) Channels

RC controller and ground stations send the RC signals to the vehicle over different channels:

Nº Channel	Parameter
Channel 1 or Ailerons	Roll
Channel 2 or Elevator	Pitch
Channel 3 or Throttle	Speed
Channel 4 or Rudder	Yaw
Channel 5	Flight Mode
Channel 6	Camera pitch
Channel 7	WP Saving / Camera roll
Channel 8	- (Usually not used)

Table 2 Radio Channels of ArduCopter

Functions of channel 5 to 8 can be configured for other purposes. The values sent to the vehicle, usually called the Pulse Width Modulation signal (PWM), oscillate between 1000 and 2000. It is recommended to calibrate the RC controllers to have a better performance.

Channel 1 and 2 are usually controlled with the right stick in the RC controller and channel 3 and 4 with the left stick.

Figure 4 Radio controller used in this project

2.2.3. Ground Control Station (GCS)

A Ground Control Station (GCS) is a software able to pilot, assist and monitor the vehicle. There are different ground control stations options in the market, with different prices, characteristics and devices:

Name	Free/Payment	OS	Description
Mission Planner	Free	Windows	Allows you to plan, control and analyze all aspects of a flight. Device calibration tools are available. Python code can be executed.
APM Planner	Free + open source	Windows, Linux, Mac	Open source software similar to Mission Planner.
Tower	Free	Android	3DR android app to control the device using a user friendly interface
Andropilot	Free + open source	Android	Android-based ground control station that focuses on simplicity and robustness.
UGCS	Payment (Free version)	Windows	Plan your mission for single or multiple vehicles, all from a single piece software. Focused in professional use.
QGroundControl	Free + open source	Windows, Linux, Mac	QGroundControl is an alternative ground station written in C++ using the Qt libraries
MAVProxy	Free + open source	Windows, Linux, Mac	Command line based ground station. Used for development use. Is the base for droneAPI.

Table 3 Main ground control station softwares

In this project, the ground station software used are Mavproxy and Mission planner. The communication between the computer and the vehicle is stabilized using a telemetry kit included with IRIS. A 915 MHz receiver/transmitter is connected to the computer or laptop via USB.

Figure 5 Mission Planer GUI

2.3. State of the art on ground target chasing

Computer vision and autonomous flight are two fields widely studied. There are a lot of works on autonomous flight by path following using aerial images or even flight stabilization using the video signal [14],[15],[16],[17].

In the specific area of ground target following, several works with successful results has been implemented. Most of them are based on ARdrone technology [1],[2],[3],[4],[5], but there is also some works based on ArduPilot [6],[7].

Most of them follow the same structure, the same as follows this project: a visual tracker follows detects the position of the target and a controller autonomously moves the drone to the target.

3. Methodology / project development:

3.1. Development environment

The software is developed using Python 2.7 and OpenCV 2.4.11 running in Ubuntu 14.04. The main base of the software is based on MavProxy and DroneKit.

MAVProxy is a fully-functioning GCS for UAV's. The intent is for a minimalist, portable and extendable GCS for any UAV supporting the MAVLink protocol.

Features:

- It is a command-line, console based app. A basic GUI is available.
- Can be networked and run over any number of computers.
- It's portable; it should run on Linux, OS X, Windows, and others.
- It can run on small netbooks with ease.
- It supports loadable modules, and has modules to support console/s, moving maps, joysticks, antenna trackers, etc
- Tab-completion of commands.

DroneKit offers an SDK and web API to easily develop apps for your drones. DroneKit is available for android, python and javascript. Python is the most common choice for a Linux, Windows or Mac software development. DroneKit runs over MAVProxy.

MAVProxy, once is started, needs to load the module that allows the binding between MAVProxy and DroneKit. This can be done by using the command:

"`module load droneapi.module.api`"

Code write in python can be run by using the command:

"`api start code.py`"

Some important imports and functions need to be called in the code:

```
from droneapi.lib import VehicleMode
from pymavlink import mavutil


# First get an instance of the API endpoint
api = local_connect()
# get our vehicle - when running with mavproxy it only knows about one vehicle (for now)
v = api.get_vehicles()[0]
```

The API allows read information such as flight mode, altitude, location or speed. The API also allow to move the drone, two main methods are available:

- `goto()` : the input is a GPS location. If the drone is flying, it will go directly to the GPS position defined.
- `channel_override()`: The input is the pair of channel an value that are desired to change. This function simulates the behaviour of a common RC controller.

In this project, as the drone will have to move relatively to the target, the function `channel_override()` is the one used.

In order to test the code, a simulation environment is used. ArduPilot provides a Software-In-The-Loop (SITL) environment, which simulates a copter or plane, in Linux. The SITL replicates the behavior of a real quadcopter with software.

A more detailed information about DroneKit, Mavproxy and SITL are available in the official documentation.

3.2. System design

A ground target chasing system requires several components: the vehicle itself, a camera, an image processing unit and a communication system that intercommunicates all the components. In this project, the vehicle used is the IRIS by 3DR, the camera is a GoPro Hero 3+, the image processing unit is a laptop running OpenCV and the communication system is based two sets of receivers/transmitters, one with a frequency of 5.8 GHz for the transmission of the video signal and other with the frequency of 915 MHz for the transmission of the radio signals with Mavlink protocol.

Some additional components are required. As the video signal is an analog composite signal it need to be digitalized. For this purpose an AV/USB adapter is used. Besides the communication between the laptop and the vehicle, a RC controller is also used for safety purpose. In case that the communication between the vehicle and the laptop fails, the controller can be used. An additional screen with an internal radio signal receiver, the Live View kit by 3DR, is included to visualize in real time the video feed.

Figure 6 Ground target chasing system description

The main objective of this project is make the drone able to follow moving targets in the ground. Some assumptions are made: first of all the system will work in *Loiter* flight mode so the drone itself will control the altitude and will stabilize the possible perturbations caused by the wind. Also it is assumed that the ground target doesn't change its altitude and it moves slower than the drone.

Two modes of ground target are designed: static and dynamic. The first one chase the target without changing the position established, just by rotating and tilting the camera. The dynamic mode chase the target just by changing the pitch and roll maintaining the yaw and the camera's tilt constant.

The system is compound by two different parts: a tracker and an image-based visual servoing system (IBVS). The tracker has the main function of detect the selected target on the video signal and provide the position (X,Y) to the IBVS. The IBVS has the function of transform the information provided by the tracker to a signal ready to be send to the drone in order to move it appropriately.

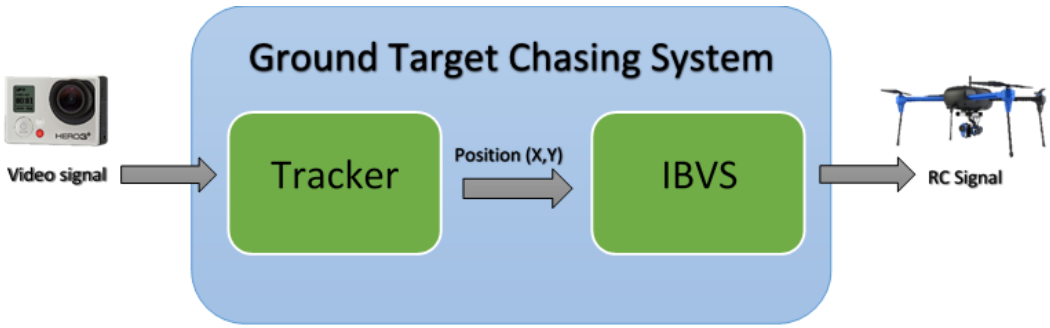


Figure 7 Block diagram of the ground target chasing system

3.3. Tracker development

Tracking is the process of detecting and following a target's position over the time in a video or set of images.

Object tracking is a field widely studied in computer vision, even in the case of aerial images. Tracking objects using aerial images presents some challenges respect common object tracking like the little size of the target to be tracked and/or the shakiness of the camera. To overcome this set of problems, many works have been proposed [1],[2].Some of them are focused in tracking several targets at the same time [9],[11].

Depending on the altitude of the camera different approaches are selected. For low altitude images, the methods for common cameras at ground level are usually used [1],[2],[3],[4]. For a high altitude, different approaches are presented, most of them based in video stabilization [8],[9],[10],[11].

Our tracker is designed in order to work in the range of altitudes that the drone will fly, between 3 to 10 meters approximately, and with the quality of the GoPro's video signal. As only one target will be followed and the altitude will not be very high, a simple tracking method can be used. One important aspect is that our tracker should work in real time as its information provided (the position of the target on the image) is the main element to follow the target.

All the trackers implemented follow the same structure: the user selects a bounding box on the screen and the tracker tries to follow the object inside the box. The output of the tracker is the position of the center of the window or tracking box that contains the object and the box itself.

Different approaches using OpenCV capabilities were tested:

1. Color Based
2. Feature Based
3. Motion Based
4. Tracker Combination:
    a. Tracker framework
    b. Mean-shift based

3.3.1. Color Based

The first method implemented is based on tracking a target using the color information. This method has two main components: the mean-shift / cam-shift algorithm and the back projection.

- Mean-shift / Cam-shift:

Mean shift is a non-parametric feature-space analysis technique for locating the maxima of a density function. This iterative method is started fixing a window containing some samples of the data. The mean of the samples inside the window is computed. Next, the center of the window is shifted to the mean by an iterative process until it converges. A kernel defines the characteristics of the window and the distance measure. This approach is similar to Kmeans or EM algorithm.

The cam-shift algorithm is a more sophisticated variation of the mean-shift. The main problem of the mean-shift algorithm is that the window (so as the kernel) always has the same size and orientation, fact that could lead to a wrong tracking if the target has a changing size. The cam-shift presents a solution to this problem: once the mean-shift algorithm has converged, a new step is added: the size of the window is updated and rotated in order to find the best fitting window to the samples of data. Then the process follows as same as the mean-shift algorithm.

- Back projection:

Back Projection is a way of recording how well the pixels of a given image fit the distribution of a histogram model. The algorithm consist in changing each pixel of the original image to the value of the probability to occur in the image of a sample. Each pixel is compared with the histogram of the sample: if the pixel has a color abundant in the histogram, the probability will be high, so the pixel will be close to white, and if it doesn't, the probability will be small and the pixel will be close to black.

The tracking method proposed starts as follow: First, the image is transformed into the HSV space. Then the user selects a window which includes the object to be tracked. The histogram of hue is computed inside that window and back projection is used in all the following frames.

The gray-scale image resulting of the back projection is used as input of the mean-shift (or cam-shift) algorithm. The output of the system will be the center of the window on each fame.

In order to automatically select an appropriate number of bins of the histogram, the variance of the HSV image is computed. If the image is simple, with a uniform background, a less amount of bins will be used (8 to 16 bins). If the image is more complex, a bigger amount of bins will be used (32 to 64 bins).

The main drawback of this algorithm is that a high saturated colors are required to perform a good tracking. In the case of a low-saturation images, all the colors are close to the grayscale and therefore, the hue component is useless. The selection between mean-shift and cam-shift will depend on the quality and saturation of the image. If the image has low saturation, lots of noise or the target and background have close colors, is recommended to use mean-shift, as cam-shit will adapt wrongly the size and orientation of the window and the tracking process will completely fail.

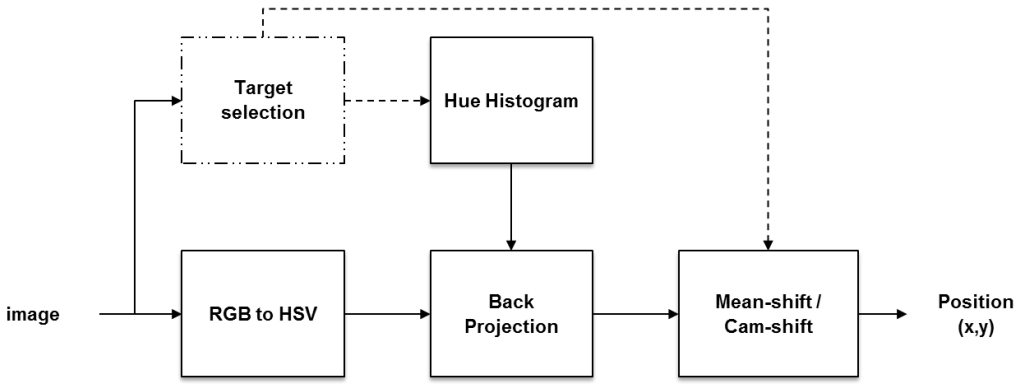


Figure 8 Color-Based tracker block diagram

3.3.2. Feature Based

The next tracker implemented is based on the use of visual features. A feature is defined as an "interesting" part of an image, as it can be corners, edges or other prominent points. Features are usually used as a starting point for many computer vision algorithms. Features has two main components: feature point and feature descriptor.

- The feature point is the position (X,Y) of the pixel on the image where the feature has been detected.

- The feature descriptor is a set of numbers that describes some characteristics (i.e. gradient information) of the image around the correspondent feature point. Usually they have properties such as invariability to translation, rotation and/or scale transformations.

The method presented starts by computing features inside the window selected by the user. In the following frames, a new set of features are computed inside the previous window and 20 pixels around. Features could be computed over all the image, but searching only near of the previous target's position the process is speed up.

This new set of features is matched with the initial one using FLANN (Fast Library for Approximate Nearest Neighbors). FLANN is an algorithm based on kmeans that speeds up the process of matching different set of descriptors, computing the distance between the feature descriptors.

Once the two sets of features are matched, the features without any match are suppressed and the other ones are plotted. The image resulting is composed by a black background and white dots located in the position of the features with a match.

This black and white image is used, as in the previous method, as an input of the mean-shift / cam-shift algorithm. The window will be translated (also rotated and escalated in cam-shift) and a new sets of features will be computed inside (and 20 pixels around) the window. Then the process will be repeated, matching the new set with the initial. If a lots of matches (>15) occur inside the window, the initial sets of features descriptors will be updated by the new matching set of features. Adding this updating step, we avoid losing the target when this change its pose, form or size, but at the same time we risk on losing the object when occlusions occurs or when the background is really crowded.

The election between mean-shift and cam-shift will depend on the size of the object being tracked and on how much crowded is the background. If the object to be tracked is really big, it will have lots of features all over it, and possibly the distribution of this will not be uniform. In that case, the algorithm will start tracking the part of the object that has more features. For example, if you try to track a face that covers almost all the image, this algorithm will end up tracking just some part of the face, an ear for example. That problem doesn't affect in this project because the objects to be tracked will be small as the images used will be aerial images. If the background is really crowded, it will be lots of features, so is interesting to use mean-shift, as cam-shift will adapt wrongly and it will end up following an undesired target.

The library of OpenCV provides an easy way to use several features: SURF, SIFT, ORB, FAST, GFTT… and also an implementation of the FLANN matching algorithm. Different features has been tested and he ones with better results where SIFT, SURF and ORB. As real time is an important aspect of the system, ORB features where the ones used.

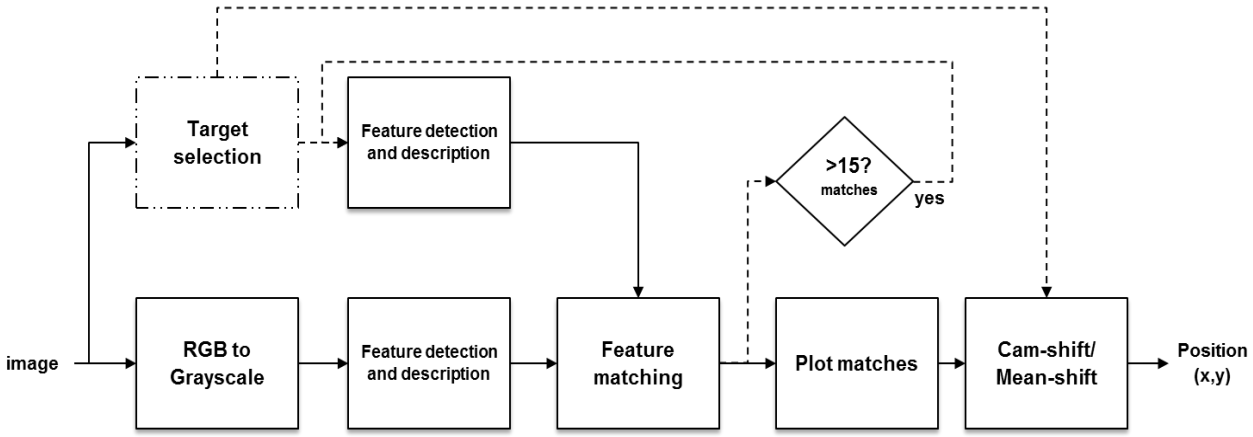


Figure 9 Feature-based tracker block diagram

3.3.3. Motion Based

A third tracker based in the use of motion information is presented. This tracker should not work alone but as a complementary tracker for the two presented before.

The main goal of this tracker is to detect which objects are moving in the scene. In order to accomplish that, visual features ORB are used again. First of all, the images are transformed to a grayscale, then, features are detected and matched between frames. Once the features are matched between consecutive frames, a perspective transformation is estimated and outliers are suppressed using RANSAC algorithm.

Random sample consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of data. The initial assumption is that the data consists of "inliers" (true positive matches), and "outliers" (false negative matches). The RANSAC estimates a matrix H of perspective transformation using the different pairs of matching feature points provided by the FLANN algorithm.

With the matrix H, we apply a perspective transform to the each frame in the video, except the first one. Next, a difference between frames is computed. Appling the perspective transform, allow us to suppress the movement of the camera, so the difference between frames will detect only moving objects. Besides some error appears in the image of the difference between frames, it can be suppress by using a threshold: moving to 0 all the pixels with a value under 30. The difference image resulting, after applying the threshold, is a gray-scale image that can be directly used with mean-shift or cam-shift.

The main drawback of this method is that if the target stays immobile the tracking will completely fail, so a complementary tracker is needed.

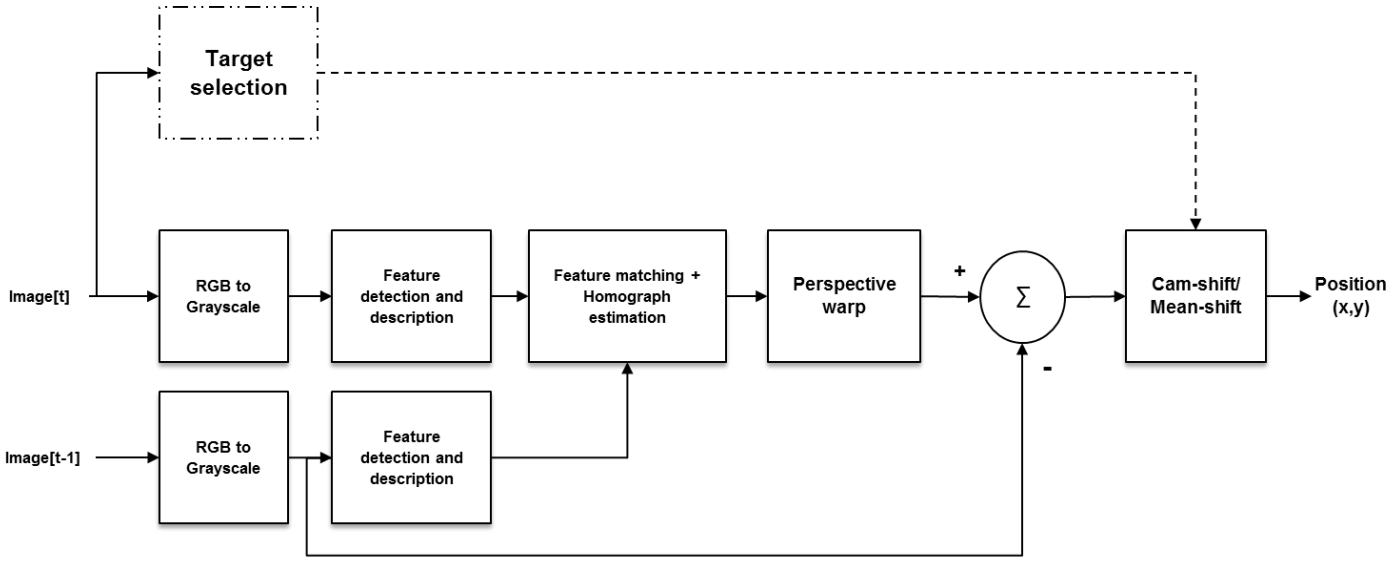


Figure 10 Motion-based tracker block diagram

3.3.4. Tracker combination

In order to combine the three previous trackers exposed, two different methods are presented:

Tracker framework

A framework is used in order to combine as many as trackers wanted. Each tracker runs individually providing a window and a position of the target. Then all the windows and positions are combined in order to get an overall window. The first overall window is the initial one provided by the user. The next overall window are selected as follow:

1. The distance between the previous overall window and the new set of windows are computed.
2. The window with less distance is the new overall window.
3. For all the other windows:
    a. If the distance is twice or more bigger than the minimum distance the tracker is reset using the overall window
    b. If the distance is less than 2 times the minimum distance, the tracker keeps running.

This method allows combine as much as tracker desired. Different combinations of color-based, feature-based and motion-based tracker are used, using both mean-shift and cam-shift. This way of combining the trackers provides good results with image with low noise and average or good quality. But working with low resolution and noisy images, the object is constantly loosed or wrongly tracked. As in our system the signal is noisy due to the transmission via radio, a new system is presented.

Mean-shift based

This method is based on the previous one. The main difference is that instead of combining the trackers in the level of “positions” and “windows” it’s combined before applying mean-shift (also valid with cam-shift).

The gray-scale images provided by color-based, feature-based and motion-based tracker are weighted, combined and used as an input of the mean shift.

After several testing, the combination that better fits our signal is the one with the following weights:

	Weight
Color-Based	2/6
Feature-Based	3/6
Motion-Based	1/6
Mixing method: Mean-shift	

Table 4 Weights of the different trackers

In this system a mask is applied in order to limit the search of the target near the previous position detected. This masks puts in black all the pixels that aren’t contained in a box with a size of 31x31 centered in the previous position detected.

In summary, the tracker used in this project, is a combination of three weighted trackers. One based in the color information (Hue histogram), other based in the use of visual features (ORB) and the last one using motion information (inpair frames difference).

3.4. IBVS development

The Image-Based Visual Servoing system has the main function to compute the information provided by the tracker and transform it into an understandable signal to be sent to the drone in order to move it properly. For this purpose, a simple combination of PID controllers are used.

A proportional-integral-derivative controller (PID controller) is a control loop feedback mechanism widely used in control applications. A PID controller uses an error value as the difference between a measured process variable (PV) and a desired setpoint (SP). In our case, the error is the scaled distance between the center of the frame of the video, the "setpoint", and the position of the object detected in the frame, the process variable. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable, in our case it is the radio signal that is being sent to the drone, directly related with the position of the target in the image detected by the tracker.

The position of the pixel in the object's center provided by the tracker is scaled using the size of the image. This new position, called as the error, has a new range of $\{-0.5,0.5\} \in \mathbb{R}^2$, being (0,0) the center of the image. As this error has 2 dimensions (x,y position at the image), 2 PID will be needed. Each PID will control a different channel of the drone for move the drone in each dimension.

The PID value can be described over time using the following expression:

$$\boldsymbol{PID_x(t) = A_x \cdot (Ap_x \cdot e_x(t) + Ai_x \cdot \int_0^t e_x(\tau)d\tau + Ad_x \cdot \frac{de_x(t)}{dt})}$$

$$\boldsymbol{PID_y(t) = A_y \cdot (Ap_y \cdot e_y(t) + Ai_y \cdot \int_0^t e_y(\tau)d\tau + Ad_y \cdot \frac{de_x(t)}{dt})}$$

$$\overrightarrow{\mathrm{PID}}(\boldsymbol{t}) = (\boldsymbol{PID_x, PID_y})$$

A, Ap, Ai and Ad are constants that will define the behavior of the controller. Each component of the PID controller will have all 4 constants. The function $\vec{e}(t)$ is the value of the position of the target provided by the tracker rescaled using the following expression:

$$\vec{e}(t) = e_{x,y}(t) = (\frac{target_x - center_x}{width}, -\frac{target_y - center_y}{height})$$

Where $\boldsymbol{target_x}$ and $\boldsymbol{target_y}$ are the position of the pixels in the image of the center of the target detected.

Ap is the gain that scales the proportional term, usually called P:

$$\boldsymbol{P(t) = Ap \cdot e(t)}$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable and in contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller.

Ai is the constant that scales the integrative term, usually called I:

$$\boldsymbol{I(t) = Ai \cdot \int_0^t e(\tau) d\tau}$$

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously.

Ad is the constant that scales the derivative term, usually called D:

$$\boldsymbol{D(t) = Ad \cdot \frac{de(t)}{dt}}$$

Derivative action predicts system behavior and thus improves settling time and stability of the system. If the derivative term has a totally random behavior, it is usually suppressed. In our case it is supposed to not be completely chaotic, as usually the target chased will follow an almost straight path with an almost constant velocity.

Finally A is the amplitude that rescales the whole controller. Additionally, each dimension of our PID can be described in function of time using the following expression:

$$\boldsymbol{PID(t) = A \cdot (P(t) + I(t) + D(t))}$$

This PID signal is transformed into an understandable signal for the drone, in our cased called **s:**

$$\mathbf{s}(t) = \tilde{s}(t) + \boldsymbol{PID}(t)$$

The signal $\tilde{s}(t)$ is the last signal that the drone has received, so in the ideal case:

$$\tilde{s}(t) = s(t-1)$$

$$s(t) = s(t-1) + \boldsymbol{PID}(t)$$

Sometimes can happen that the ground station sends a signal but, due to transmission problems or others delays, the signal doesn't arrive to the drone.

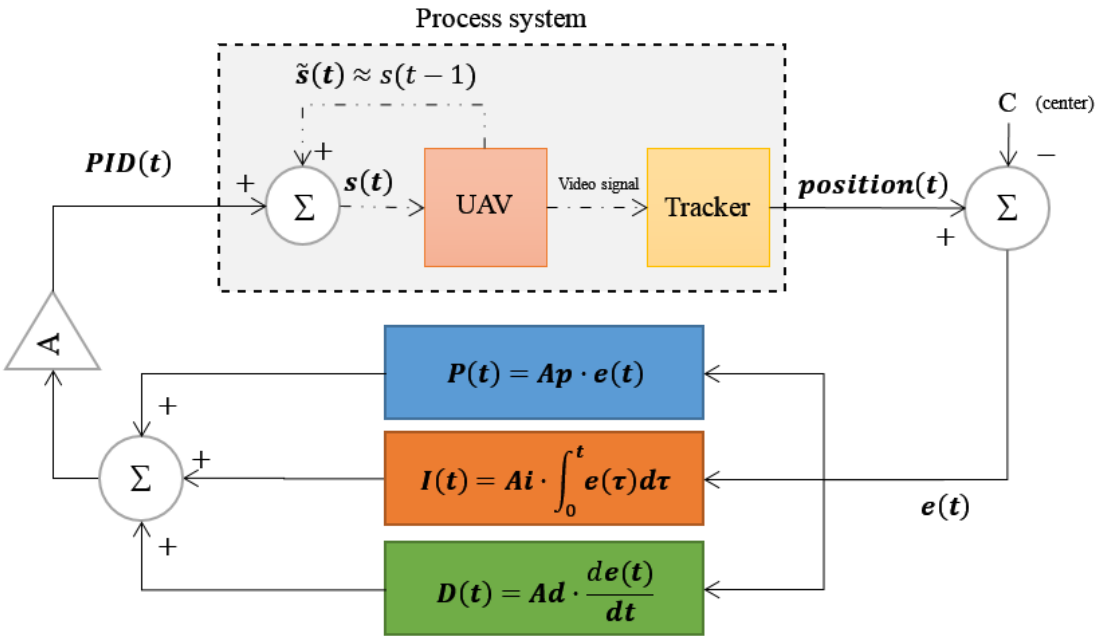


Figure 11 PID controller block diagram

The complete system can be seen as a PID controller that controls a process in order to keep its output, the process variable (PV), as much close as possible to a setpoint (SP), in other words, to reduce the error. In this system, the process variable is the position of the pixel correspondent of the center of the object being tracked and the setpoint is the center of the frame. The process system, or usually called plant in control theory literature, is the set of the UAV, tracker and ground station communicating all together. This process

transforms the signal provided by the PID controller into a new position (SP) by previously sending a signal ($\boldsymbol{s(t)}$) to the UAV. The UAV is moved when it receives the signal, and therefore, the point of view of the camera changes. This produces a change of the position of the target detected by the tracker and, consequently, a different error is sent to the PID controller.

PID controllers are mainly useful when the process is unknown. In our system, lots of parameters of the process are unknown. The relationship between the radio signal $\boldsymbol{s(t)}$ and the position and behavior of the drone in the real world is unknown. Besides this information could be approximated by founding the acceleration and velocity of the device, aspects as wind could lead some inaccuracies. Also, the main problem, is that the real size and distance of the target being followed is unknown. We can only work with the size of the 2D projections of the objects on the images captured with the camera. To overcome this problem, extra information of the real world is needed. Despite this difficulties some works [1] solve this problems by making strong assumptions, as for example, the object being tracked has a fixed and known real size and the initial real distance to it is known.

In order to implement the PID an approximation to discrete time is needed. The proportional term is approximated as the discrete sample of the error:

$$\boldsymbol{P}[\mathrm{n}] = \boldsymbol{Ap} \cdot \boldsymbol{e}[\mathrm{n}]$$

The derivative term is approximated as the difference between two samples:

$$\boldsymbol{D}[\mathrm{n}] = \boldsymbol{Ad} \cdot (\mathrm{e}[\mathrm{n}] - \mathrm{e}[\mathrm{n}-1])$$

For the integrative term, the trapezoidal rule is used:

$$\boldsymbol{I}[\mathrm{n}] = \boldsymbol{Ai} \cdot \sum_{0}^{n} \frac{e[k] + e[k-1]}{2}$$

The trapezoidal rule is based in approximate the integral by a cumulative sum. In the case of trapezoidal rule, the mean between two consecutive samples is used, but a simple cumulative sum of the raw samples is also valid.

The ArduPilot-based drones expects a signal in-between the range of 1000 to 2000. In order to don't exceed the range, the signal is clipped by setting to 0 the $\boldsymbol{PID(t)}$ value until $\mathbf{s}(\boldsymbol{t})$ gets over 1000 or under 2000. Other aspect to take into account is that the value of $\boldsymbol{PID(t)}$ should be in the range of (-1000,1000), so the constants A, Ap, Ai and Ad should be defined properly. The range of $\boldsymbol{e(t)}$ is (-1,1), the range of $\frac{\boldsymbol{de(t)}}{\boldsymbol{dt}}$ is (-2,2) and the range of $\int_0^t \boldsymbol{e(\tau)d\tau}$ is $\{-\infty, \infty\} \in \mathbb{R}$. There are several well-known methods to tune this constants in order to better fit the controller to a specific problem, as it can be the Ziegler-Nichols method, the Tyreus Luyben method or the Choen-coon method. For safety reasons and simplicity, the method chose is a simple manual tuning, as the other ones would require moving the drone. In order to easily tune it, a simple GUI is created and the user is able to change the parameters A, Ap, Ai and Ad in a different range of values.

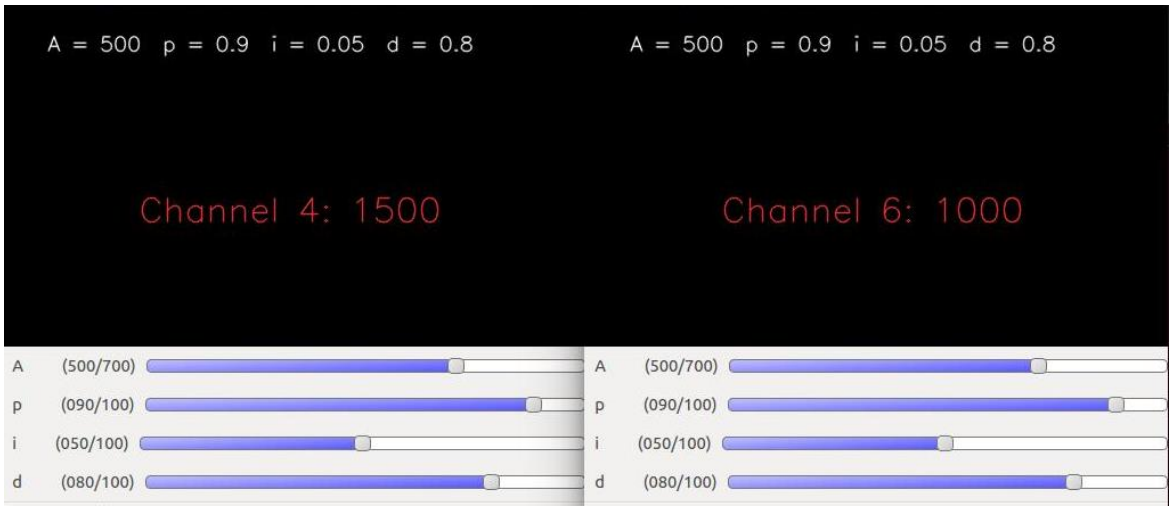


Figure 12 GUI for tuning the PID values

As is presented in other works [1],[2], the integrative term is usually suppressed or with a very low gain. The proportional term and derivative term are the main parts of the controller, and usually the proportional gain is slightly bigger than the derivative gain. In the system proposed, the parameters have the following ranges:

Parameter	Min	Initial ch 6	Initial ch 4	Initial ch 1	Initial ch 2	Max
A	0	500	100	70	70	700
Ap	0	0.9	0.9	0.9	0.9	1
Ai	0	0.05	0.05	0.05	0.05	0.1
Ad	0	0.8	0.8	0.8	0.8	1

Table 5 Initial, minimum and maximum value of the different parameters of the PID

Depending in which mode (static or dynamic) the system is working, the signal $\mathbf{s}(t)$ will be send into different channels. In static mode, the component x of the signal, $s_x(t)$, will be the responsible to change the tilt of the gimbal of the camera, so it will be send through channel 6. The component y of the signal, $s_y(t)$, will be the responsible to change the yaw of the drone, so it will be send thought channel 4. In dynamic mode, $s_x(t)$ will change the pitch of the drone thus it will be send in channel 1. The $s_y(t)$ will change the roll of the drone thus it will be send on channel 2.

$$\vec{s}(t) = (s_x, s_y)$$

As it is presented in the previous table, the initial values of the channels used in dynamic mode, 1 and 2, have a small value of *A* as it is related with the amount of acceleration of the drone. If the constant A is too much big it would lead to an oscillating movement, losing of the object being tracked, or even into a crash.

Other transformation of the signal PID(t) to s(t) is presented:

$$\mathbf{s}(t) = \boldsymbol{PID}(t) + 1500$$

In this case, PID(t) should be in the range of (-500,500). This method should work better in case that the relationship between the signals sent and the position of the drone was lineal. But this is not our case, just the tilt of the gimbal has a lineal relationship with the signal sent. Also that method is affected by the problem that sending a signal doesn't mean that the drone actually receives it. Some problems may occur using this method, for example, if the PID signal starts growing (or decreasing) because the e(t) is bigger (or lower) than 0, but the drone doesn't receive any signal and the PID signal will constantly grow (or decrease). This scenario could lead to some abrupt movements when the drone finally receives that signal. So the first approach of $\mathbf{s}(t)$ is the one choosed.

When the drone receives a signal with value 0, it breaks the connection with the laptop (ground station) and it can be controlled again with the RC controller. For safety reasons, when the target is loosed in the tracker or the user clicks the screen, a signal with value 0 is sent.

4. Experimental testing and Results

The whole system wasn't tested together due climatic conditions and schedule issues. However, all the parts of the system were tested individually.

4.1. Tracker testing

The tracker that have the best performance over all the trackers previously presented is the combination of trackers based on mean-shift. The following results show the performance of this tracker and the behavior of three sub-trackers of which is formed.

The tracker works at a rate of 13 fps which is fast enough for a good ground target chasing. Almost all the computational time is spend in the trackers of motion-based and feature-based in the step of detecting and computing features and in the perspective transformation matrix estimation.

The video used for the testing was recorded in the MSEE building at Purdue University with a height of about 5 meters. A target wearing blue is the one being tracked. The target appears and disappears of the scene and moves with an almost constant velocity.

Figure 13 Target appearance and selection

In the figure 13, the target in blue is appearing at the scene (left image), on the center of the frame, and the target is being selected by the user (right image). A white square is drawn in order to indicate the user which target is selecting.

Figure 14 Moving target being tracked

In the figure 14 the target is being tracked. A square is displayed over the target. To make the interface more intuitive, a line joins the square and the center of the frame. Depending on the distance between the target and the frame's center, the square and the line will be redder or greener. If the target is far from the center, the vehicle will move fast, so a red color is used.

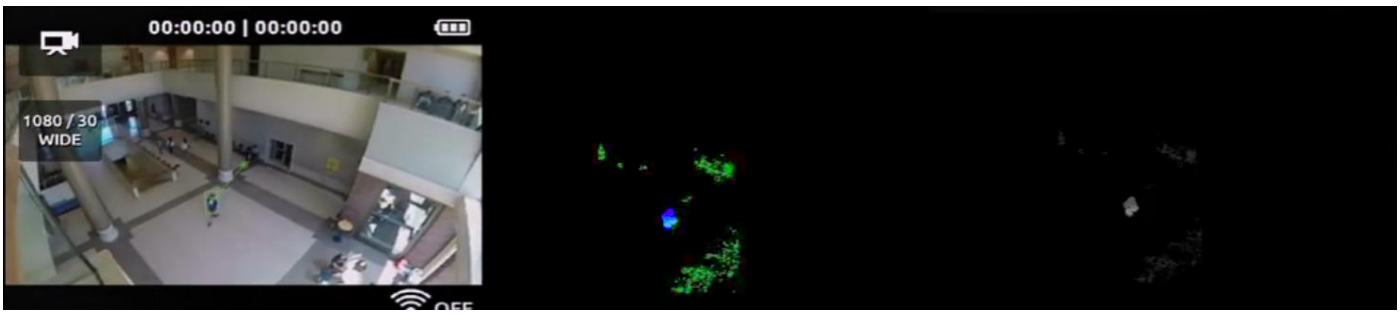


Figure 15 Combination of the three trackers

The combination of the three trackers is showed in the figure 15 and 16. In the left the target being tracked is shown. At the image in the center, each sub-tracker is represented individual by a different color. The blue dots are the feature points, the green pixels represents the output of the back projection on the color-based tracker, and the red dots, are the output of the motion-based tracker. Finally, at the image on the right, a weighted sum of the three outputs is applied. This grayscale image is the input of the mean-shift algorithm that returns the position of the box displayed in the image of the left.

Figure 16 Combination of the three trackers

Finally, when the target disappears of the scene, the tracker fails and a selection of a new target is required (figure 17).

Figure 17 Target is lost when disappearing of the scene

The tracker has been tested with another sequence recorded outside the building. In this case the same target as the previous example is followed.

Figure 18 Target appearance and selection

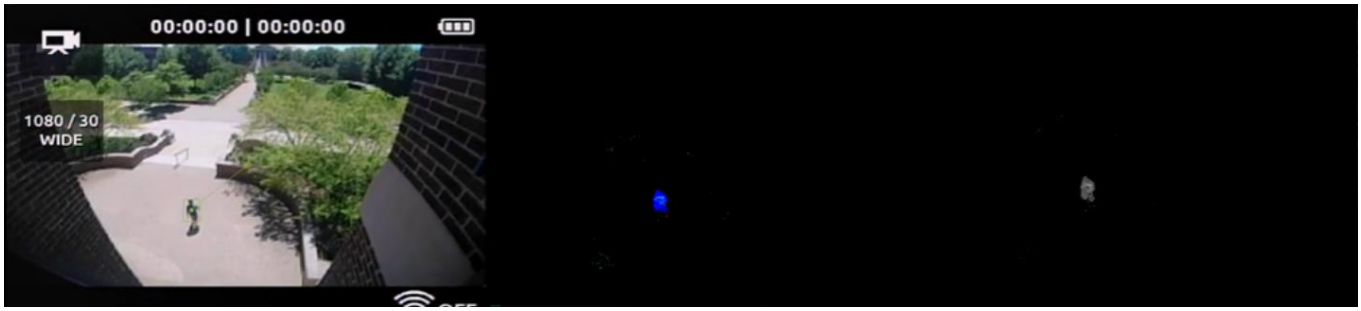


Figure 19 Combination of the three trackers

As in the previous example, the behavior of each individual tracker is presented. In this case (figure 19 and 20) the color-based tracker performs better as the color of the target differs more from the background and is more saturated that in the previous example.

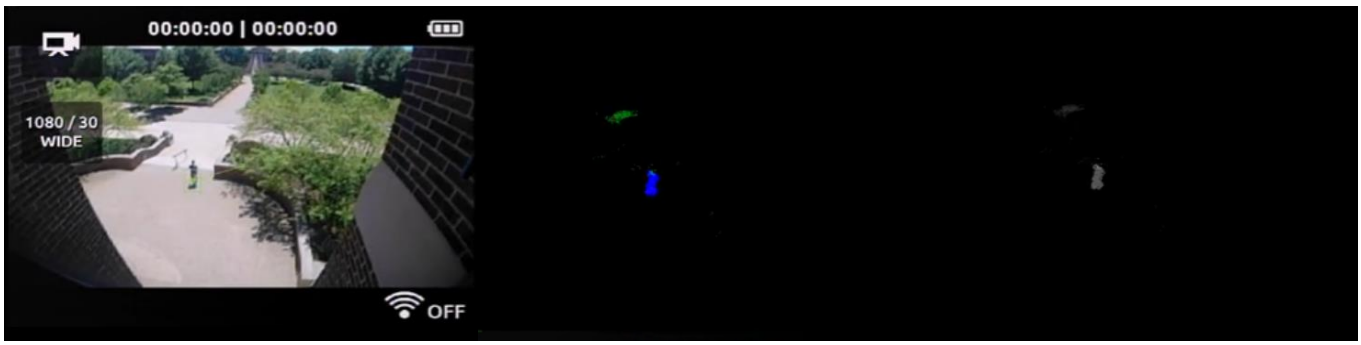


Figure 20 Combination of the three trackers

After the testing, the tracker seems to perform well but is not much resistant to occlusions or scenes with low saturated colors. In some occasions, while tracking an object surrounded by other moving objects, the tracker fails and starts tracking the wrong target. In the case of the disappearance of the target on the scene, the recovering of the tracking when the target reappears is not taken into account, so a new selection of the target is required by the user.

4.2. IBVS testing

Due to the impossibility of flying the vehicle, the testing of the IBVS was made by only tilting the gimbal of the camera and tracking an object in one dimension. In this case only the channel 6, the one responsible of the movement of the camera, is modified.

The set of the following images presents how the camera follows the target, in this case a shoe, when it is moving.

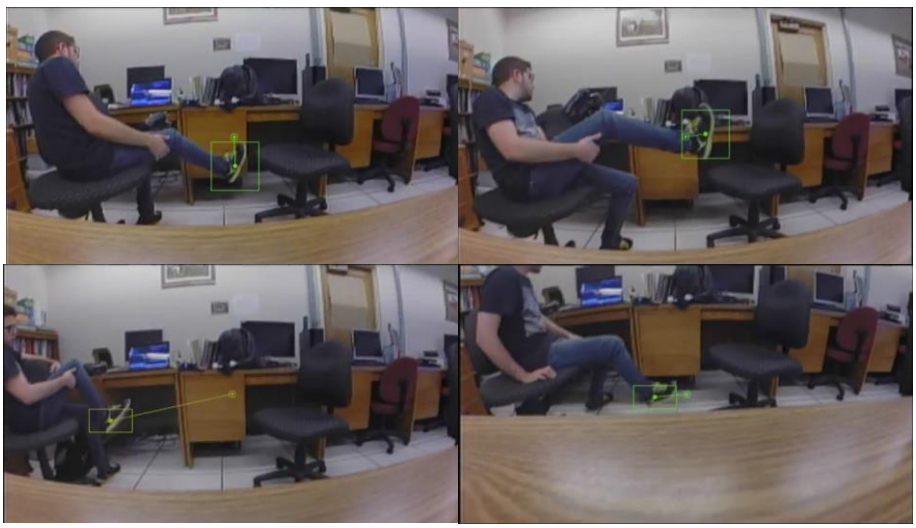


Figure 21 Camera moving while following the target

In the figure 21 can be seen how the camera tilts while following the target (the shoe). The tilt of the camera is limited by looking straight (0º respect X-axis) when receiving a Mavlink signal of a value 2000 and looking down (-90º respect X-axis) when receiving a Mavlink signal of a value of 1000.

Different sets of constants of the PID controller (A, Ap, Ai, Ad) has been tested. Experimentally, the ones that gives a more fluid behavior are 500, 1, 0.05, 0.8 for A, Ap, Ai and Ad respectively. For a higher values the system tends to oscillate and for lower values the system moves too slow.

5. Budget

Depending on the thesis scope this document should include:

- Components list with approximate costs (prototype)
- Design and prototyping costs separate by main tasks (hours person x cost)
- Economic and/or financial viability analysis

If the object of the thesis is not a prototype at least you should include in this section an estimation of the number of hours you have dedicated to the thesis, evaluated at cost of junior engineer.
If you have used a specific software you should also include the license and amortization costs.

Prototype's components, hardware and software list

Product	Price ($)	Shelf life (years)	Period used (months)	Depreciation value ($)
IRIS	750	4	5	78.12
Tarot Gimbal	210	4	5	14
GoPro Hero 3+ Silver Edition	300	5	5	25
Live View Kit	350	4	3	21.8
5.8GHz wireless receiver	17	4	3	1
AV to USB Diamond Adapter	33	5	3	1.65
Laptop	300	5	5	25
Windows 8.1	120	5	5	10
Ubuntu 14.02	0	-	-	0
Total	1660			176.57

Table 6 Components, hardware and software budget

Design cost

Role	Estimated Hours	Price/Hour ($/hour)	Total ($)
Junior Engineer	700	8	5600

Table 7 Human resources design cost

Concept	Price ($)
Human resources	5600
Prototype components, Hardware and software	176.57
Estimated electricity (0,1 kW over 600 hours at 0.125 $)	7.5
Estimated internet (10 $ per month over 6 month)	60
Total	5844.07

Table 8 Design cost

Economic Viability

The project developed is oriented to research and it is not designed to be a final commercial product. This project could be for interest to companies working with drones, or any other unmanned vehicle based on ArduPilot, and computer vision.

The 96.15% of the overall cost of the project comes from the salary of the engineer due the big amortization of the hardware and software. The price of the project is viable and realistic as it fits perfectly to an engineering project of this size and characteristics.

One main point to highlight is that the shelf life of an unmanned vehicle, as a commercial quadcopter, may vary widely depending on factors such as the level of expertise of the pilot, the geography of the flight zone and the weather. As this project is designed to work in a zone free of obstacles and absence of wind or rain, a shelf life of 4 years is a good approximation.

6. Conclusions and future development:

The ground target chasing system is based on two main parts: a tracker and an IBVS controller. The tracker has the function to detect the target on the image over the time. The tracker is usually developed using computer vision techniques. The IBVS controller has the function of move the vehicle using the information provided by the tracker. A good IBVS controller should be able to move the vehicle in a natural and smooth way using usually control theory techniques.

Besides the system developed performs well, at least the parts tested, some aspects could be improved. The tracker is not resistant enough to occlusions or to the disappearance and reappearance of the targets. Besides is resistant enough to the noise produced by the radio transmission of the video, a human supervision is required if the color of the video has low saturation or the scene is really crowded. This aspects could be improved by using a more complex tracker. Some techniques such as Kalmar filter could be used for future improvements.

The IBVS could be improved using the information of the vehicle for tuning all the parameters of the PID controllers. Other type of controllers besides PID, such as H-infinite could be considered. A good tuning would provide a smooth and natural movements to the vehicle. Finally, using information of the IMU, as it is used in previous works [1],[2],[3],[4], for both tracker and IBVS controller could provide a considerable improvement.

As a conclusion, the system developed is oriented to research but it could be a start point for a more robust and reliable product for a commercial use and eventually become a reliable alternative to the use of GPS-based ground target chasing.

Bibliography:

[1] Xuqiang Zhao; Qing Fei; Qingbo Geng, "Vision based ground target tracking for rotor UAV," Control and Automation (ICCA), 2013 10th IEEE International Conference on , vol., no., pp.1907,1911, 12-14 June 2013 doi: 10.1109/ICCA.2013.6565085

[2] Teuliere, C.; Eck, L.; Marchand, E., "Chasing a moving target from a flying UAV," Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on , vol., no., pp.4929,4934, 25-30 Sept. 2011 doi: 10.1109/IROS.2011.6094404

[3] Pestana, J.; Sanchez-Lopez, J.L.; Saripalli, S.; Campoy, P., "Computer vision based general object following for GPS-denied multirotor unmanned vehicles," American Control Conference (ACC), 2014 , vol., no., pp.1886,1891, 4-6 June 2014. doi: 10.1109/ACC.2014.6858831

[4] Mondragón, I.F.; Campoy, P.; Olivares-Mendez, M.A.; Martinez, C., "3D object following based on visual information for Unmanned Aerial Vehicles," Robotics Symposium, 2011 IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC) , vol., no., pp.1,7, 1-4 Oct. 2011. doi: 10.1109/LARC.2011.6086794

[5] Pestana, J.; Sanchez-Lopez, J.L.; Saripalli, S.; Campoy, P., "Vision based GPS-denied Object Tracking and following for unmanned aerial vehicles" Conference: Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on DOI: 10.1109/SSRR.2013.6719359

[6] Justin Griggs B.S.E.E , "Intelligent Data Acquisition and Processing for Unmanned Aerial Vehicles". M.S: Thesis, Graduate Faculty of Texas Tech University, August 2012

[7] J. Hartmann, B. Brown, S. Sri and E. Kivelevitch , "Target Detection using Image Processing Techniques" University of Cincinnati, Cincinnati, Ohio, 45221 5-9 January 2015, Kissimmee, Florida. AIAA 2015-2030. DOI: 10.2514/6.2015-2030

[8] Qingji Gao; Zheng Chao Zeng; Dandan Hu, "Long-term tracking method on ground moving target of UAV," Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese , vol., no., pp.2429,2432, 8-10 Aug. 2014. doi: 10.1109/CGNCC.2014.7007550

[9] Perez, A.; Chamoso, P.; Parra, V.; Sanchez, A.J., "Ground vehicle detection through aerial images taken by a UAV," Information Fusion (FUSION), 2014 17th International Conference on , vol., no., pp.1,6, 7-10 July 2014

[10] Rathinam, S.; Almeida, P.; ZuWhan Kim; Jackson, S.; Tinka, A.; Grossman, W.; Sengupta, Raja, "Autonomous Searching and Tracking of a River using an UAV," American Control Conference, 2007. ACC '07 , vol., no., pp.359,364, 9-13 July 2007. doi: 10.1109/ACC.2007.4282475

[11] A. Qadir, J. Neubert, W. Semke, "On-Board Visual Tracking with Unmanned Aircraft System (UAS)" University of North Dakota, Grand Forks, North Dakota, 58201, American Institute of Aeronautics and Astronautics, May 2012

[12] Chaumette, F.; Hutchinson, S., "Visual servo control. I. Basic approaches," Robotics & Automation Magazine, IEEE , vol.13, no.4, pp.82,90, Dec. 2006. doi: 10.1109/MRA.2006.250573

[13] Hanchen Lu; Feng Zhang; Jiang Wu, "A new real time environment perception method based on visual image for micro UAS flight control," Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese , vol., no., pp.2515,2519, 8-10 Aug. 2014. doi: 10.1109/CGNCC.2014.7007563

[14] Canhoto, A.; Shiguemori, E.H.; Domiciano, M.A.P., "Image sequence processing applied to autonomous aerial navigation," Signal and Image Processing Applications (ICSIPA), 2009 IEEE International Conference on , vol., no., pp.496,499, 18-19 Nov. 2009. doi: 10.1109/ICSIPA.2009.5478706

[15] Cesetti, A.; Frontoni, E.; Mancini, A.; Zingaretti, P., "Autonomous safe landing of a vision guided helicopter," Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on , vol., no., pp.125,130, 15-17 July 2010. doi: 10.1109/MESA.2010.5552081

[16] Schwertfeger, S.; Birk, A.; Bulow, H., "Using iFMI spectral registration for video stabilization and motion detection by an Unmanned Aerial Vehicle (UAV)," Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on , vol., no., pp.61,67, 1-5 Nov. 2011. doi: 10.1109/SSRR.2011.6106770

Appendix I: Hardware specifications

IRIS vehicle

IRIS Features	
Weight with battery	1282 g
Maximum payload	400 g
Flight time	16-22 min
Motor-to-motor dimension	550 mm
RC Controller	Any PPM compatible RC unit
Telemetry	3DR Radio 915 MHz
Camera mount	GoPro tarot gimbal
Battery	5100 mAh
Motors	920 kv
Maximum velocity	22.7 m/s (81 km/h)
Maximum wind	5-6 m/s (18–21 km/h)
Maximum safe altitude	120 m
Safe distance from people or vehicles	30 m

Tarot Gimbal Features	
Weight	200 g
Control accuracy	0.1º
Control angle range roll	-45 ° ~ 45 °
Control angle range tilt	-135 ° ~ 90 °

Pixhawk specifications	
Processor	32-bit ARM Cortex M4 core with FPU
	168 Mhz/256 KB RAM/2 MB Flash
	32-bit failsafe co-processor
Sensors	MPU6000 as main accel and gyro
	ST Micro 16-bit gyroscope
	ST Micro 14-bit accelerometer/magnetometer
	MEAS barometer
Interfaces	5x UART serial ports, 1 high-power capable, 2x with HW flow control
	Spektrum DSM/DSM2/DSM-X Satellite input
	PPM sum signal
	RSSI (PWM or voltage) input
	I2C, SPI, 2x CAN, USB
	3.3 and 6.6 ADC inputs

5.8 GHz Transmitter features	
Weight	60 g (with antenna)
Transmitter Frequency / Channel Amount	5705-5945MHz / 8CH
Output Power	200mW/33dBm
AV Input	Analog AV composite signal input
Power Supply	DC 7-12 volts
Current Supply	850mA
Size	68*26*28mm

GoPro Camera

GoPro HERO3+ Silver Features	
Weight	74 g
Battery life	1:30 – 3:00 h
Ports	Mini USB, Micro HDMI, microSD
Playback	HDMI, Composite (mini usb), Wi-fi
Internal memory	microSD class 10 (or UHS-1) up to 64GB

Video Resolution	Fps	FOV	Screen Resolution
1080p	60, 50, 30, 25	Ultra Wide, Medium, Narrow	1920x1080
960p	60, 50, 30, 25	Ultra Wide, Medium, Narrow	1280x960
720p	120, 100, 60, 50, 30, 25	Ultra Wide	1280x720
WVGA	120, 100, 60, 50	Ultra Wide	848x480

Payload

Total payload allowed = 400 g	
Tarot gimbal	200 g
GoPro	74 g
5.8 GHz Transmitter	60 g
Total payload	334 g

Appendix II: Project Plan

Work Packages:

Project: Ground target chasing with a UAV	WP ref: WP1	
Major constituent: UAV maintenance and adjustment	Sheet 1 of 3	
Short description: This work package includes all the modifications applied to the drone and its respective calibration.	Planned start date:02/03/2015 Planned end date: 10/07/2015	
	Start event: 02/03/2015 End event:	
Internal task T1: UAV maintenance Internal task T2: LiveView kit mounting and testing Internal task T3: Firmware update and calibration Internal task T4: Tarot Gimbal calibration and fixing	Deliverables:	Dates:

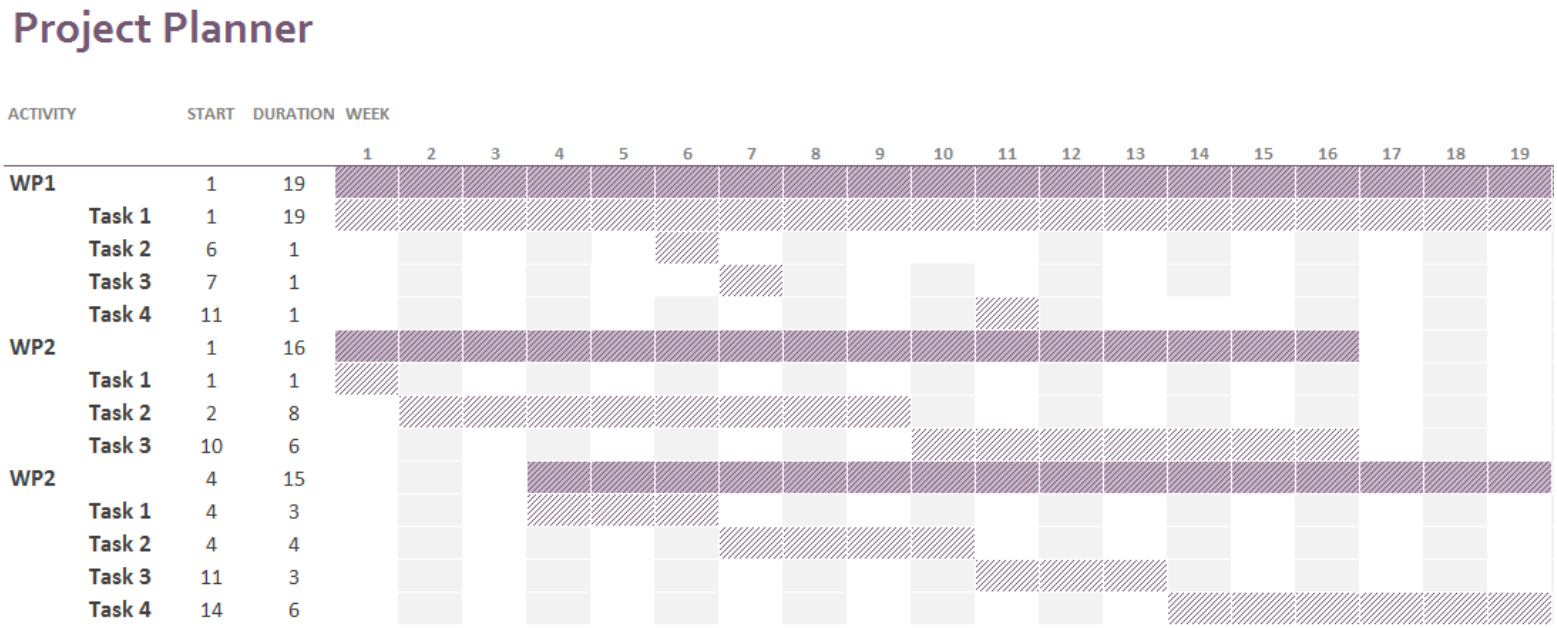
Project: Ground target chasing with a UAV	WP ref: WP2	
Major constituent: Tracking system	Sheet 2 of 3	
Short description: This work package includes the development of a tracking system using Python and OpenCV. Some simple programs are developed in order to get familiar with the usage of Python and OpenCV. Then some algorithms of object tracking are developed and tested in aerial images.	Planned start date: 02/03/2015 Planned end date: 22/06/2015	
	Start event: 02/03/2015 End event:-	
Internal task T1: Python and OpenCV installation Internal task T2: Simple OpenCV programs implementation. Internal task T3: Object tracking system development and testing.	Deliverables:-	Dates:-

Project: Ground target chasing with a UAV	WP ref: WP3	
Major constituent: IBVS Controller	Sheet 3 of 3	
Short description: This package includes the development of an IBVS Controller using the DroneKit API and Python. A PID-based controller will be implemented in order to control the drone and it will be combined with the tracking system. First only the gimbal will be moved in order to track the target then it will be extrapolate to the whole UAV.	Planned start date: 08/03/2015 Planned end date:10/07/2015	
	Start event: 08/03/2015 End event:-	
Internal task T1: Setup of DroneKit and simulation environment. Internal task T2: Simple code implementation using DroneKit. Internal task T3: PID controller for gimbal Internal task T4: PID controller for the whole drone	Deliverables:-	Dates:-

Milestones

WP#	Task#	Short title	Date (week)
1	1	UAV maintenance	1
1	2	LiveView kit mounting and testing	6
1	3	Firmware update and calibration	7
1	4	Tarot Gimbal calibration and fixing	11
2	1	Python and OpenCV installation	1
2	2	Simple OpenCV programs implementation.	2
2	3	Object tracking system development and testing.	10
3	1	Setup of DroneKit and simulation environment	4
3	2	Simple code implementation using DroneKit.	4
3	3	PID controller for gimbal development	11
3	4	PID controller for the whole drone development	14

Grant Diagram



Glossary

APM: ArduPilot Mega

EM: Expectation Maximization

FAST: Features from Accelerated Segment Test

FLANN: Fast Library for Approximate Nearest Neighbors

GUI: Graphic User Interface

HSV: Hue Saturation Value

IBVS: Image-Based Visual Servoing

ORB: Oriented FAST and Rotated BRIEF

PID: Proportional Integral Derivative

PV: Process Variable

RANSAC: Random sample consensus

RGB: Red Green Blue

SIFT: Scale-Invariant Feature Transform

SITL: Software In The Loop

SP: Set Point

SURF: Speeded Up Robust Features

UAV: Unmanned Aerial Vehicle