UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MAIK BASSO

A FRAMEWORK FOR AUTONOMOUS MISSION AND GUIDANCE CONTROL OF UNMANNED AERIAL VEHICLES BASED ON COMPUTER VISION TECHNIQUES

Porto Alegre
2018

MAIK BASSO

A FRAMEWORK FOR AUTONOMOUS MISSION AND GUIDANCE CONTROL OF UNMANNED AERIAL VEHICLES BASED ON COMPUTER VISION TECHNIQUES

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Controle e Automação

ORIENTADOR: Prof. Dr. Edison Pignaton de Freitas

Porto Alegre
2018

MAIK BASSO

A FRAMEWORK FOR AUTONOMOUS MISSION AND GUIDANCE CONTROL OF UNMANNED AERIAL VEHICLES BASED ON COMPUTER VISION TECHNIQUES

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Porto Alegre, Março de 2018.

DEDICATÓRIA

Dedico este trabalho aos meus pais Devanir Giovani Basso e Eliane Marisa Cadoná Basso, que apesar de todas as dificuldades estiveram sempre ao meu lado em todas as minhas escolhas, me orientando e me motivando sempre com muito amor, carinho e compreensão. Dedico este trabalho ao meu irmão mais novo, Gabriel Cadoná Basso, ao qual consegue mesmo em tempos de dificuldade ter sempre um sorriso enorme no rosto, me motivando e me alegrando em tempos difíceis.

Dedico este trabalho aos meus avós paternos, Adelino José Basso e Ondina Piaia, e a minha avó materna Delmiria Vanin Cadoná que sempre me deram o apoio necessário para que eu conseguisse enfrentar as dificuldades encontradas durante o curso. Também dedico este trabalho ao meu avô materno, Alceu Cadoná, que apesar de não estar mais entre nós, tenho certeza que deve estar muito feliz com todas as nossas conquistas.

Dedico este trabalho a minha namorada, Jessica Andressa Kloster, por estar sempre ao meu lado, me apoiando e acima de tudo compartilhando das angústias e alegrias que este curso proporcionou nas nossas vidas.

Dedico este trabalho a todos os colegas e amigos que se fizeram presentes neste período da minha vida, sempre me aconselhando e dando apoio para que eu consegui-se superar todos os obstáculos no meu caminho.

Por fim, dedico este trabalho à Deus, por estar sempre ao meu lado me ouvindo e me protegendo nas horas mais difíceis.

A todos um muito obrigado!

AGRADECIMENTOS

Ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, pela oportunidade de realização de pesquisa nas minhas áreas de interesse.

Ao meu professor e orientador, Edison Pignaton de Freitas, agradeço a todo o tempo que dedicaste a me orientar em meu trabalho. Agradeço por me entusiasmar e não desistir do meu trabalho mesmo nos momentos mais difíceis. Agradeço sempre pelo seu apoio, saiba que sou muito grato pelas oportunidades que colocastes na minha vida. Agradeço por nossa amizade. Muito obrigado.

Ao professor Christian Bredemeier, responsável pelas plantações de milho localizadas na fazenda experimental da UFRGS, por permitir que a área fosse utilizada para os testes de campo desenvolvidos e apresentados neste trabalho.

Ao CNPQ pela concessão de bolsa para a minha manutenção durante o período de dedicação exclusiva ao meu trabalho de pesquisa.

A todas as pessoas que se fizeram presentes nessa etapa da minha vida me apoiando para que a conclusão deste curso fosse possível.

"Eu quero, eu sei, eu posso e eu consigo!"

Eliane Marisa Cadoná Basso

RESUMO

A computação visual é uma área do conhecimento que estuda o desenvolvimento de sistemas artificiais capazes de detectar e desenvolver a percepção do meio ambiente através de informações de imagem ou dados multidimensionais. A percepção visual e a manipulação são combinadas em sistemas robóticos através de duas etapas "olhar"e depois "movimentar-se", gerando um laço de controle de feedback visual. Neste contexto, existe um interesse crescimente no uso dessas técnicas em veículos aéreos não tripulados (VANTs), também conhecidos como drones. Essas técnicas são aplicadas para posicionar o drone em modo de vôo autônomo, ou para realizar a detecção de regiões para vigilância aérea ou pontos de interesse. Os sistemas de computação visual geralmente tomam três passos em sua operação, que são: aquisição de dados em forma numérica, processamento de dados e análise de dados. A etapa de aquisição de dados é geralmente realizada por câmeras e sensores de proximidade. Após a aquisição de dados, o computador embarcado realiza o processamento de dados executando algoritmos com técnicas de medição (variáveis, índice e coeficientes), detecção (padrões, objetos ou áreas) ou monitoramento (pessoas, veículos ou animais). Os dados processados são analisados e convertidos em comandos de decisão para o controle para o sistema robótico autônomo. Visando realizar a integração dos sistemas de computação visual com as diferentes plataformas de VANTs, este trabalho propõe o desenvolvimento de um framework para controle de missão e guiamento de VANTs baseado em visão computacional. O framework é responsável por gerenciar, codificar, decodificar e interpretar comandos trocados entre as controladoras de voo e os algoritmos de computação visual. Como estudo de caso, foram desenvolvidos dois algoritmos destinados à aplicação em agricultura de precisão. O primeiro algoritmo realiza o cálculo de um coeficiente de reflectância visando a aplicação auto-regulada e eficiente de agroquímicos, e o segundo realiza a identificação das linhas de plantas para realizar o guiamento dos VANTs sobre a plantação. O desempenho do framework e dos algoritmos propostos foi avaliado e comparado com o estado da arte, obtendo resultados satisfatórios na implementação no hardware embarcado.

Palavras-chave: Computação Visual, Processamento de imagens, Hardware Embarcado, Software Embarcado, Controle Autonomo de missão, VANT, Agricultura de Precisão.

ABSTRACT

Cumputer Vision is an area of knowledge that studies the development of artificial systems capable of detecting and developing the perception of the environment through image information or multidimensional data. Nowadays, vision systems are widely integrated into robotic systems. Visual perception and manipulation are combined in two steps "look" and then "move", generating a visual feedback control loop. In this context, there is a growing interest in using computer vision techniques in unmanned aerial vehicles (UAVs), also known as drones. These techniques are applied to position the drone in autonomous flight mode, or to perform the detection of regions for aerial surveillance or points of interest. Computer vision systems generally take three steps to the operation, which are: data acquisition in numerical form, data processing and data analysis. The data acquisition step is usually performed by cameras or proximity sensors. After data acquisition, the embedded computer performs data processing by performing algorithms with measurement techniques (variables, index and coefficients), detection (patterns, objects or area) or monitoring (people, vehicles or animals). The resulting processed data is analyzed and then converted into decision commands that serve as control inputs for the autonomous robotic system. In order to integrate the visual computing systems with the different UAVs platforms, this work proposes the development of a framework for mission control and guidance of UAVs based on computer vision. The framework is responsible for managing, encoding, decoding, and interpreting commands exchanged between flight controllers and visual computing algorithms. As a case study, two algorithms were developed to provide autonomy to UAVs intended for application in precision agriculture. The first algorithm performs the calculation of a reflectance coefficient used to perform the punctual, self-regulated and efficient application of agrochemicals. The second algorithm performs the identification of crop lines to perform the guidance of the UAVs on the plantation. The performance of the proposed framework and proposed algorithms was evaluated and compared with the state of the art, obtaining satisfactory results in the implementation of embedded hardware.

Keywords: Computer Vision, Image Processing, Embedded Hardware, Embedded Software, Autonomous Mission Control, UAV, Precision Agriculture.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

COTS	Commercial Off-The-Shelf
CRD	Crop Row Detection
CPU	Central Processing Unit
CSV	Comma-Separated Values
DC	Direct Current
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GNDVI	Green Normalized Difference Vegetation Index
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HD	High Definition
JSON	JavaScript Object Notation
MAVLink	Micro Air Vehicle Link
MB	Megabyte
NDVI	Normalized Difference Vegetation Index
NIR	Near Infrared
NoIR	No Infrared
OpenCV	Open Source Computer Vision Library
OpenCL	Open Computing Language
PC	Personal Computer
RAM	Random Access Memory
RHT	Random Hough Transform
RGB	Red, Green and Blue
RGB-D	Red, Green, Blue and Depth

RTL	Return To Launch
RTC	Real Time Clock
SLAM	Simultaneous Localization And Mapping
TCP	Transmission Control Protocol
TPF	Time Per Frame
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UFRGS	Federal University of Rio Grande do Sul
UGV	Unmanned Ground Vehicle
USB	Universal Serial Bus
VANT	Veículo Aéreo Não Tripulados

SUMÁRIO

1 RESUMO EXTENDIDO

Este capítulo apresenta de forma resumida, o presente trabalho, o qual é intitulado "Um framework para controle de missão e guiamento autônomo de veículos aéreos não tripulados, baseado em técnicas de visão computacional".

1.1 Introdução

Estudos recentes estão focados no desenvolvimento de sistemas baseados em técnicas de visão computacional para dar autonomia a veículos aéreos não tripulados (VANTs), sendo que, esses sistemas abrangem diversas áreas do conhecimento. Devido a avanços recentes nessas pesquisas, foi possível estender o escopo desses sistemas para a área de agricultura de precisão, pois nesta área, diversas aplicações são desenvolvidas com o uso de tecnologias utilizando VANTs, como mapeamento, monitoramento e mais recente, aplicações envolvendo pulverização de agroquímicos sobre as plantações.

Neste contexto, este trabalho apresenta um estudo de caso em agricultura de precisão, onde que foram desenvolvidos algoritmos utilizados para dar suporte e trazer autonomia a VANTs, que fazem a aplicação de agroquímicos. Para tanto, dois algoritmos foram propostos: o primeiro algoritmo, realiza o cálculo de um coeficiente de reflectância utilizado para realizar a aplicação pontual, autorregulada e eficiente de agroquímicos; o segundo algoritmo, realiza a identificação das linhas de plantas do cultivo, para realizar o guiamento dos VANTs sobre a plantação.

Diante o exposto, o presente trabalho também propõe o desenvolvimento de um framework com arquitetura generalista, projetada para suportar e atender aos requisitos de diferentes aplicações, tais como, aplicações militares, inspeções e monitoramento que exigem a integração de sistemas embarcados em VANTs, com aplicações de visão computacional. O framework proposto foi utilizado para integrar os algoritmos propostos nos estudos de caso em agricultura de precisão, com o hardware integrado ao VANT usado nos testes de campo.

O framework foi projetado para fornecer ferramentas para o controle autônomo de missão e orientação de VANTs através da troca de mensagens entre o hardware embarcado e os algoritmos de visão computacional e execução de comandos. O framework é altamente escalável e permite a integração com vários tipos de topologias de missão, além de que, sua arquitetura permite a integração entre um ou mais VANTs e um ou mais algoritmos de visão.

1.2 Experimentos e Resultados

Primeiramente, foram realizados experimentos para comprovar a viabilidade de utilização do framework proposto, sendo que, em laboratório, foram propostos experimentos para averiguar o desempenho na troca de mensagens, entre os algoritmos propostos e o hardware controlador de voo dos VANTs. Nos testes realizados, o tempo médio para executar um comando foi de $237,82$ ms. Esta medida, compreende desde o momento em que a mensagem está preparada para o envio do lado do algoritmo de visão, até o momento em que é decodificado e executado no formato de comando pelo framework. Considerando que esse tempo envolve dois processos de comunicação e a execução do comando, pode-se constatar que o framework proposto atende as necessidades temporais da aplicação. Após isso, foram realizados testes com simulador, nos quais, os algoritmos propostos foram capazes de executar missões através da conexão entre o framework proposto e o simulador do hardware dos VANTs.

Também, foram realizados testes de laboratório envolvendo os dois algoritmos de visão propostos. O algoritmo, para automação do processo de aplicação de agroquímicos, alcançou desempenho de $1,64$ FPS, para imagens em alta resolução e $45,98$ FPS, para a menor resolução avaliada. Já o algoritmo proposto para a condução autônoma dos VANTs, sobre as linhas da plantação, obteve resultado de $93,62$ FPS, para a resolução mais baixa e $1,63$ FPS, para resolução de alta definição. Ambos os algoritmos, passaram por um longo processo de ajustes, para que sua eficiência fosse mantida, mesmo nas resoluções de imagem de tamanho inferior. Processo este que possibilitou aos algoritmos atender os requisitos temporais dos estudos de caso propostos.

A eficácia do framework e dos algoritmos propostos também foi comprovada em testes de campo realizados. Nesses testes, o framework foi utilizado para dar suporte ao funcionamento e testes com os algoritmos dos estudos de caso em ambiente real de uso. Durante os testes com algoritmo de aplicação de agroquímicos, para um conjunto de 120 amostras coletadas durante o experimento realizado, a diferença média entre as medidas do algoritmo proposto, comparado com o equipamento comercial que realiza estas medidas, foi de $0,0055416667$, com um desvio padrão de $\pm 0,0033378296$.

Nos testes de campo realizados com o algoritmo de condução autônoma, o sistema foi capaz em realizar a condução do VANT sobre uma plantação de milho em estágio inicial de crescimento. O algoritmo também foi capaz de utilizar as linhas como objeto de referência para a condução e se mostrou invariante às condições ambientais, como luminosidade e ruído de fundo da imagem. Além disso, o framework se mostrou eficaz, destinando todos os comandos ao hardware, obedecendo o tempo requerido pela aplicação.

1.3 Contribuições

A principal contribuição deste trabalho é a concepção do framework, no entanto, o trabalho também apresenta outras contribuições que são:

- Algoritmo para a aplicação autorregulada de agroquímicos;
- Algoritmo para detecção de linhas de cultura para o guiamento autônomo dos VANTs sobre plantações de agricultura de precisão;
- O desenvolvimento de um filtro proposto para o pré-tratamento das imagens e posterior identificação das linhas de plantações;
- O filtro para remover os resultados falsos positivos detectados pelo algoritmo de detecção de linha;
- O algoritmo responsável por gerar os parâmetros de orientação com base em linhas de colheita previamente detectadas;

1.4 Conclusões

Todos os objetivos planejados para o trabalho foram atingidos e os requisitos definidos para aplicações alvo do estudo de caso foram cumpridos. O framework proposto foi capaz de servir como plataforma para implementação dos algoritmos propostos nos estudos de caso. O desempenho das diversas partes do sistema proposto pôde ser comprovado através de testes realizados em ambiente de laboratório e em testes de campo.

2 INTRODUCTION

Computer vision is an area of knowledge that studies the development of artificial systems capable of detecting and developing the perception of the environment through image information or multidimensional data, obtained by different types of sensors (HUANG, 1996). Computer vision is a useful mode of robotic detection, since it imitates the human visual sense and allows the extraction of non-contact measurements of the environment. Computational vision involves working with higher concepts and algorithms related to artificial intelligence, which involves intense programming so that it fits with the activities and requirements of the system. The scope of computer vision encompasses several areas of knowledge as can be seen in Fig. 1 (PRASAD, 2012). The computational vision differs from image processing because in addition to the image processing, signals obtained from sensors and other means the computer vision aim to analyze, to understand and to provide the interaction of the system with the environment in which it is inserted.

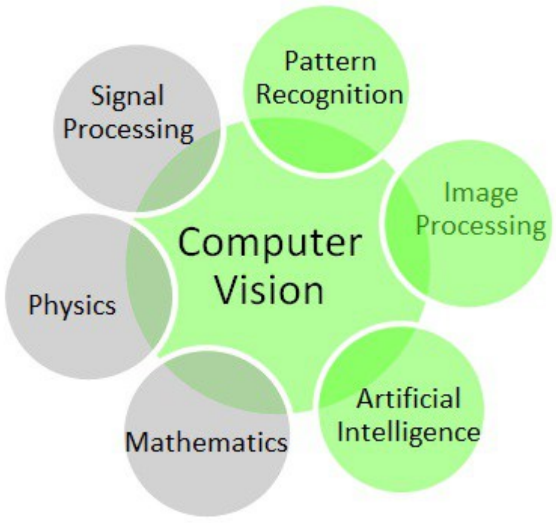


Figura 1: Computer vision scope (PRASAD, 2012).

From the engineering point of view, the computer vision is a valuable tool to build autonomous systems that can accomplish some of the tasks that the human visual system can perform, and, in many cases, overcome the human capabilities (HUANG, 1996). Currently, vision systems are widely integrated with robotic systems. Commonly, visual perception and manipulation are combined in two steps "look"and then "move"(CONTICELLI; ALLOTTA, 2000). The accuracy of the results depends directly on the accuracy of the visual sensor. An alternative to increase the overall system accuracy is to use a visual feedback control loop, decreasing the obtained error.

In this context, there is an increasing interest in using computer vision techniques in Unmanned Aerial Vehicles (UAV), also known as drones. In their case, these techniques are applied to position the UAV in autonomous flight mode, or to perform aerial surveillance and detection of regions of interest or points of interest (ROIs or POIs) (CAMPOY et al., 2008). This trend is driven by different facts, such as the miniaturization of electronic components, including sensors (driven by other technologies such as smartphones); the increase of computational power for the on-board CPUs; and the cost reduction of other components of this type of robotic platform. With today's technology, the variety and complexity of mission tasks modern UAVs are demanded to achieve, they require higher levels of autonomy. The main part of a standalone UAV is the navigation control system and its supporting subsystems. The autonomous navigation system uses information from several subsystems to perform three main tasks: estimating the location and orientation of the UAV (location), identifying obstacles in the environment (obstacle detection), and then making decisions (decision-making). These decisions are critical to maintaining the control loop and to provide navigation in an unknown environment (AL-KAFF et al., 2018). Computer vision is an indispensable tool to achieve these goals.

Computer vision systems installed in and integrated to the architecture of the UAVs generally have similar architecture and take three steps in its operation, which are: data acquisition in numerical form, data processing and data analysis. Cameras, proximity sensors, and ultrasonic sensors typically perform the data acquisition step. After the data acquisition, the embedded computer performs data processing by performing algorithms with measurement techniques (variables, index and coefficients), detection (patterns, objects or ROI/POI) or monitoring (people, vehicles or animals). The resulting processed data is analyzed and then converted into decision commands that serve as input to the autonomous robotic system (TOMIC et al., 2012).

With all this advancement in embedded hardware and the developed techniques, the scope of use of the UAVs has been extended to a number of application areas, such as precision agriculture (PURI; NAYYAR; RAJA, 2017). This usage in precision agriculture is further subdivided into a number of other study and application sub-areas such as wildlife and crop monitoring, mapping of areas and plantations, and also for autonomous spraying of agrochemicals on the lines of cultures (LUO et al., 2017; FAIçAL et al., 2014) as shown in Fig. 2.

The use of agrochemicals in precision agriculture is essential to maintain production quality and scalability. The use of UAVs for spraying avoids soil compaction and also reduces the waste of agrochemicals, compared to massive spraying using conventional manned aircrafts. However, unfavorable cli-

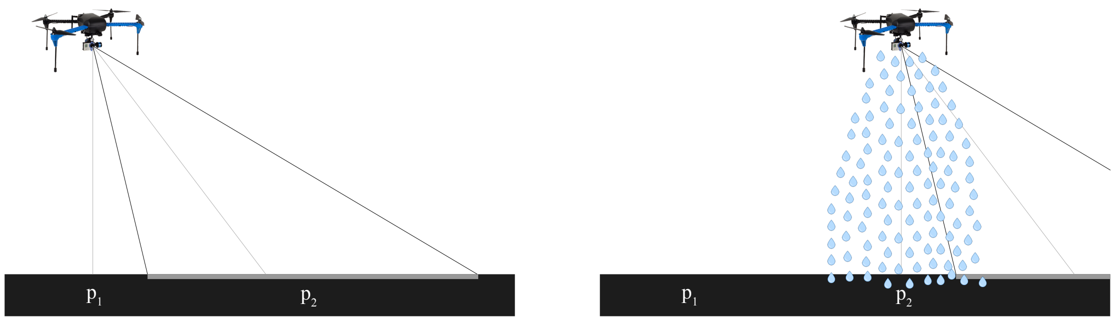
Figura 2: A man controls a drone to spray pesticides (DAWEI, 2015).

matic conditions and error in the global positioning system (GPS) may hamper the smooth functioning of these systems. These errors may cause the UAV to deviate its trajectory and not to cover the entire desired area, or also to invade surrounding crop fields where the application of these agrochemicals is not recommended (FAIÇAL et al., 2014).

For these reasons, this work proposes as a case study the development of two algorithms to provide autonomy to the UAVs intended for application in precision agriculture. These algorithms run in parallel on embedded hardware. The first algorithm performs the calculation of a reflectance coefficient used to perform the punctual, self-regulated and efficient application of agrochemicals. The second algorithm performs the identification of the crop rows to perform the autonomous guidance of the UAVs on the plantation.

However, these proposed algorithms have their performance affected by two reasons. The first refers to the brightness, complexity and abrupt variation of the background of the images. The second refers to the temporal requirements of the system. The model scenario of application of agrochemicals with the use of UAVs can be observed in Fig. 3.

In the proposed spray scenario model the UAV moves at a constant speed between two and three meters per second. The proposed computer vision algorithms use as sersor a camera present in the vehicle. During the mission, the image captured at the instant in which the vehicle is on the point p_1 is realized at the instant $t = 1$. At a later time, $t = 2$, the vehicle has moved to the point p_2. At this point, the vehicle must perform the actions following the commands generated from the previously acquired image and data processing.



(a) Moment $t = 1$. (b) Moment $t = 2$.

Figura 3: Specification of the spray scenario using UAVs.

Considering the velocity of the vehicle, it is evident that the proposed system has few milliseconds to perform all the application flow and yet to provide the necessary response to the actuator.

In addition to the problems with timing requirements, another problem encountered in computer vision applications used in UAVs is that there is no tool or standard to facilitate the integration of these systems with the hardware. In this context, this work proposes the development of a framework with a generalist architecture that is designed to support and meet the requirements of different applications, such as military applications, structure inspections and monitoring that require the integration of embedded systems in UAVs with vision applications.

The proposed framework is designed to provide tools for autonomous control of mission and guidance of UAVs through the exchange of messages and execution of commands based on the computer vision algorithms. The framework is highly scalable and enables integration with several types of mission topologies. Its architecture enables the integration between one or more UAVs and one or more computer vision algorithms.

In order to validade the proposed framework, it was used in this work to integrate the algorithms proposed for the precision agriculture case study with a COTS UAV hardware used in the field tests.

As a summary of the activities developed in this work, the following items stand out:

- The implementation of the proposed framework was performed;
- The algorithms proposed in the precision agriculture case study were implemented;
- In order to carry out the laboratory tests, videos were acquired using the proposed hardware. These videos were then used as input parameters for these experiments performed in the laboratory;
- The final complete solution, composed of both the framework and the proposed algorithms, was installed in an embedded hardware, however

the proposed framework allows the decoupling of these parts of the system to distribute the processing load between two or more embedded hardware;

- Performance and functional tests were performed with the framework and the developed algorithms in the laboratory;
- Field tests were carried out as a final validation of the proposed solution;

2.1 Objectives and Contribution

The main objective of this work is the development of a framework called "Drone-Control"for mission control and guidance of autonomous UAVs. This framework provides for the exchange of messages between the flight controller and computer vision algorithms, providing multiplicity in the system architecture, both in the number of UAVs that can be controlled and in the number of algorithms that control them.

The proposed framework can also be considered a facilitator in the integration of the algorithms with the hardware, since it has several previously implemented commands that can meet the different needs of different computer vision applications.

Besides the main objective, this work has some secondary objectives that are:

- Development of an algorithm for self-regulated application of agrochemicals;
- Development of an algorithm for guiding autonomous of UAVs in precision agriculture plantations;

Some constraints and elementary requirements were imposed to achieve the objectives:

- The framework must be generalizable, its application must be possible in several areas of knowledge within the scope of UAVs and computer vision;
- All the developed solutions should run on low cost embedded hardware installed in the UAVs;
- All algorithms should run onboard and in flight time;
- The proposed algorithms and their implementations must meet the temporal requirements of the applications they target for;

The main contribution of this work is the conception of the framework, however, the work also presents other contributions that are:

- Algorithm of self-regulated application of agrochemicals;
- Algorithm for detection of crop rows;

- The proposed filter for the pretreatment of the images and later identification of the crop rows;
- The filter to remove the false positive results detected by the line detection algorithm;
- The algorithm responsible for generating the guiding parameters based on previously detected crop rows;

2.2 Work Organization

This work is organized as follows. First is presented in Chapter 3 a study on the different techniques of computation vision that are being used in UAVs. Also in this chapter, the techniques of image processing from the literature that are used in the development of this work are described. Then, in Chapter 4 a more directed study about the related works is presented, as well as the comparison with the present work.

In Chapter 5 the methodology used in the construction of the proposed framework is described. Afterwards, Chapter 6 describes the methodology used in the implementation of the case study in precision agriculture.

Next, the implementation details for the hardware and software architectures used in the development of this project are presented in Chapter 7. In Chapter 8 the laboratory and field experiments are presented as well as the analysis of the obtained results.

Chapter 9 presents the conclusions obtained with the development of the framework and the implementations of the proposed case study. Finally, directions for future work are discussed.

3 BACKGROUND CONCEPTS REVIEW

3.1 Computer Vision in UAVs

Autonomy is characterized as the ability to move and act in the environment in which it is inserted, to perceive the environment through sensors, to adapt or to change the environment, to learn from past experiences, to construct representations of the environment, and to develop a process of interacting with the environment. However, the performance of a computer vision system has limiting and determinant factors. For example, in a visual computing system that uses image processing, there are factors that can affect the image acquisition, such as occlusion, blur movement, rapid pose variation, disordered background environments and onboard mechanical vibration, among others (GONZALEZ; WINTZ, 1977).

In addition, the type of environment is a determining factor for the techniques, algorithms and specific hardware to be used. Internal environments can be controlled and most often rely on solutions based on beacons, proximity sensors and image processing for data acquisition. In this case, as the environment is controlled, the scene illumination can be adjusted and the sensors can be pre-positioned, which facilitates the development and execution of these systems. On the other hand, these environments usually have more obstacles and the space of navigation is much more restricted, which can lead to spend much of the processing for these verification. However, these conditions can be further hampered when the internal environments are unknown by the stand-alone system, as in the case of rescue operations (TOMIC et al., 2012).

In outdoor environments, often known to be variable, with uncontrolled environmental factors, often require solutions based on image processing techniques to provide data acquisition. In addition, in outdoor operations, most navigation systems are based on the Global Positioning System (GPS) (WARD et al., 2016). In this environment, the constant variation of the scene luminance and the large variation of the background in the acquired images are important complicating factors for the operation of the image processing algorithms. Environmental noise is also something that prevents the proper functioning of the sensors that use this form for data acquisition.

On the other side, constructions and buildings block the signal from sensors and global positioning systems making it even harder to handle the exceptions caused by these factors. Each exception treated in a computer vision system has a computational cost that can be high, depending on the time requirements to which this system is constrained. Considering applications using embedded hardware, some projects simply become unfeasible. For this reason, the research in this area is focused in trying to optimize the software for high performance and better use of hardware resources, so that less processing power is required and positively impacting the energy consumption of these systems, which are usually driven by batteries.

In order to understand the current state of research in the area of visual computing used in UAVs, Sections 3.1.1 and 3.1.2 present a study about techniques of computational vision used in drones, along with their relation to the development of mission control software for use in internal and external environments. The text highlights the importance of this research area to provide support for autonomy in the navigation and the interaction for these airborne robots with the environment in which they are inserted.

3.1.1 Indoor Environments

Differently of the UAVs that operate in the external environment, UAVs that operate indoors usually have a reduced size, which impacts on the amount of hardware they can carry and on their autonomy. These factors also impact on a computational capacity, which is further reduced when compared to larger vehicle platforms. The GPS signal is blocked against constructions or buildings. Thus, the indoor UAV can not recognize its location in the environment (SUWANSRIKHAM; SINGKHAMFU, 2017). This impacts on one of the most common applications developed and studied for indoor environments, which refers to how to provide localization to these systems. In addition, indoor environments may have poor lighting and because of this, they require that artificial lighting be installed in the hardware, which can significantly increase battery consumption.

In this context, alternative navigation techniques are required which allow the vehicle to operate successfully in these areas. In addition, it is often desirable that these navigation techniques do not require external infrastructure. One of the most popular techniques for indoor navigation of air vehicles is the use of a laser with track finder, which measures the distance to the objects that are in its focus. With this equipment, it is possible to use the technique known as simultaneous localization and mapping (SLAM) to create a three-dimensional map of the environment and to locate the vehicle in the environment. Inertial navigation with the aid of monocular vision is another paradigm applicable to internal environments. The cameras provide a wealth of information about the environment and they are low cost and lightweight (MAGREE; JOHNSON, 2014).

Under these circumstances, in the work presented by (MAGREE; JOHNSON, 2014) two integrated navigation systems are developed that combine the techniques of visual SLAM, using a camera and laser SLAM with an inertial navigation system. The monocular SLAM visual system has fully correlated characteristics and modelled the vehicle states. The SLAM laser system is based on a scan matching map and leverages the visual data to reduce ambiguities in the vehicle pose estimation. The system is validated in simulation with 6 degrees of freedom and in real flight test. According to the authors, one of the main features of the work is that the system is validated with a controller in the navigation loop. Fig. 4 shows the trajectory traversed by the UAV in the simulation of an exploration mission using the proposed techniques.

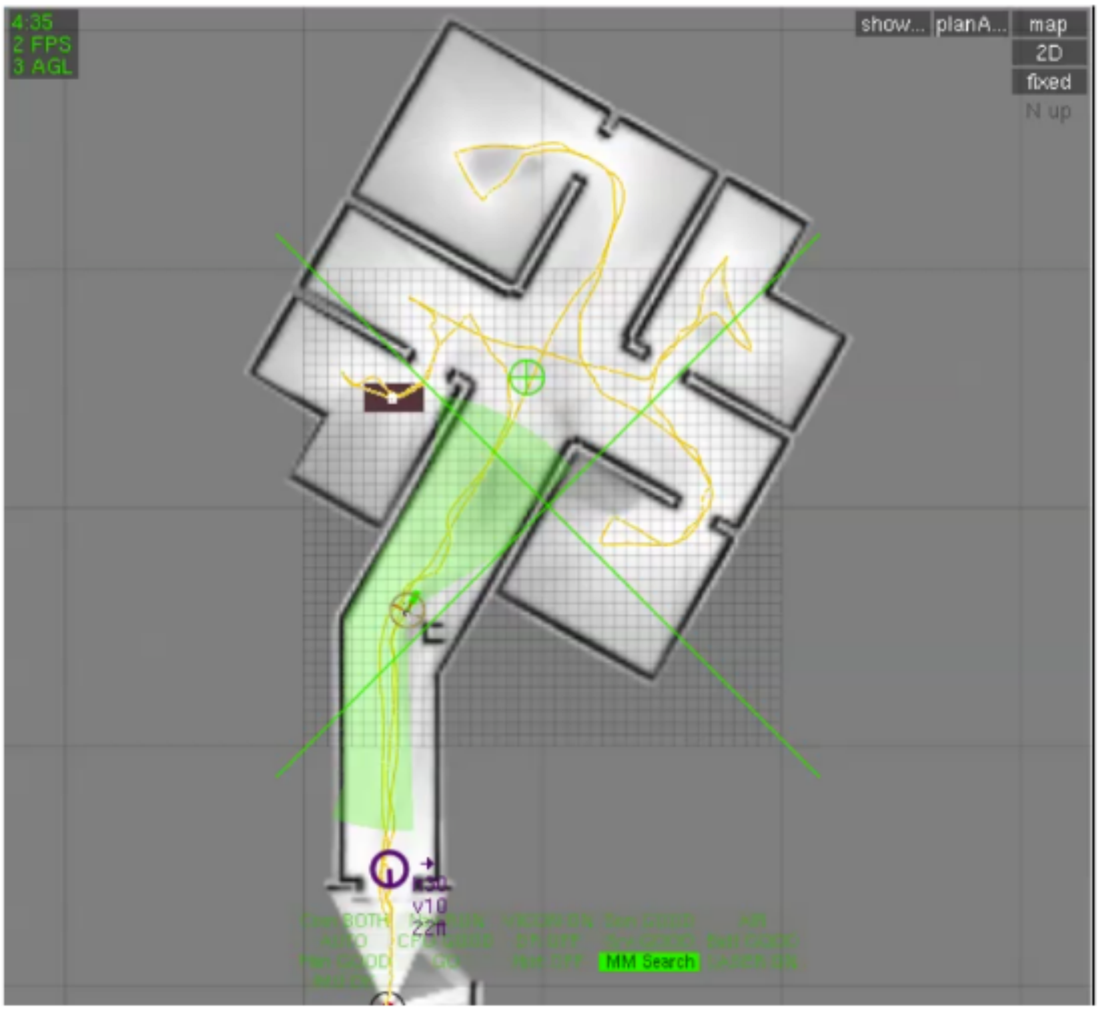


Figura 4: Result of the SLAM map and trajectory of the UAV while exploring a simulated indoor environment (MAGREE; JOHNSON, 2014).

Although, the laser scanner can provide accurate depth information, it is very expensive and heavy. With the use of an RGB-D camera, which can provide RGB and depth images, a simultaneous location and mapping method based on this type of camera is proposed in (LIU; GUO; MENG, 2016). An RGB-D SLAM algorithm was used to locate the camera and construct the 3D map of the used test environment. The developed system can achieve the pose and the trajectory of the camera. In addition, chart optimization and loop closure are adopted to eliminate cumulative errors. However, in this solution, when the camera moves too fast, loss of frames occurs and few points are detected. According to the author the performance of the algorithm and its accuracy need to be improved.

In vision-based navigation systems, the UAV path must be known a priori. Data from the surrounding environment of the UAV's flight path is taken and analyzed to identify the basic features of that path before the flight mission begins. Then the real-time data taken from the on-board sensors during the flight mission is compared to the vehicle's visual memory to identify the known characteristics and estimate the UAV's motion. In this context, the work proposed by the authors in (MOHAMED; PATRA; LANZON, 2011) describes an indoor, inexpensive and simple navigation system using three laser beams fixed to the UAV and pointed to the ground.

A proposed algorithm uses the camera to capture the laser points on the ground and to determine their coordinates. The positions of the laser points are used to obtain complete information on the position and orientation of the UAV. The proposed navigation system can be classified as a vision-based navigation system. However, it does not depend much on the quality of videos taken from the vision camera and does not require an image processing algorithm with high computational cost employed. An illustrative simulation study is conducted to demonstrate the validity of the proposed navigation system. According to the author's results, the proposed system is more efficient over a range of angles between the laser beams because it is possible to obtain the laser beam length without the need for extra sensors or estimation algorithms. This work manages to provide location for navigation through a simple and easy method, but it is not possible to predict collisions with obstacles along the way.

Collision with obstacles are another well known problem in indoor environments, so an indoor location assistance system is proposed by (WU et al., 2017). This work demonstrates the development of an indoor navigation system, specifically for industrial applications that require customized detection technologies to aid navigation. A custom sensing array, with ultrasonic transceivers, was developed to locate the drone's position in a known closed environment and to provide feedback to the navigation system. Six people were recruited to pilot the drone with and without the navigation system in an enclosed room for a predefined target at a known location. Two approaches were used, the first when the UAV was at line of sight of the pilot, and the second with no line of sight. The duration of the flight, the number of collisions and the distance from the target were recorded and used to measure the performance of the proposed application. Using the proposed navigation system, it was possible to reduce the flight duration by an average of 19.7 % during an obstructed line of sight. In addition, the UAV preventative collision detection and navigation has been improved. The navigation system provides a detection radius of 7.65 m and a position accuracy of 1 cm.

A localization technique known as Air-SSLAM was presented in (ARAúJO et al., 2017). This technique uses a stereo camera setup besides two or more cameras that can be used. Fig. 5 demonstrates the operation of the proposed system.

Stereo images are captured, and then the image features are extracted. A

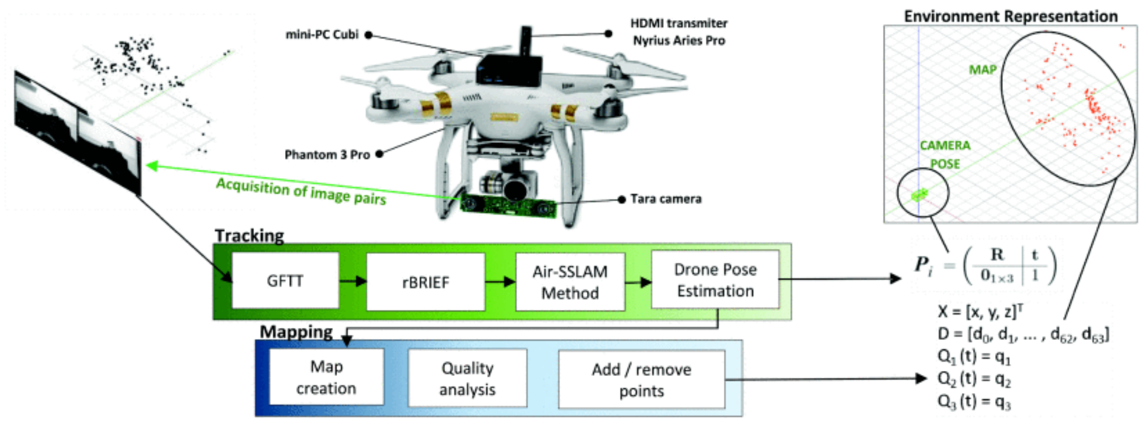


Figura 5: Outline of Air-SSLAM architecture (ARAúJO et al., 2017).

mapping between the features of the images is performed through a descriptor, and an initial map is generated with the depth estimate of each pair of features. This process is then repeated using the initial map as a reference. The maintenance and updating of the long-term map is continuously carried out by analyzing the quality of each correspondence, as well as inserting new features in unexplored areas of the environment. According to the author the work presents three main contributions that are the development of a new method to combine the features in an efficient way, the development of three quality indicators to accelerate the process of mapping and the maintenance of the map with distribution that in its technique, is performed by zones in the images. The results are promising according to the authors, because the method works with constant update of the map of the environment and with an approximate number of 200 features.

Another promising method of indoor navigation is proposed in (BILLS; CHEN; SAXENA, 2011). This method does not require the construction of a 3D model of the environment. On the other hand this technique classifies the type of internal environment in which the UAV is inserted and then uses vision algorithms based on perspective suggestions to estimate the desired direction to fly. Fig. 6 demonstrates the system detecting the escape route in indoor images (corridors). Images are processed with Canny edge detector and the Hough transform is used to find the vanishing point using a proposed grid-based approach.

During the tests with the proposed solution, it was identified that these algorithms require a significantly lower computational power, allowing the UAV to react quickly and navigate through several internal environments.

Until then, techniques have been described that are generally used in uncontrolled or unknown environments. The studies reported in (PHANG et al., 2010) demonstrate the development of a UAV using visual computing techniques to provide vehicle navigation in a controlled environment. The indoor environment used in the tests had colored tracks fixed to the ground. These tracks were detected by the visual computing system and decoded in the form of instructions used to power the vehicle's navigation system. Important data such as attitude, speed and acceleration of the UAV along with real-time vi-

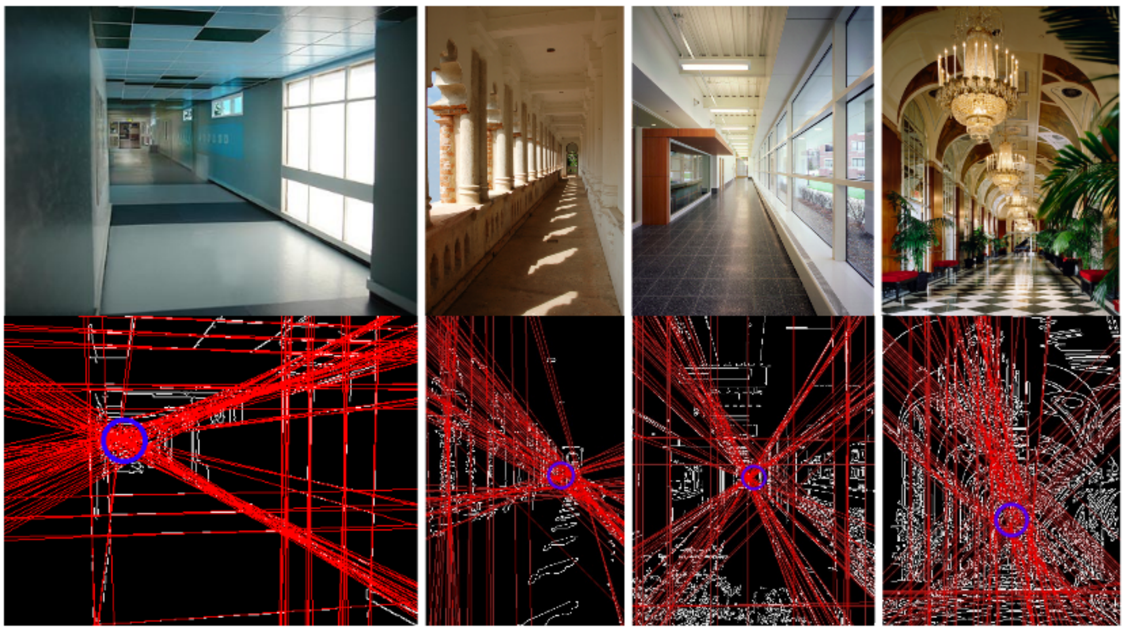


Figura 6: Images of the navigation system running inside a corridor (BILLS; CHEN; SAXENA, 2011).

deo are sent as feedback to the ground station through communication links to give commands and also for monitoring purposes. The algorithms of processing and control by computer vision were evaluated and obtained good performance.

Using a scenario controlled with markers, the system is proposed in (PESTANA et al., 2016) is a completely autonomous solution used to participate in the IMAV 2013 competition (IMAV 2013), which has won the first prize of the competition to the authors. The proposed solution is a system composed of multiple UAVs without centralized coordination, whose UAV agents share their positional estimates. The ability to navigate and detect collisions is a consequence of the behavior of each member participating in the group of UAVs. Fig. 7 demonstrates the system execution scenario.

All processing takes place at a base station outside the vehicles. For each vehicle there is running in the ground station an instance of the proposed architecture that communicates through WiFi with the UAV. Visual markers are used to detect and map obstacles and to improve pose estimation. In the executed tests, each UAV presented the ability to navigate avoiding obstacles and even collision with other members of the system.

In the case of indoor environments, what can be observed from the use of visual computing techniques in UAVs, is that most of these works are designed to develop systems with the ability to navigate and/or collision detection/avoidance in objects or obstacles. Efforts are concentrated in the development of aerial vehicles to carry out the exploration of the environment and interaction with it, preventing missions and optimizing their flight time. Another relevant point to notice is that the research focuses on developing solutions for indoor environments that are generally uncontrolled and unknown, which has the

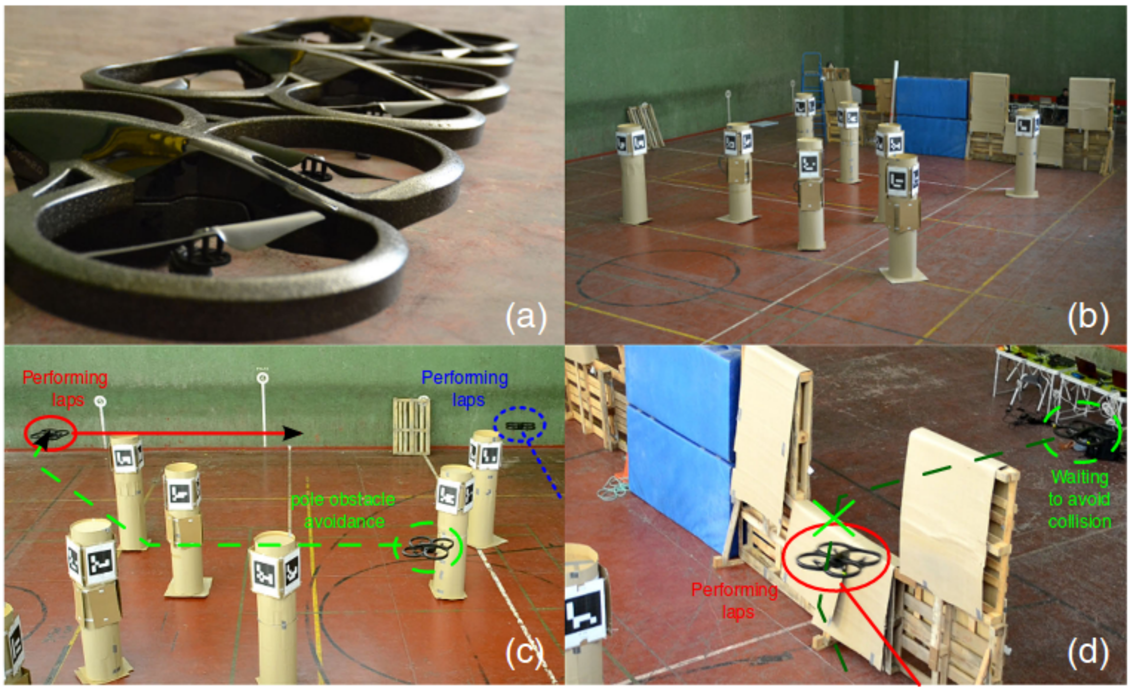


Figura 7: The architecture and the scenario application presented in (BILLS; CHEN; SAXENA, 2011).

ability to increase the level of autonomy but also the level of difficulty in implementing and merging the architecture of these systems. Another important direction of studies that must be highlighted is on platforms that consider the interaction of several vehicles with the environment (and among themselves), being collaborative or independent.

3.1.2 Outdoor Environments

Unlike techniques presented in indoor environments that are restricted to basically navigational problem resolution and collision detection/avoidance, the visual computing techniques employed for vehicles intended for outdoor use have a much broader application scope due to the diversity of the environment and the great number of possible applications to be developed. Because of all this diversity of outdoor environments, vision applications based on position-attitude control, pose estimation, mapping, obstacle detection, target tracking are easily found in the current literature (KANELLAKIS; NIKOLAKOPOULOS, 2017).

However, these techniques tend to have a higher computational cost because they must deal with the sudden variations of the data acquired by the sensors in an uncontrolled environment. These systems also usually have more than one component in their execution loop in order to treat each of the parts of the application to which the proposed technique covers. This point requires a greater effort on the part of the researchers, who usually have to optimize their techniques in an extreme way to be able to meet the temporal requirements of these systems.

For the localization problem, these techniques of outdoor use most often have the availability of reference of global positioning systems to assist in the determination of the position of the UAV. The accuracy of these systems also depends on the number of satellites to which the vehicle has visibility. However, these GPS-based systems do not provide a reliable solution in environments such as urban areas or forests that may reduce the visibility of the vehicle to the available satellites (AL-KAFF et al., 2018).

Currently one of the main computational techniques used in UAVs is the detection of patterns. This technique is so versatile that it allows the development of a great number of applications. A good example of such applications may be the detection of fire areas. These applications are characterized by the architecture shown in Fig. 8.

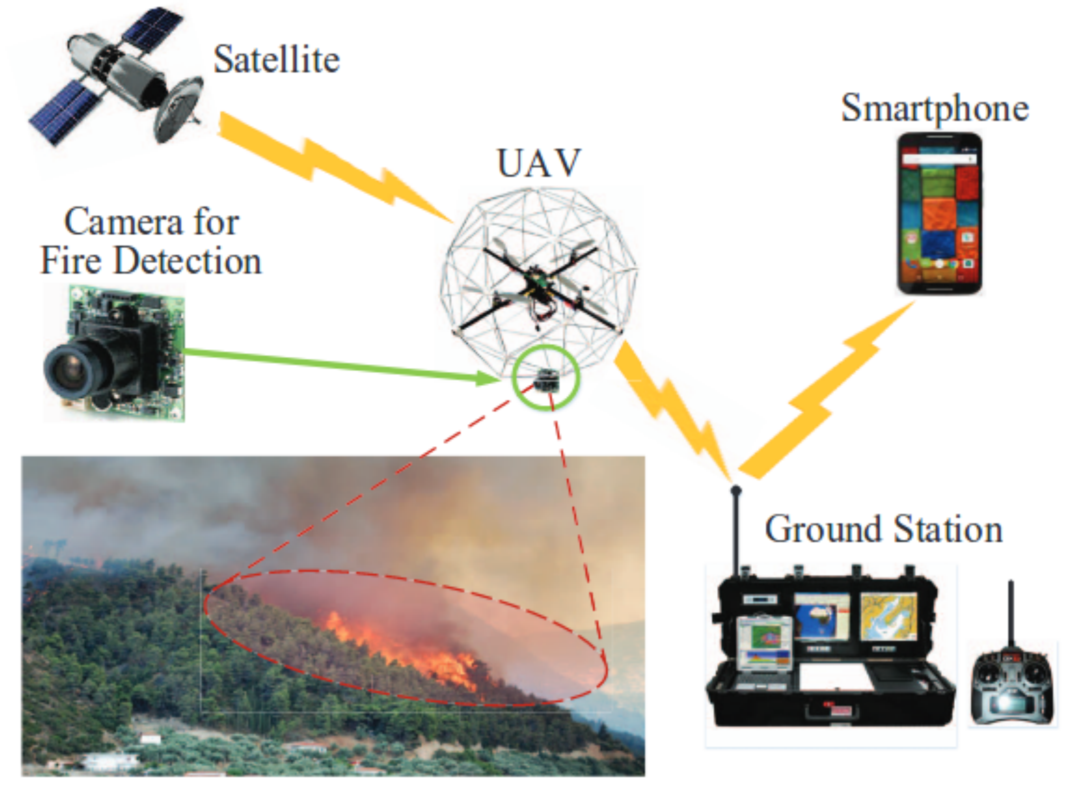


Figura 8: Conceptual vision-based forest fire detection system (YUAN; LIU; ZHANG, 2016).

As described in (YUAN et al., 2016) a method for monitoring and detection of forest fire is presented. The proposed algorithm for fire detection uses color and motion features to improve the performance of forest fire detection and increase its reliability. The motion and color resources are extracted using optical flow method and color-based decision rules, respectively. In this work, experiments were performed using a low cost camera installed at the bottom of the UAV to search for and detect a fire.

In the proposal reported in (YUAN; LIU; ZHANG, 2016), a new method of detecting forest fire using color and motion features for forest fire-fighting applications is presented. The first step of the software consists in the extraction of fire color pixels as regions of fire candidates, making use of the chromatic characteristic of the fire. The second step for this proposed system is the use of an optical flow to calculate motion vectors of the candidate regions. The motion is estimated from the results of the optical flow to distinguish fire from other false positives. By applying a threshold and performing morphological operations on the motion vectors, the binary images are obtained. Then the fires are located in each binary image using the proposed method. The results obtained in the development of this work are promising, in addition the technique presented good performance and reliability in the fire detection.

Following the line of detection techniques, the work developed by (MORANDUZZO; MELGANI, 2014) presents a solution for the detection and counting of cars to be used in UAVs. The proposed method does a step of screening asphalted areas in order to restrict areas where cars can be detected and thus reduce false positives. A resource extraction process based on a characteristic transformation is used to detect the key-points identified in the image. Then, it uses a classifier to determine what parts of the image are cars and what parts are not. The last step of the method is the grouping of the key-points belonging to the same car. Then the number of cars present in the scene is calculated. The results presented demonstrate the efficiency of the algorithm in the detection of cars, but it is suggested by the authors an improvement in the methodology to reduce the number of duplicate key-points to make the algorithm even faster and more efficient.

Inspections also use detection techniques based on computer vision. As presented in (LI et al., 2017), an autonomous UAV-based inspection system is implemented for asset evaluation and fault detection for large-scale photovoltaic systems. The detection of the defects is done through the first order derivation of the Gaussian function and the correspondence of characteristics in the images. Two typical visible defects of photovoltaic modules are characterized, namely, snail trails and dust shading. Field experiments demonstrate that the system can perform inspection and condition monitoring tasks on large scale photovoltaic systems in an autonomous and supervised manner with significantly improved efficiency. The performance evaluation under various conditions confirms that the proposed inspection system can adapt to different slopes within a certain flight height.

There are also several detection-based computer vision techniques to assist UAVs while landing. In the work presented in (JUNG; BANG; LEE, 2015), a robust system with detection and marker tracking technique is proposed. The algorithms are proposed to assist in the task of landing the UAVs. Robust tracking is done using the multiple ellipse connection with a concentric marker. The simulation results show that the algorithm is robust and very accurate.

Another system to aid the landing of UAVs in a moving vehicle with real-time image processing system based on marker detection is proposed in (LEE; JUNG; SHIM, 2016). Marker detection is based on a color detection algorithm with a morphological filter and the tracking orientation is based on the relative distance between the UAV and the vehicle. The captured RGB image was transformed into HSV, so that a noise filter was applied using a threshold. The target tracking algorithm is based on the relative distance between the UAV and the target. The flight test was performed with the vehicle moving in the open air environment, and the algorithm was validated for the autonomous landing.

A standalone vision-based tracking system is presented in (CHENG et al., 2017) and it is used to track a target from a UAV. In order to deal with the losses due to occlusion or loss of the detected object, this work uses a robust and computationally efficient visual tracking scheme in which a correlation filter and the re-detection algorithm. Target status is estimated from visual information. Extensive real-time flight experiments were performed in outdoor environments, where computing is fully implemented in the on-board computer. Experimental results illustrate that the proposed real-time vision-based tracking system achieves the tracking performance required by the application.

Another work that traces objects is proposed in (PESTANA et al., 2014), where an architecture that allows the user to specify an object in the image that the UAV must follow from an approximate constant distance is developed. In the event of loss of tracking of the object in the image, the system begins to hover and awaits the retrieval of tracking or a next detection, which requires the use of odometry measurements for self-stabilization. The proposed software uses the forward-facing camera images and some of the IMU data to calculate the references to the control loops. The results obtained with the work relativity show that the system was able to perform the visual detection with targets of variable size, that the system was able to follow a person with velocity of approximately 2.5 m/s for a time of approximately 45 seconds.

A new method for detecting, locating and recognizing trees with a UAV equipped with a monocular camera is presented in (SHAH; KHAWAD; KRISHNA, 2017). Trees are detected on a frame basis using the latest generation convolutional neuron networks, inspired by the recent rapid advances shown in the literature. The trees are detected and their position is marked on the global positioning system. Localized trees are segmented, characterized by resource descriptors, and stored in a database that contains their GPS coordinates. The trees detected on later flights are compared with the data in the database to avoid duplication of information. The proposed method is also able to identify if the trees are absent from the expected locations with GPS marked, allowing to immediately alert the authorities concerned about possible illegal deforestation.

In addition to counting trees, other vision applications in UAVs such as counting other types of commercial plants help to generate production statistics in precision agriculture. In this context, the work presented in (MALEK et al., 2014) demonstrates an economical and attractive technology for the automatic detection of palm trees to be used in UAVs. From the in-flight image, it first extracts a set of key-points using the algorithm to detect features. Afterwards, these key-points are analyzed with a trained classifier in a training data set. Finally, the palm trees are identified. Then, to capture the shape of each tree, it merges the key-points with an active contour method. Finally, the texture of the obtained regions are analyzed with local binary patterns to distinguish palms from other vegetation. The results of the experimental tests were acquired in two different farms and confirm the promising capabilities of the proposed structure in the palms detection.

The work presented in (FREITAS et al., 2016) demonstrates an rigorous study on the use of embedded hardware coupled to a UAV to perform real-time image processing. A POI detection algorithm has been proposed to perform the tests on the hardware. In the article it is shown that it is possible and feasible to use a low cost processing board in a mini UAV for image processing. In this work, a practical application is demonstrated using image processing in UAVs, it is shown the preliminary results of an application for the detection of power lines in order to perform the autonomous guidance of the UAV on the lines for cable inspection for example. Fig. 9 shows an image illustrating the results in the detection of three electrical power lines. Notice that the system successfully perform the detection, even considering a noise environment in the background.

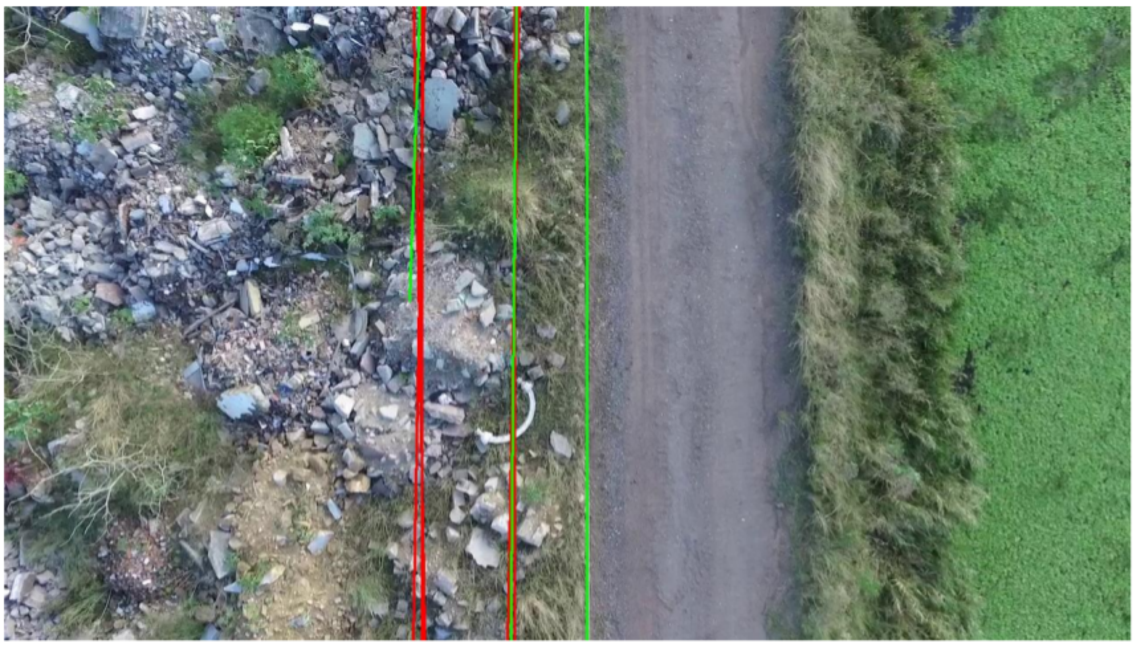


Figura 9: Electrical power line detection application (FREITAS et al., 2016).

Another emerging technique is the monitoring and tracking of ice for use in marine ice management applications. This technique was reported in (LEIRA; JOHANSEN; FOSSEN, 2017) and it is aimed to be used to detect and track the movement of icebergs and other ice sheets in an arctic environment autonomously. An occupancy grid map algorithm and a locations of interest generator are used, coupled with a mission control system. According to the developers of the project, one of the contributions of their work is the interface of the algorithm with a vision-based object detection module to generate an occupancy grid map of a real-time predefined search area using on-board processing data. A generator of places of interest has also been developed, generating locations where the UAV should investigate based on the occupation grid map. The results were obtained based on test flights performed and it was possible to verify that the system was able to successfully create an occupation grid map based on the automatic segmentation of thermal images on-board and in real-time, in ice regions and in regions without ice. However, in the tests carried out it is possible to observe some delays in the communication and the fact that the mission controller was implemented in a ground base station.

As can be observed, the visual computing applications used in outdoor environments have a great diversity due to the number of possible applications. Advances in technology increasingly allow UAVs to be used in a variety of tasks and they are thus changing the way some processes are conducted, such as in the case of electric transmission line inspections or even in the inspection of photovoltaic panels. However, this huge number of applications requires systems with increasingly complex and computationally costly architecture. This is one of the biggest problems of computer vision outdoor applications. There is a need to use several algorithms engaged in an application loop competing for a processing time on a low cost processor. To distribute tasks and processing, studies from this moment on start and turn to applications available to multiple execution agents.

3.1.3 Other Approaches Handling both Indoor and Outdoor Environments

In addition to the most common approaches, there are studies being developed to perform in indoor and outdoor environments, such as the work demonstrated in (CHEN; GUO; LI, 2016), which presents a complete strategy of tracking a terrestrial target in complex internal and external environments with a UAV based on computer vision.

Navigation techniques have also gained focus on multi-environment studies, as demonstrated in (CHOWDHARY et al., 2013). The proposed architecture efficiently combines visual information from a monocular camera with measurements of inertial sensors. The presented algorithms are validated on multiple platforms in real conditions, through a 16 minute flight test, including a stand-alone landing, of a 66 kg rotorcraft UAV operating in an uncontrolled external environment without using GPS and through of a micro-UAV operating in a disorderly interior environment.

Another interesting proposal in this context is presented in the work reported in (HUH; SHIM; KIM, 2013), in which an integrated navigation sensor module, including a camera, a laser scanner, and an inertial sensor, for UAVs to fly both indoors and outdoors environments. A real-time navigation algorithm based on estimating algorithms is proposed. The algorithm merges the image features with laser track data to estimate vehicle status and position. The proposed on-board navigation system can provide real-time 3D navigation without any pre-assumptions. The experimental results demonstrate the performance of the proposed system and prove their efficiency for multiple environments.

A proposal to identify people in natural accidents and to conduct scanning missions to identify and trigger the rescue of endangered people is presented in (APVRILLE; TANZI; DUGELAY, 2014). This work is in the initial phase of development, presenting the contextualization of different applications and various scenarios in which the UAVs can act. The initial results demonstrate the level of autonomy that could be reached by using computer vision techniques aiming both indoor and outdoor environments, but more concrete evaluations and results are still under the way.

As can be observed, these techniques tend to be more robust in terms of accuracy, but also have a higher computational cost because they have to deal with the disturbances and disparities in the data collected by the sensors due to the great variation of the scene and the environmental conditions. However, they can meet a larger number of requirements for a particular mission, both indoors and outdoors.

3.2 Image Processing Techniques

This section presents the image processing methods available in the literature that were used to implement this work. The Section 3.2.1 demonstrates the method used in the development of the case study concerning the self-regulated application of agrochemicals. In Sections 3.2.2 and 3.2.3 are presented methods used in the development of the proposed guidance algorithm.

3.2.1 NDVI

The Normalized Difference Vegetation Index (NDVI) is a graphical indicator that can be used to analyze remote sensing measurements (MYNENI et al., 1995). According to (WANG et al., 2007), NDVI is an index composed of mathematics of spectral bands, caught by sensors like satellites, RGB and infrared cameras. From the NDVI it is possible to discriminate the health of vegetation, according to the described in (BASSO et al., 2001). As shown in (WANG et al., 2007) and (TORRES-SáNCHEZ; LóPEZ-GRANADOS; PEñA, 2015), the NDVI reflectance coefficient can be calculated from (1).

$$NDVI = \frac{NIR - R}{NIR + R} \tag{1}$$

where NIR is the infrared channel value and R is the value of the red visible channel of the pixel being analyzed.

The result obtained in the application of this index is a value comprised between the scale of $[-1, 1]$. The analysis of this measurement is performed as shown in Fig. 10. The higher value represents the greater the amount of green mass present in the area analyzed or the health of the plantation is higher. The lower value represents the lower presence of green mass, which indicates that there are no plants or that the health of the plantation is bad.

Considering the usage of automated UAVs to perform spraying of pesticides and fertilizers, the usage of NDVI would be suitable to provide information to deliver the chemicals according to the identified needs.

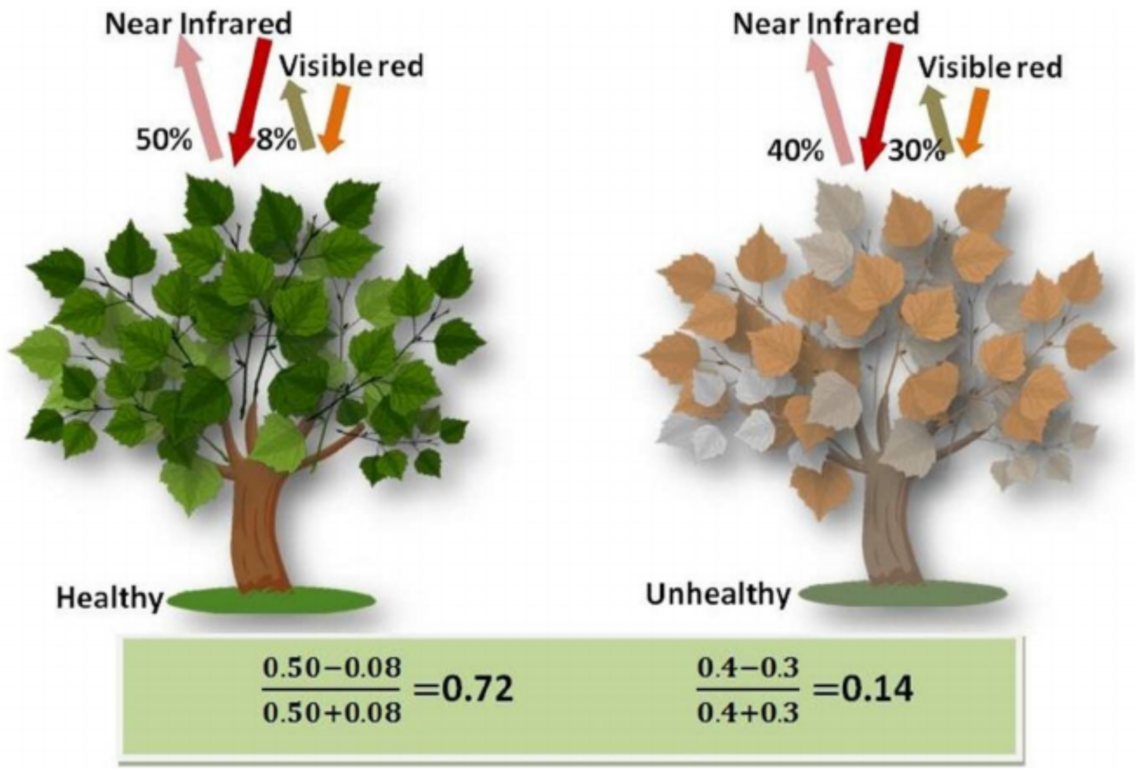


Figura 10: Example of NDVI calculation (WU et al., 2014).

3.2.2 2G-R-B Transform

The 2G-R-B Transform is proposed by (WOEBBECKE et al., 1995) and represented by (2). This transform consists of a mathematical of spectral bands to calculate the green excess for all points in the image. The result obtained with the use of the transform in (2) is a value between the scale $[0, 255]$ and represents the gray intensity in the gray scale image.

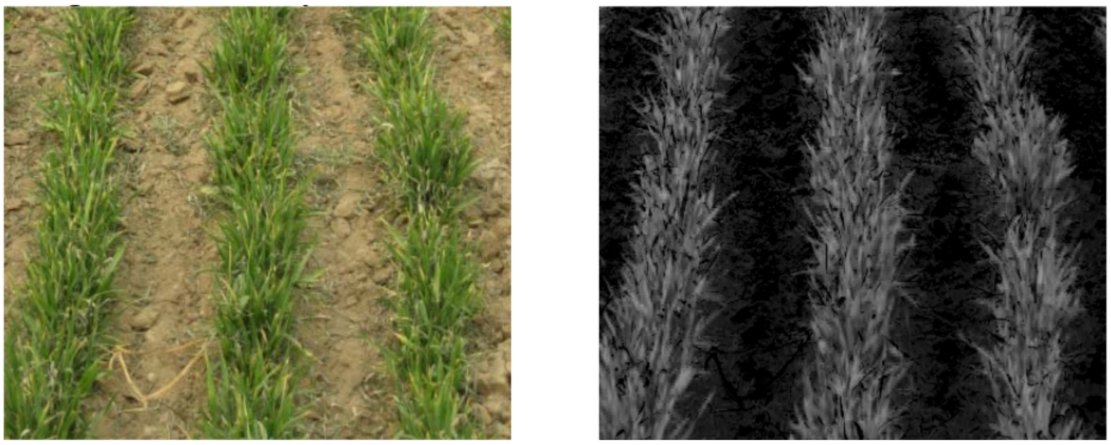
$$f(x,y) = \begin{cases} 0 & \text{if } 2G \leq R + B \\ 2G - R - B & \text{if } Others \\ 255 & \text{if } 2G \geq R + B + 255 \end{cases} \tag{2}$$

As shown in (JIANG; ZHAO; SI, 2010), this transform was efficient in relation to the luminance variation, presenting good results when used in images with high incidence of solar rays and also the images captured in days with cloudy weather. The luminance variation is strongly present in the images used in this work. An example of using it transform is presented in Fig. 11.

Although the transform is efficient in relation to luminosity, it emphasizes in the grayscale image beyond the desired plants other objects and invasive plants that have the predominance of green color. This problem needs to be addressed with other solutions such as application of thresholds.

3.2.3 Hough Transform

The Hough Transform is proposed and patented initially by Paul Hough in 1962 for the identification of patterns (HOUGH, 1962). This transform had its



(a) Original image. (b) Gray scale image.

Figura 11: 2G-R-B Transform example (JIANG; ZHAO; SI, 2010).

functionality extended by Richard Duda and Peter Hart in 1972 (DUDA; HART, 1972) to be widely used in the area of visual computing for the identification of lines, circles and ellipses.

The operation of this transform consists of applying (3) to a grayscale image and obtaining the new coordinates for the parameter space (ρ, θ), where ρ is a distance from the origin to the nearest point on the line, and θ is the angle between the x axis and the line connecting the origin to the point closest to the line.

$$\rho = x \cdot \texttt{cos}(\theta) + y \cdot \texttt{sin}(\theta) \tag{3}$$

As a result, each line in the original image is represented by a sine in the parameter space. A voting scheme is defined in which each vote represents a senoid crossing a point. These votes are stored in the accumulator space. The points with the highest number of votes are possible candidates for straight lines in the original image. Incrementally sorting these points by the number of votes and then converting them back to the vector space (x, y), the lines of the image are identified. The results of application of Hough Transform in an image are present in Fig. 12.

Finally, the algorithm returns a list of sorted and ordered lines according to the number of votes, however, depending on the application and the level of details present in an image, this does not guarantee that the identified lines are the desired lines. To eliminate these false positive results identified, some strategies need to be applied, however this varies according to the application being developed.

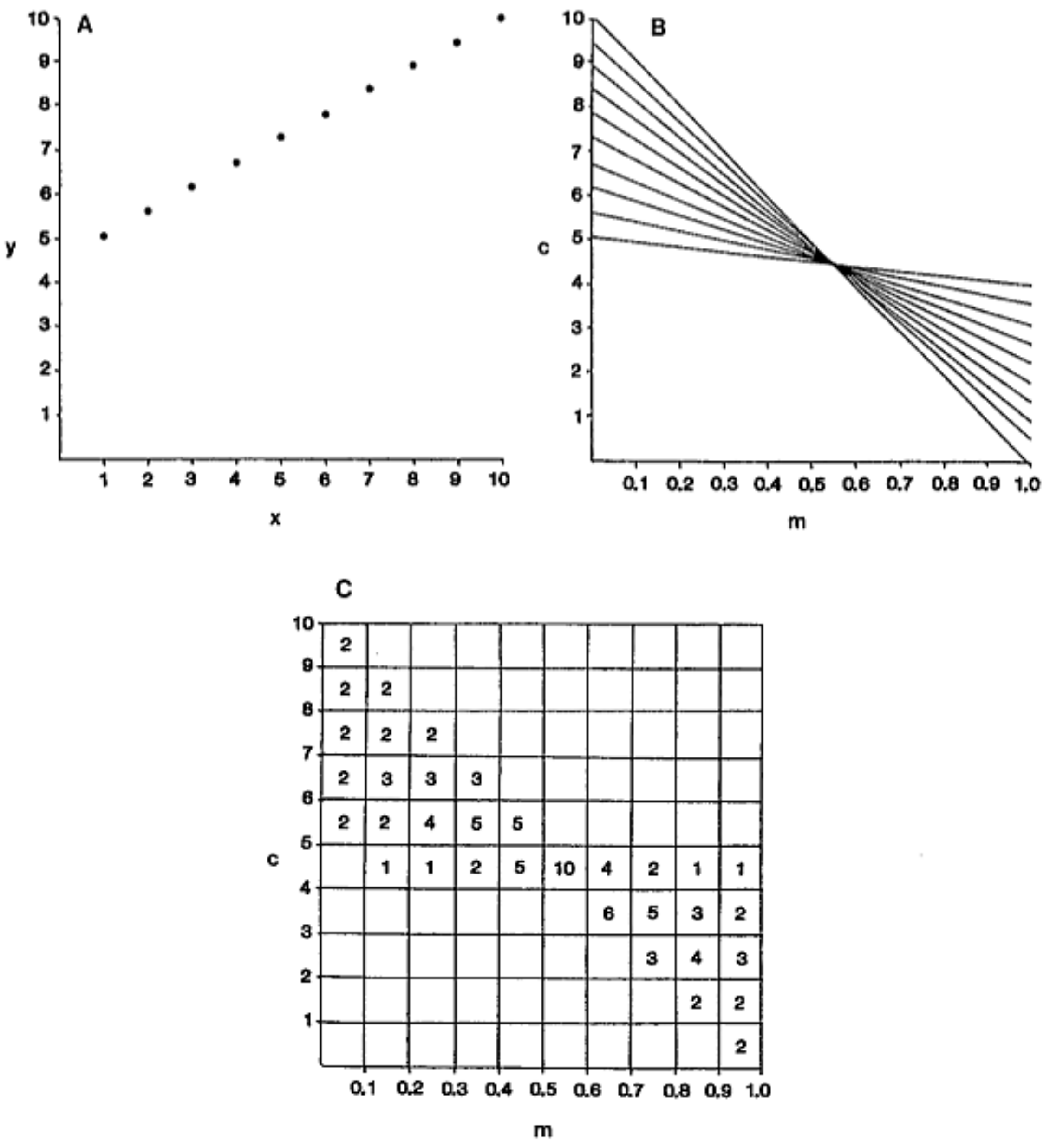


Figura 12: The basis of the Hough transform for line detection: (A) (x, y) point image space; (B) (m, c) parameter space; (C) accumulator space corresponding to (B) (ILLINGWORTH; KITTLER, 1988).

4 RELATED WORKS

A new technique for vision-based navigation is presented in (ZHANG; LIU; WU, 2011). In this technique, the problem with position estimates is formulated as a tracking problem and solved by a particle filter. The results of the simulation indicate that the proposed technique provides good results of position estimation, with no apparent accumulation of errors and robust for the variations of the path traveled.

A framework for the detection of trails based on support vector machine-based is proposed in (LIU et al., 2017). The proposed system is based on vision and scene comprehension by a UAV operated in unstructured external environments. A simple linear iterative grouping superpixel segmentation algorithm is used in the proposed system structure to ensure the accuracy of the segmentation of the scene. Visual detection of significant objects or persons was performed using a multibox detector algorithm. Experimental results show that the proposed framework is a practical solution for a UAV to conduct autonomous trail detection and tracking in outdoor environments, and also able to enhance the practicability of outdoor scene understanding.

A new vision tracking and tracking system is presented in (MINAEIAN; LIU; SON, 2016) and plans to make use of different capabilities of a cooperative vehicle team.The scenario considered in this paper is a team of an unmanned aerial vehicle (UAV) and multiple unmanned ground vehicles (UGVs) tracking and controlling crowds on a border area. A custom motion detection algorithm is applied to track the crowd of the mobile camera mounted on the UAV. The proposed localization algorithm converts the locations of images of the crowds into their real-world positions, using perspective transformation. It is also estimated the geographical locations of the detected individuals. The experimental results demonstrate the effectiveness of the motion detection algorithm and the algorithm of human detection for UGV as well as the location of detected points in the real world.

In (YIN et al., 2016) a robust visual detection-learning-tracking framework is proposed for the autonomous refueling of UAVs. Two classifiers are built in the proposed structure. The first classifier is trained offline to detect the drogue object of aerial refueling. The second classifier is structured for online operation tracking the drogue object. A strategy of combining the classifiers is proposed. The results were obtained based on tests in several challenging video sequences that did not show the effectiveness and robustness of the proposed framework. However, the FPS rate is not satisfactory and the authors propose as future work the implementation of this framework in hardware.

A method of navigation of UAVs based on vision and with prediction of failures in the use of GPS is proposed in (ZHANG et al., 2014). If GPS positioning fails, scene matching techniques are used in the dynamic map for real-time latitude and longitude information. UAVs can perform a variety of intelligent missions based on the proposed method. The system was developed in Visual C ++ and is also used as an application simulation environment. In the simulation tests performed the system achieved the performance and accuracy required by the application. No practical tests were performed.

An approach to the problem of obstacle detection and vision tracking for the navigation of UAVs is proposed by (WU; SUI; WANG, 2017). In the detection phase, the object of interest is automatically detected and located from a calculated saliency map through the background connection of the image in each frame. In the tracking process, a Kalman filter is employed to provide a rough prediction of the state of the object (KALMAN, 1960), which is further refined through a local detector incorporating the saliency map and time information between two consecutive frames. The proposed approach does not require any manual initialization for tracing, and has achieved expected tracing performance for the sequences of images tested. Although the proposed tracker is very good at most of the tested image sequences, but it can not detect partially occluded objects.

It is proposed by (RAJA, 2011) an automated landing system for a UAV using image reference technique. The images are captured in the landing phase, where the UAV slides through a specific angle to land. A feedback control loop using the gyroscope signal response is used. The UAV captures frames during its flight and compares them with the frames of reference. With this you can calculate the speed, position and angle of inclination of the landing location. The results showed that the developed system was effective for the test cases used.

A low cost, open source system where all image processing is performed onboard the UAV using Raspberry Pi 2 connected to a camera is described in the work reported by (CHOI et al., 2016). Raspberry Pi and flight controller are physically connected via serial and communicate via MAVProxy. A proposed algorithm detects a target. If the target is detected, the position of the object in the frame is represented in Cartesian coordinates and converted into estimated GPS coordinates passed to the flight controller. The results show the accuracy of the algorithm is 99 % to detect objects of interest and the UAV is able to navigate and make the decisions onboard.

As demonstrated in the related works above, a large number of visual computing applications have been integrated to run omboard in UAVs. These studies aim to solve problems and enable the use of these vehicles in various applications and areas of knowledge. However, there is no standardization or tool that allows the integration of these algorithms, systems and frameworks proposed in an easy and robust way with the flight controllers.

Therefore, this work solves this problem by proposing and developing a framework that can standardize and enable the integration of several computer vision systems with different kinds and models of flight controllers arranged in heterogeneous UAVs. The proposed framework abstracts the various actions performed in flight in commands with simple parameters.

In addition, its architecture enables the development of other commands needed to meet different applications. The proposed framework architecture also predicts multiplicity in the system, both in the number of UAVs and in the number of connected computer vision algorithms.

In the Sections 4.1 and 4.2 are presented the related works to the developed case study in precision agriculture, where the framework was used as the basis for the development of the proposed applications.

4.1 NDVI Related Works

A proposal of using UAVs to perform pesticides spraying on crop fields is presented in (FAIÇAL et al., 2014). The proposed solution is based on a cooperation between UAVs and a wireless sensor network deployed on the field to determine the exact locations where the pesticides should be sprayed, according to the readings of the sensor nodes. It is possible to state that the sensor nodes deployed on the ground drive the UAV to locations where the spraying is demanded. Besides the same application goal, their approach is very different to the one here presented once our proposal is entirely contained in the UAV embedded system, without the need of information provided by an external entity (in their case, the sensor nodes on the ground).

An algorithm aiming to automatically detect wildlife with UAVs is presented in (WARD et al., 2016). This algorithm, which is written in Python, uses a pixel based object detection approach. It makes use of a thermal camera and OpenCV, an open source computer vision library wrapped for Python. The results show that the system is capable of autonomously locating animals from a predetermined height and generates a map showing the location of the animals. Despite the challenging image processing task reported in this work, its results are not used to drive actuation decisions as the one presented in this current paper.

A real-time system capable of classifying plant diseases was proposed in (AKRAM et al., 2017). The system developed in this work is based on an image processing algorithm that transforms the image into three color spaces, which are simultaneously processed. The algorithm constructs a vector of characteristics used to identify diseases. In addition, an On-Chip communication architecture was proposed that allows efficient interconnection between the three digital signal processing cores, each processing its own color space, which provided a better performance to the system. According to the authors, this work can be considered a basis for the development of an autonomous system that will not only be able to classify the diseases in the leaves of plants, but recommend the appropriate pesticide with the least human intervention.

The work reported in (ZHOU et al., 2015) presents an efficient road detection and tracking framework for UAV captured videos. According to the authors, the GraphCut segmentation method of (BOYKOV; VEKSLER; ZABIH, 2001) was used to extract initial road areas, and then track the road areas in the subsequent frames by combining a fast context-aware homography-alignment road tracker and an online GraphCut approach for road detection. The results indicate the effectiveness of the proposed framework, with the precision of 98.4% and processing 34 frames per second for 1046 x 595 videos in average. The main difference of this solution in relation to the work here proposed is due to the difference in the target application. While the road detection can segment the image focusing areas of interest, the precision agriculture application addressed in this paper must analyze the whole images.

A new technique for power lines detection from UAV remote sensed images acquired by a GoPro camera is presented in (KN et al., 2015). The use of the pixel intensity-based clustering method is performed followed by morphological operations. K-means clustering (MACQUEEN, 1967) is applied for clustering and the number of clusters is automatically generated using Davies-Bouldin index (SENTHILNATH et al., 2016). The authors classify the results as good quality both from a visual perspective and from the performance metrics, achieving an efficiency level of 99%. Again, the difference in relation to the current work is due to the particularities of the different target applications.

The work presented in (O. DEMERECI M. VARUL; ODABAS, 2015) reports a remote sensing and image processing method for precision agriculture applications. This paper aimed to identify the number of plants on the field based in images taken by UAVs using image processing methods. Performing pre-processing of the images and using K-Means as clustering method, it was obtained 86.3% accuracy rates, and using color-based segmentation method it was obtained 89.2% accuracy rates. The real time requirements affecting their problem are not as hard as those related to the chemical spraying scenario addressed in this current paper.

Taking part of some of these works as a base, an integrated real-time image processing system was developed, supported by the NDVI algorithm for use in UAV-based precision farming applications. In this case study, a low-cost hardware architecture based on (WARD et al., 2016) was abstracted. The abstraction techniques of (ZHOU et al., 2015) and (KN et al., 2015) allow the mathematical optimization of the proposed NDVI algorithm. Abstracting techniques from (ZHOU et al., 2015) and (KN et al., 2015) allows the mathematical optimization of the proposed image processing algorithm. Following (WARD et al., 2016) and (O. DEMERECI M. VARUL; ODABAS, 2015) it was possible to abstract techniques for the use of infrared camera, and from (O. DEMERECI M. VARUL; ODABAS, 2015) e (AKRAM et al., 2017), techniques for image processing with illumination variations present in images used in precision agriculture applications. Thus, the proposed solution presented in this case study advances the state of the art in the area by combining parts of different approaches into a new system proposal incorporated for the autonomous control of a UAV sprayer actuators.

4.2 Crop Row Detection Related Works

An efficient method for detection of crop rows that uses a dynamic programming technique to combine evidence of image and prior knowledge about the geometric structure that is searched in the image is proposed in (VIDOVIĆ; CUPEC; HOCENSKI, 2016). The method is able to accurately detect both straight and curved crop lines. The proposed approach is experimentally evaluated in a set of 281 real-world camera images of corn, celery, potato, onion, sunflower and soybean crops. The proposed algorithm achieves satisfactory accuracy, but its performance on an Intel Core i5 processor desktop reaches about 1749.1 ms for images at resolution of 320x240 and 4527 ms for resolution of 640x480. The author proposes as future works the implementation of the algorithm in GPU and implementation in a real application in an autonomous machine.

The gradient-based Random Hough Transform (RHT) algorithm is proposed in (JI; QI, 2011) to perform the detection of crop rows. This work proposes to improve the calculation speed and reduce the computational cost of the Crop Row Detection method. It was evident from the experiments that the gradient-based RHT method reduced computation and effectively improved the calculation speed. The performance of RHT applied to images with a resolution of 400 x 300 was 0.802 s while for Hough Transform it was $1,715$ s. According to the author, RHT can effectively improve detection speed.

An automated specialized system for precise detection of crop rows in corn fields based on images acquired from a vision system is proposed in (GUERRERO et al., 2013). The algorithm was developed based on two modules that use image processing techniques. The first is the separating the green plants of the rest of the image. The second is based on the geometry of the system where the expected crop rows are mapped on the image, and then a correction is applied through an estimator and a linear regression. The experiments were carried out using as hardware specific equipment for real-time processing, which has an Intel Core i7 processor and an FPGA. The total processing time for images with resolution 1392x1038 was tested for different estimators, resulting in averaged $9,568$ s for the worst case and 0.476 s in the best case.

A new robust crop row detection algorithm for an agricultural machine guidance system is proposed in (JIANG; WANG; LIU, 2015). The algorithm consists of five steps that are the gray level transformation, binarization, candidate center point estimation, confirmation of real central points and detection of crop rows. The algorithm is based on the creation of regions of multiple interest during the procedure of evaluation of candidate points based on a constraint of geometry that the inter-row space for algorithm optimization. The tests were performed on an Intel Core 2 Duo computer and the algorithm took an average of 61 ms to recognize all of the crop rows for images in the size of 640x480 while the detection rate reaches 93 % of the real crop rows in the images.

A new method for detecting curved and straight crop rows in images captured in maize fields during the initial growth stages of the plants is proposed in

(GARCíA-SANTILLáN et al., 2017). The proposed method was designed with the necessary robustness to deal with adverse situations and consists of three phases: image segmentation, identification of starting points to determine the beginning of harvest lines and detection of crop rows. The experiments were performed on an Intel Core i7 computer and the algorithm ranged from 86.3 % to 92.8 % in the detection rate, taking about 0.65 s to process each image from the test databese of the used images.

A method for detecting crop rows based on mathematical morphology is proposed in (JINLIN; WEIPING, 2010). The crop rows are obtained with the least squares method, according to the centroids of the individual target cultures and/or the center points of several crop pixel lines with a continuous region in the image. The results were obtained by running the algorithm on a PC with AMD Turion 64 X2 TL-50 dual-core processor, showing that the algorithm leads the robot to drive along the line with a high precision 29.5 mm and with an expected processing speed between 0.7 s and 1.3 s per frame, which, according to the author, meets the needs of image processing for the application.

An approach for guiding agricultural robots to perform different types of operations, such as weed removal, spraying and fertilizing is used in a machine vision based crop rows detection system is proposed in (JIANG; ZHAO; SI, 2010). The image preprocessing was used to obtain the binarized image, then the binarized image is divided into several line segments; third, a vertical projection method was presented to estimate the position of the crop rows for each frame. The lines were detected by the Hough transform. The algorithm performs in 70 s to determine all crop rows. The tests were performed with images at resolution of 400x300 on a PC with a processor of 1.8 GHz.

Comparing the here proposed solution with the related works found in the literature, it is noticed that the proposed solution presented better performance in the identification of the crop rows. Comparing the different methodologies, it can be observed that the implementations that use the Hough Transform to detect the lines are usually those that have high computational cost. In this case study, this problem was solved due to the filter pretreatment of the image and filter of lines applied after the Hough Transform. Once using them, it is possible to adjust the parameters of the Hough Transform in order to improve its performance. Another important observation is that this case study was implemented in a low cost embedded hardware, suitable to small scale UAVs, and even assuming this condition, it achieved a good performance even compared to the implementations in which the tests were performed in desktop PCs. The accuracy of the proposed solution for the test scenarios used in the validation phase also reached the 100 % rate of the actual crop rows identified in all test cases. The proposed system was tested in a real application and obtained the expected (desired) results.

5 PROPOSED FRAMEWORK DRONE-CONTROL

The proposed framework called "Drone-Control", is a fundamental tool to provide the integration between computer vision algorithms with the UAVs, and it can perform the control of mission and guidance of the UAVs by sending commands to the omboard navigation system. A mission can be characterized as a set of commands that have to be executed. This chapter describes in detail the components of the proposed framework, which are the messages, the commands, the server and the clients.

5.1 Messages

The messages are used to exchange information and parameters between the architectural components that are linked to the framework. Through them it is possible to send and receive commands and parameters that allow the construction of the applications.

The format of these messages follows patterns determined by structures called JavaScript Object Notation (JSON) (BRAY, 2014). JSON is an independent open standard format for parsing data. The exchange of data can occur between systems developed with the same computational language or with different languages. In short, a JSON object is a textually presented array. Fig. 13 demonstrates an example of JSON message that can be sent to the framework by a computer vision client algorithm.

```
1 {
2     "command": "setPosition",
3     "args": {
4         "x": 10,
5         "y": 5,
6         "z": 2,
7     }
8 }
```

Figura 13: Message example.

The framework preset two elements for the array structure that makes up a message. The elements are *"command"* (Fig. 13, line 2) and *"args"* (Fig. 13, line 3). The *"command"* element is a string that indicates the name of the command or function that should receive the parameters of the message. The *"args"* element is a sub-array that holds any type of information and input parameters required for the development of a command. It can be appended to the element *"args"* 0 or n parameters.

The process of encoding a message consists of converting an array present in a program implemented in a given language to the array in textual format. This textual element is then called message. The decoding process consists of converting the textual element to an array or code object of a given language. In the implementation of the proposed framework, the native libraries of languages were used for the coding and decoding processes of the messages, except for the C++ language that does not have this library natively.

5.2 Commands and Commands Library

A command library is linked to the framework in a class form. The class that represents the command library is called *DCCommands*. An example of this class of commands can be seen in Fig. 14. The commands in the framework constitute implementations of methods in the class *DCCommands*. The name of the commands is equivalent to the name of the methods implemented in the command library and should be indicated in the *"command"* element described in the Section 5.1.

```
 1  #!/user/bin/python
 2  class DCCommands:
 3          def setPosition(self, args):
 4                  x = args["x"]
 5                  y = args["y"]
 6                  z = args["z"]
 7                  ...
 8                  return None
 9          def getPosition(self, args):
10                  ...
11                  return """{
12                          "x": """+ pos.x +""",
13                          "y": """+ pos.y +""",
14                          "z": """+ pos.z +""",
15                  }"""
```

Figura 14: Library of commands.

All methods implemented in this class should receive as input parameter the array of arguments already described in Section 5.1. The access to the value of the parameters of this array can be done within the methods as shown in lines 4, 5 and 6 of Fig. 14. There are basically two types of methods, those that return arguments and those that do not return. Even methods that do not return arguments should use the return command to return *None* (Fig. 14, line 8).

This indicates to the framework that there is no need to return a response message to the client algorithm that requested execution of the command. If there is a need to send a response, the return command must receive a string representing the argument array (Fig. 14, lines 11-15). The response parameters do not have to be the same as the input parameters.

Each command is independent and can connect to different types of hardware. There are be commands to attend and execute actions referring to all hardware instaled in the UAVs, such as a subsystem responsible for the spraying of agrochemicals. In addition, data and parameters can be shared between commands using global variables in the command library.

Currently the integration with the flight controllers of the UAVs is implemented in the command base through the use of the Micro Air Vehicle Communication Protocol (MAVLink) protocol (MEIER et al., 2013) running on established serial connection. For mission control and autonomous guidance, such as the implementation of the case study proposed in this project, several commands have been developed. The developed commands serve to control positioning, take-off, landing, return to home position, camera movement, among others used for testing.

5.3 Server

The server is the basis for the operation of the framework. The server is responsible for connecting to the hardware and initializing all processes for connecting to the client-side computer vision algorithms. Within the framework there may be one or more instances of the server, each of which must be responsible for connecting to a specific vehicle. The execution flow of the server can be seen in Fig. 15.

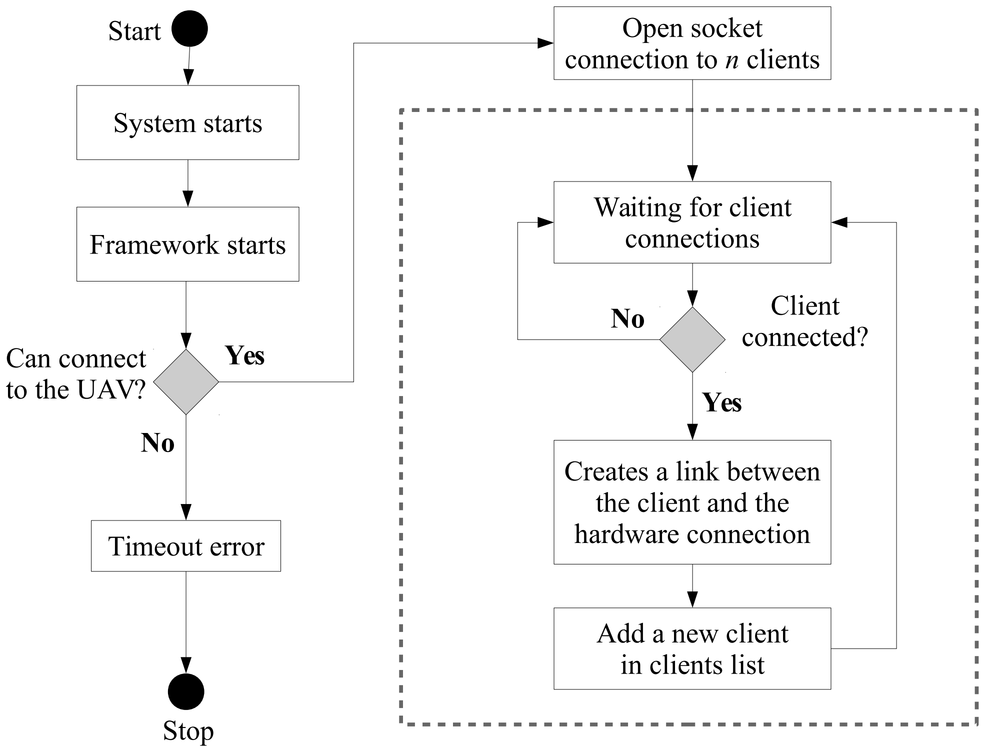

Start
System starts
Framework starts
Can connect to the UAV?
Yes
No
Timeout error
Stop
Open socket connection to *n* clients
Waiting for client connections
No
Client connected?
Yes
Creates a link between the client and the hardware connection
Add a new client in clients list

Figura 15: Framework server flow.

From the boot of the operating system on the embedded hardware, the framework is automatically initialized. From the command constants described in Section 5.2, the server verifies that it can connect to the hardware. If this connection can not be executed, the process is interrupted by indicating the error.

If the connection is successful, a socket server based on Transmission Control Protocol (TCP) (FOROUZAN, 2002) is initialized for connection to *n* different clients. From that moment the server goes into an infinite execution loop until the vehicle and the embedded hardware are completely off. In this execution loop, the server is waiting for new clients to connect. If any client software makes the connection, a link between the client and the hardware connection in the command base is created. The client is then added to a list of clients for execution on an asynchronous process to the server, this process is described in Section 5.4. After the server processes are fully initialized, if there are clients registered for auto boot, a process on the server automatically initiates these clients.

5.4 Clients

Clients are computer vision algorithms responsible for sending messages to the mission control and guidance of the UAVs. The basic composition of a framework client can be seen in Fig. 16.

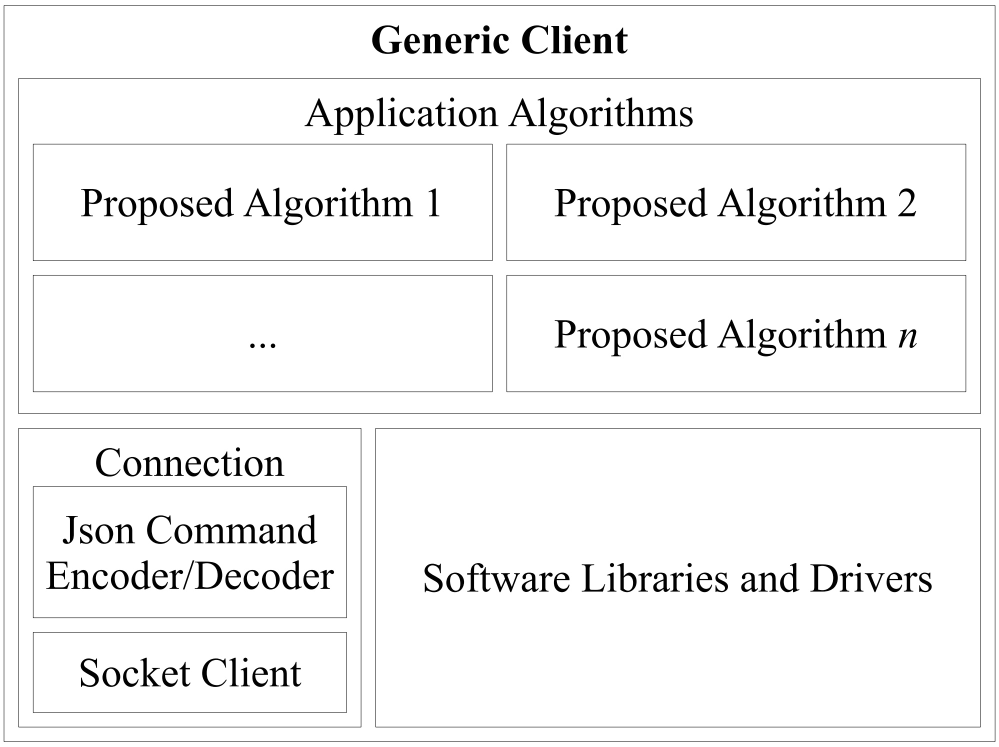


Figura 16: Generic framework client architecture.

The generic framework client architecture is divided into three main modules. The first module, at the bottom, on the left, is composed by the software of connection with the proposed framework. This connection software consists of a Socket client library along with an encoder and decoder of messages in the JSON format. From this connection module it is possible to exchange information with the proposed framework through lists of data encoded in textual format.

The second module consists of the software libraries and drivers used in the application. The last module is composed of one or more algorithms proposed for the development of the proposed client.

Each client connected to the server as described in Section 5.3. The client has its execution stream independent of the server execution. When a client connects to the server it creates a new instance of this client on the server and this instance receives the information for the connection to the hardware, after which its main execution stream initializes two other asynchronous processes that serve to receive commands and execute them, respectively. After the initialization of the asynchronous processes, the main execution flow of a client instance on server gets stuck in a connection verification loop. If the connection is lost or closed the client, all its processes are terminated without causing problems in the execution of the server. The complete flow of execution of a client instance on the server can be seen in Fig. 17.

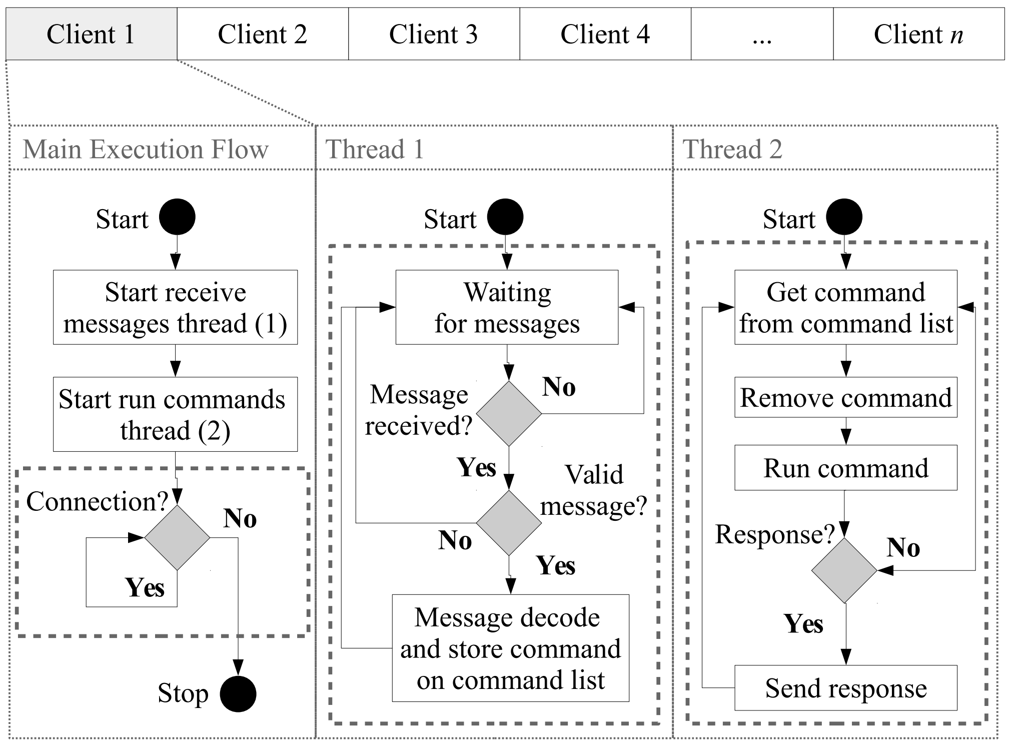


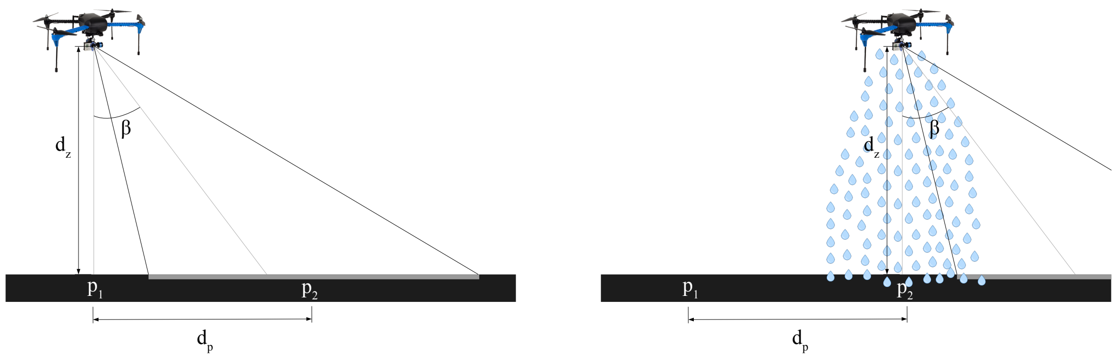
Figura 17: Framework client flow on server-side.

Parallel to this above explained execution, the thread responsible for receiving messages is waiting to receive new messages. When a message is received, its integrity is checked. If the message is valid, the JSON object is decoded as shown in Section 5.1. If the message is not valid, the process remains waiting for new commands. The command object is then stored in a list of commands and then returned to the process of receiving new messages.

Each client instance on server has its own list of commands. The list of commands works as a queue, the commands are executed on a first-come-first-served order, but in an asynchronous process separated from the thread that received the message. The execution thread gets and removes the first command from the list. The command is interpreted and the server searches for the method corresponding to the command in the command base, and then executes it. If it requires a response, the response is sent to the client. Otherwise the execution flow passes to the next command in the queue. A client's instance total execution flow ends when the connection is closed and all commands in the list are executed.

6 CASE STUDY IN PRECISION AGRICULTURE

This chapter describes the methodology used in the case study conducted in the domain of precision agriculture. The objective is to develop algorithms responsible for providing autonomy to a UAV used in applications involving spraying of agrochemicals. These algorithms must meet the temporal requirements that involve the proposed application, in order to integrate the mission control loop and guiding through the connection provided by the proposed framework. The application scenario presented in Fig. 18 is is described in detail in the following.



(a) Moment $t = 1$.　(b) Moment $t = 2$.

Figura 18: Detailed specification of the spray scenario using UAVs.

In this scenario, the UAV flies at a height represented by d_z meters above the ground, with the camera focused on an area ahead of the current UAV position. One of the dimensions of the area covered by the camera's field of view is represented by d_y. The objective is to perform the image acquisition when the UAV is located at the point p_1 (Fig. 18 a), perform the data processing and then execute the actuation processes when flying over the area associated to the previously captured image frame, that is, when its current position becomes at the point p_2 (Fig. 18 b).

For this process, the UAV needs to cover the distance between the two points represented by d_p. The average speed of the UAV is represented by v, given in meters per second. Considering for example the adjustment of the camera to a β angle such that the distance $d_p = 3$ meters and the average speed of the UAV in the proposed applications is $v = 3$, it is possible to conclude that the minimum desired speed (in terms of image acquisition) for each proposed algorithm is 1 FPS. This minimum processing rate is enough for the operation of the algorithm, but the higher the FPS rate reached by the proposed algorithms, the greater the gain in response time, which can leave the application with a flow of execution more fluid and still attending to the temporal requirements more efficiently. Moreover, a higher frame rate per second means a greater scalability the system can reach.

The first proposed algorithm is presented in Section 6.1 and is used to provide a method for performing punctual and self-regulated application of agrochemicals, which is the actuation illustrated in Fig. 18 b. The second algorithm is presented in Section 6.2 and it is used to perform the autonomous guidance of UAVs on precision agriculture plantations to follow the rows of plants, as in the example presented in Fig. 18 in which the UAV flies from point p_1 to point p_2 in a straight line. The algorithms must be efficient to meet the time requirements described above.

6.1 Proposed Embedded NDVI Client

The video captured by the infrared Raspberry Pi NoIR camera is composed of sequential frames or images which provide the sensation of movement. Each captured frame is composed of the channels corresponding to the blue and green visible light spectrum and near-infrared (NIR). The sensor captures the image obtaining the intensity of light for the respective spectrum channels in the same time t. The mathematical representation of the captured frame can be seen in (4).

$$f = \begin{bmatrix} f_{0,0} & f_{1,0} & \cdots & f_{w-1,0} \\ f_{0,1} & f_{1,1} & \cdots & f_{w-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{0,h-1} & f_{1,h-1} & \cdots & f_{w-1,h-1} \end{bmatrix} \tag{4}$$

The captured frame is represented by a matrix from maximum dimension W to the width and H to the height. Its origin falls in the upper left corner of the Cartesian plane. $f(x, y)$ is the function that retrieves the pixel in the x and y coordinate, and f is the amplitude that determines the amount of light present at that point of the digital image (GONZALEZ; WINTZ, 1977).

The pixel represented by (5), is the minimum entity that composes up an image. In the case of the infrared Raspberry Pi camera system, a pixel is represented by a vector R^3 where each position corresponds to the value of light intensity to the NIR channel, green channel and blue channel respectively. Each channel is represented by 8 bits, where the value ranges from 0 to the low intensity and 255 for higher light intensity for that point.

$$f(x, y) = \begin{bmatrix} NIR \\ G \\ B \end{bmatrix} \tag{5}$$

The NDVI algorithm, described in Section 3.2.1, operates on the image in the spatial domain, and has direct access to f. The Raspberry Pi infrared camera does not capture the channel of visible red, so, it is not possible to calculate (1). However, (WANG et al., 2007) demonstrated by the validation from two different data sets that GNDVI (Green NDVI) is a viable alternative to calculate the NDVI. The GNDVI is obtained by a mathematic operation of channels similar to the NDVI, with the difference of using the visible green channel instead of visible red channel, as shown in (6).

$$GNDVI = \frac{NIR - G}{NIR + G} \tag{6}$$

By applying (6) in each pixel of the digital image, it is possible to obtain the GNDVI value for each pixel. The average GNDVI for each processed frame can be obtained by (7).

$$GNDVI_{avg} = \frac{\sum_{x=0}^{W} \sum_{y=0}^{H} GNDVI(x, y)}{W \cdot H} \tag{7}$$

From the average GNDVI in (7) it is possible to estimate the application of pesticides and fertilizers. Although this calculation seems simple, the mathematical operations and algorithms need to be well organized and simplified to meet the time requirement that is the relationship between FPS rate and the speed of the UAV movement.

6.2 Proposed Crop Row Detection Client

This section describes the steps and operation of the proposed crop row detection algorithm. The steps that constitute the algorithm are: image acquisition; pretreatment; line detection; filtering the lines and line follower algorithm.

6.2.1 Image Acquisition

The video captured by the Raspberry Pi camera is composed of sequential frames or images which provide the sensation of movement. Each captured frame is composed of the channels corresponding to the red, green and blue visible light spectrum. The sensor captures the image obtaining the intensity of the light for the respective spectrum channels in the same time t. The mathematical representation of the captured frame can be seen in (8).

$$f = \begin{bmatrix} f_{0,0} & f_{1,0} & \cdots & f_{w-1,0} \\ f_{0,1} & f_{1,1} & \cdots & f_{w-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{0,h-1} & f_{1,h-1} & \cdots & f_{w-1,h-1} \end{bmatrix} \tag{8}$$

The captured frame is represented by a matrix from maximum dimension w to the width and h to the height. Its origin falls in the upper left corner of the Cartesian plane. $f(x, y)$ is the function that retrieves the pixel in the (x, y) coordinate, and f is the amplitude that determines the amount of light present at that point of the digital image (GONZALEZ; WINTZ, 1977).

The pixel represented by (9), is the minimum entity that composes up an image. In the case of the system of multispectral image, a pixel is represented by a vector R^3 where each position corresponds to the value of light intensity to the red, green and blue channels respectively. Each channel is represented by 8 bits, where the value ranges from 0, for the lowest intensity, and 255, for highest light intensity for that point.

$$f(x, y) = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{9}$$

The proposed algorithm operates on the image in the spatial domain and has direct access to f. Fig. 19 represents a frame acquired by the Raspberry Pi Camera which will be used as an example to describe all the steps of the proposed algorithm.

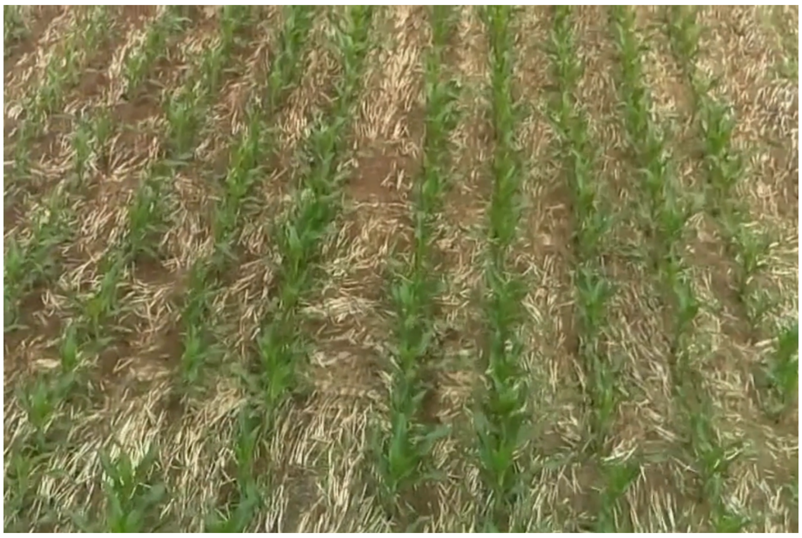


Figura 19: Frame acquired by the Raspberry Pi Camera.

6.2.2 Pretreatment of the Image

The pretreatment performed by the proposed algorithm consists of four steps necessary to remove the noise and prepare the image for the step of identifying the lines. First, the color image represented by Fig. 19 is transformed to gray scale, using the $2G - R - B$ transform described in Section 3.2.2. The Fig. 20 shows result image obtained with the application of (2) on Fig. 19.

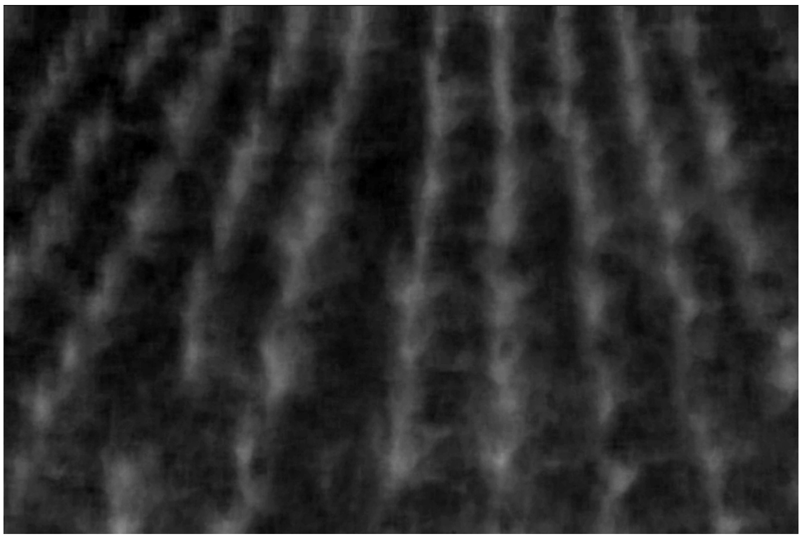


Figura 20: Gray scale image.

However, the result is unsatisfactory because much of the noise still remains in the image. Then, to solve this problem applies the second stage of the pretreatment. This step consists in the application of a threshold filter represented by (10).

$$f(x,y) = \begin{cases} 255 & \text{if } T_{min} \le G(x,y) \le T_{max} \\ 0 & \text{if } Others \end{cases} \tag{10}$$

In this step, the image is binarized respecting the following rule: when the gray value is between the range defined for $T_{min} = 60$ and $T_{max} = 255$ thresholds the pixel receives the value 255 and when it is out of the thresholds it receives 0. The threshold levels should be calibrated according to the cultures and settings of the camera used. The binarized image is demonstrated by Fig. 21.

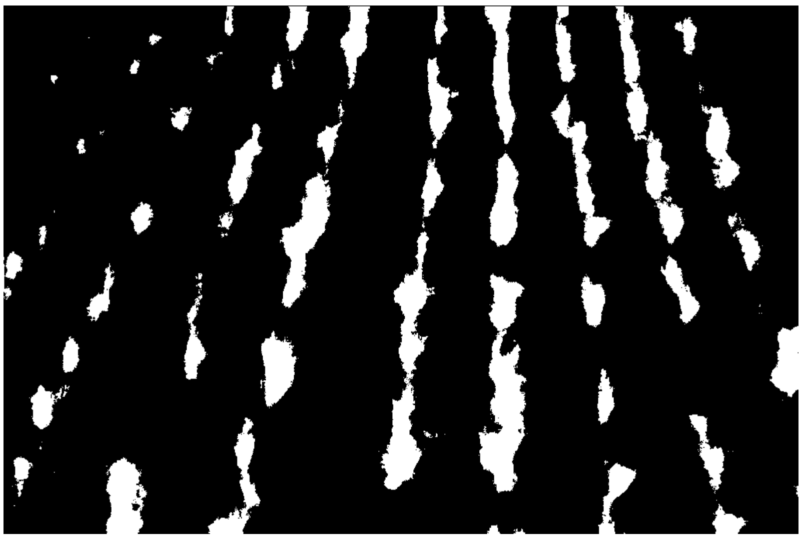


Figura 21: The binarized image.

As the binarized image has a lot of information irrelevant to the processing step, a good practice would be to apply a edge detection filter, as for example Sobel (SOBEL; FELDMAN, 1968). The big problem is that in these types of filters there may be loss of information because sometimes the edges are not preserved or even edges are represented by more than one pixel in thickness and can cause problems in the stage of the lines detection.

The Fig. 22 demonstrates the proposed horizontal filter. This filter has the purpose of traversing the image horizontally highlighting only vertical lines. The first step of the filter is to traverse the binarized image horizontally by identifying and marking the edge transitions of the objects that can be observed in Fig. 22a. Each object is composed of a cluster of pixels connected to its horizontal neighbors.

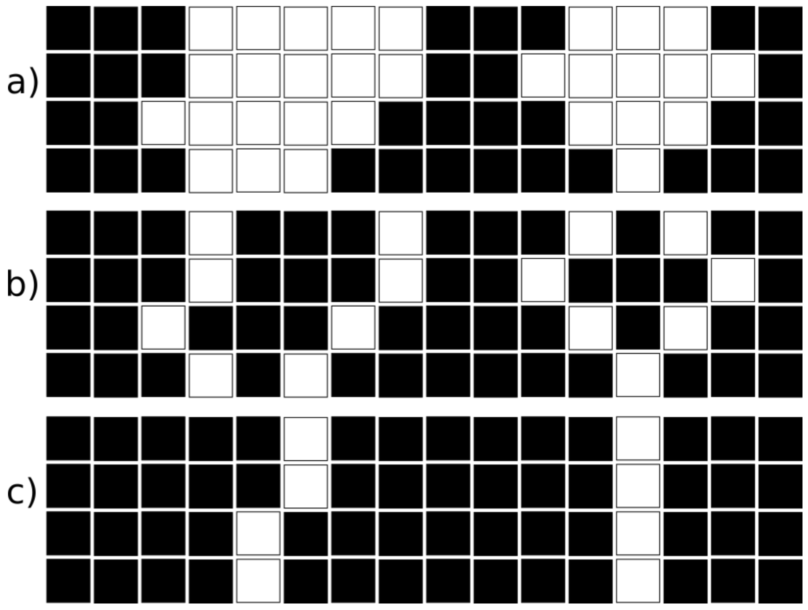


Figura 22: Proposed filter for edge detection. a) A cluster of pixels representing an object; b) Identification of edge transitions; c) Average between the right and left edges of each object.

For the identification of the objects in the step of Fig. 22b, the algorithm cross the image represented by Fig. 22, line by line, from left to right. When a transition point between 0 to 255 is identified the x_1 position is stored to that point. When the 255 to 0 transition occurs, the object is terminated, and the x_2 position is stored for that point. This process is repeated for all y. The positions x_1 and x_2 represent the edges of an object. The result of this step can be seen in Fig. 23.

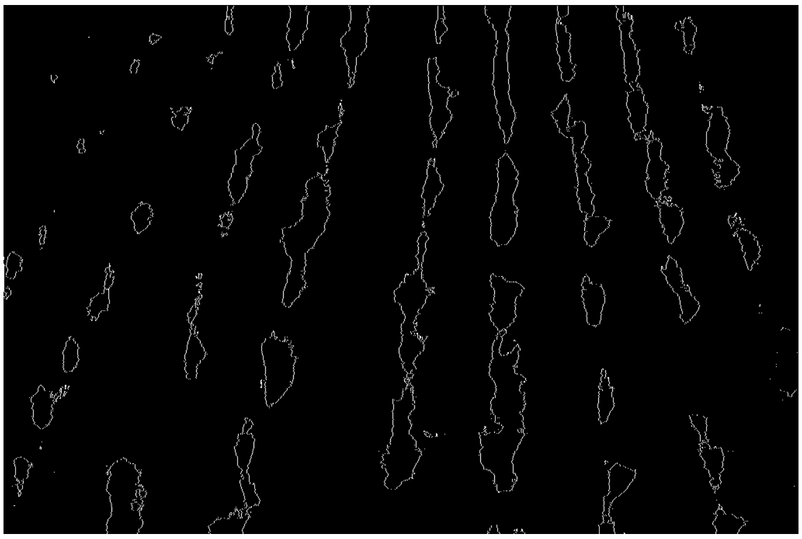


Figura 23: Result of application of the proposed edge detection filter on Fig. 21.

In this image all the transition pixels are represented by 255 and those that are not part of the transition are represented by 0. This step is efficient for the extraction of the edges of the objects of the image, because it ensures that the edges are represented exactly by one pixel of thickness and avoid the occurrence of loss of information relevant to the processing step.

The last step that composes the pretreatment of the image is represented in Fig. 22c. In this step x_c is calculated for all edges identified in the previous step using (11).

$$x_c = \frac{x_1 + x_2}{2} \tag{11}$$

For all pixels placed in the new calculated coordinates $f(x_c, y)$ is assigned the value 255. These pixels represent the centroids of each object, which in turn composes the crop lines. Fig. 24 represents the final result of the application of the proposed filter.

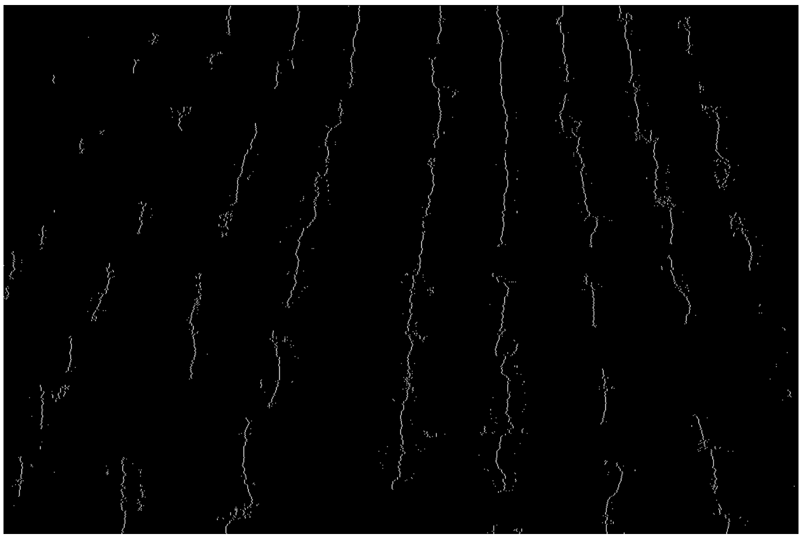


Figura 24: The result of pretreatment process.

As can be seen in Fig. 24 only relevant information for the crop row detection is present in the image to be processed. The next step consists of the processing step and consequently identification of the crop rows.

6.2.3 Line Detection

The identification of crop rows is done using the Hough Transform described in the Section 3.2.3. The generalized Hough transform is currently used as the main alternative for the crop row detection algorithms (JI; QI, 2011; JIANG; ZHAO; SI, 2010). Applying the it transform in Fig. 24 resulting from the pretreatment process, the identified planting lines are obtained. Fig. 25 represents the identified lines plotted in the original image (Fig. 19) for comparative purposes.

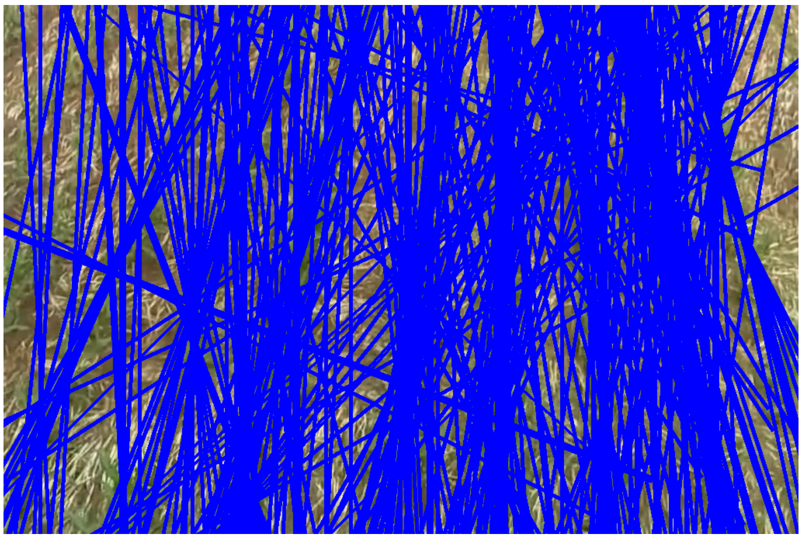


Figura 25: Result of Hough Transform.

In this step of the algorithm, it is observed that the number of lines identified by the transform is relatively large, presenting a number of false positives.

To filter these results, analytic geometry techniques described in the Section 6.2.4 are used.

6.2.4 Filtering the Lines

The Hough Transform returns several false positive lines that make it difficult to identify the actual lines corresponding to the crop rows. Thus, it becomes necessary a process of filtering and selection of results for extraction of relevant data. The first problem to be solved is that the Hough Transform returns points in the spatial domain that are allocated in coordinates outside the plane of the image. To solve this problem an array of generic straight-line points represented by (12) is defined. In this array, the points (x_1, y_1) and (x_2, y_2) are any two points of the line, points returned by the transform. The coordinates (x_q, y_q) represent a general point of the line to be discovered.

$$A = \begin{bmatrix} x_q & y_q & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} \tag{12}$$

Calculating $det(A) = 0$ and isolating x_q gives the generic equation (13) which can be used to calculate points belonging to a line.

$$x_q = \frac{x_2 y_q + x_1 y_2 - x_2 y_1 - x_1 y_q}{y_2 - y_1} \tag{13}$$

Applying (13) to the coordinates $y_q = 1$ and $y_q = h - 1$ it is possible to obtain the new straight points composed of the coordinates (x_q, y_q), points that are allocated within the image plane.

Observing Fig. 25, it is possible to see that the identified lines are arranged in the image to form an angle of $35°$ to $65°$ with the left and right sides of the image. In order to separate and filter these lines of the identified ones, the average of all the groups of lines that intersect within the plane of the image is calculated. For this operation, the equation of the generic line defined in (14) is used.

$$ax + by + c = 0 \tag{14}$$

Then, for each identified line, the coefficients a, b and c are calculated using (15) and the points composed of the coordinates (x_1, y_1) and (x_2, y_2), obtaining the generic equation of each line.

$$\begin{cases} a = y_1 - y_2 \\ b = x_2 - x_1 \\ c = x_1 y_2 - x_2 y_1 \end{cases} \tag{15}$$

The next step is to equalize all equations for all possible line combinations. Solving the system in (16) it is possible to find the coordinates of the point of intersection p.

$$p = \begin{cases} a_1x + b_1y + c_1 = 0 \\ a_2x + b_2y + c_2 = 0 \end{cases} \tag{16}$$

When $p(x, y)$ belongs to the plane of the image, the average of the two straight lines is discarded, saving the new coordinates calculated for the new line. The new calculated lines are compared again until there is no p allocated within f.

This process ensures the identification of the lines that constitute the crop rows. However, in some cases it may occur that one crop line is represented by two parallel lines. This problem is solved by calculating the angular coefficient for each line, following (17).

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{17}$$

Then d_{max} is defined as the smallest distance between parallel lines. In the case of this work it used $d_{max} = 20px$. The r_1 and r_2 are parallel lines, r_1 is represented by the points (x_1, y_1) and (x_2, y_2), and r_2 is represented by the points (x_3, y_3) and (x_4, y_4) and their distance is calculated by (18).

$$d = \begin{cases} x_3 - x_1 & \text{se } x_1 < x_3 \\ x_1 - x_3 & \text{se } x_1 >= x_3 \end{cases} \tag{18}$$

Then, m for all combinations of possible straight lines are compared, as well as the average of all straight lines having the same m and having d less or equal than d_{max}. The final result of this step of the lines identification be seen in Fig. 26.

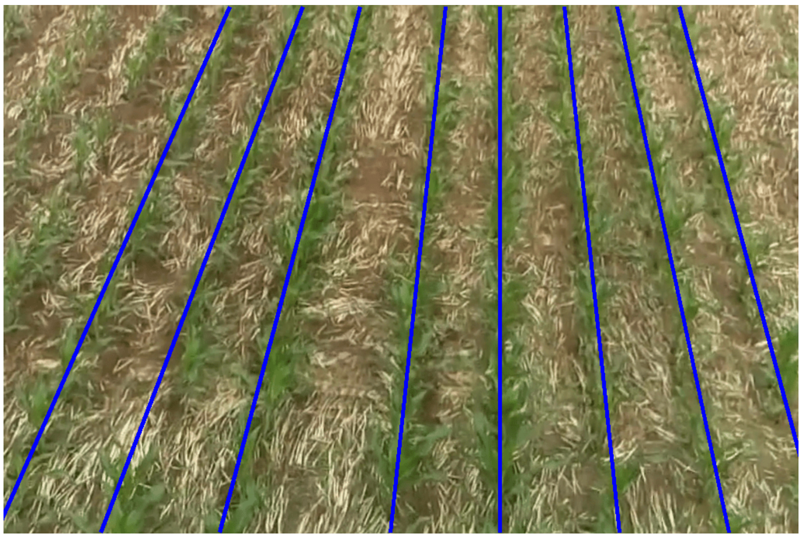


Figura 26: Final step of identifying crop lines.

6.2.5 Line Follower Algorithm

The line follower algorithm selects one of the lines identified by the detection algorithm to conduct the UAV. The operation of the algorithm consists of calculating the pixel distance from the base line to the nearest identified line and then performing the conversion to the distance in meters that must be compensated at the UAV position during the flight, in order to keep it in its trajectory. The base line to be used in this work is represented by the line R in Fig. 27 composed of the points $A(\frac{w}{2}, 0)$ and $B(\frac{w}{2}, h-1)$ which represents the horizontal center of the image.

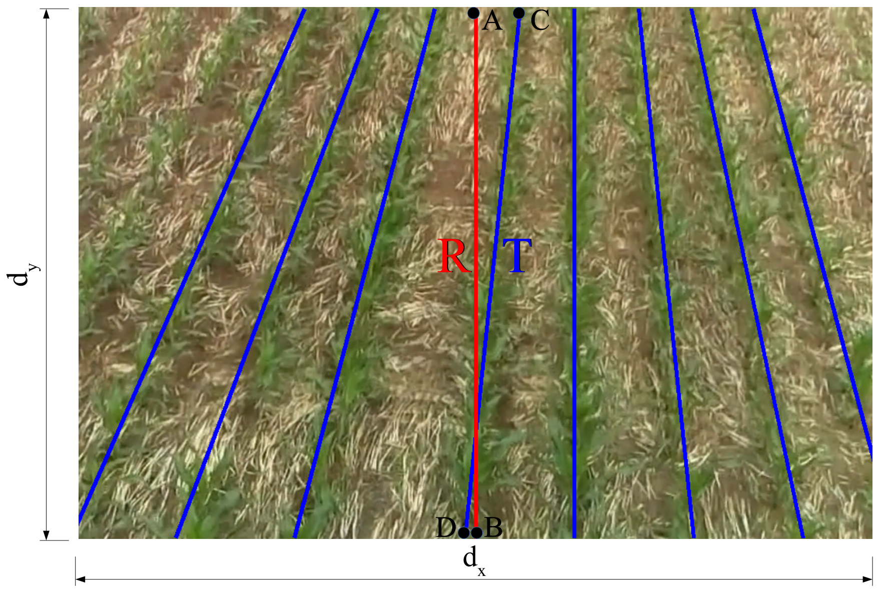


Figura 27: Lines of reference on the image.

The first step in the execution of the algorithm is the selection of one of the lines identified to be used as reference in conducting the UAV. The lowest distance in pixels in relation to the base line R was assumed as the criterion for the selection of the reference line T (Fig. 27). Where T is composed of the points C and D. This criterion is applied for the initialization of the algorithm and also for when R does not belong to the focal plane of the camera. The next step of the algorithm is to determine the distances d_x and d_y on a metric scale, corresponding to the measures w and h which represent the vertical and horizontal axis of the image respectively in pixels. Fig. 28 demonstrates the disposition of the distances d_x and d_y in the three-dimensional space, through the representation of the execution scenario of the algorithm at the exact moment of the image is captured by the camera.

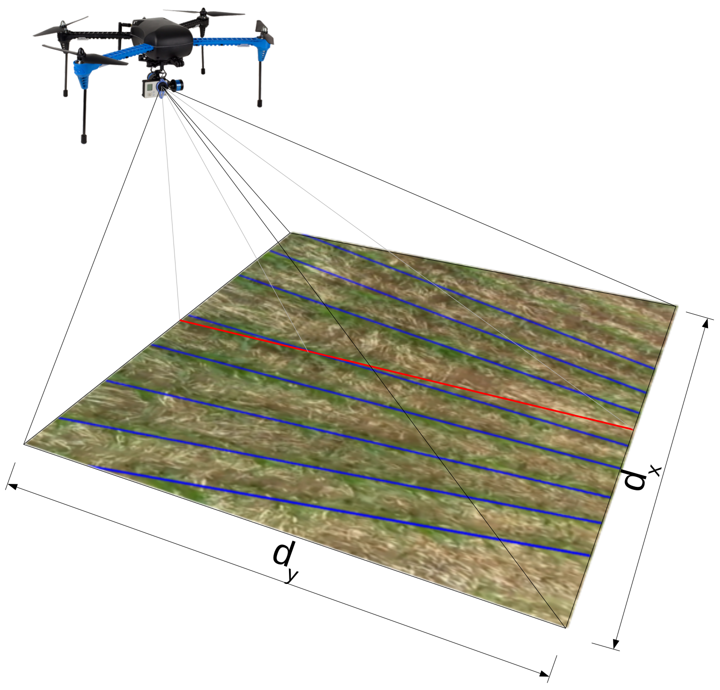


Figura 28: Camera focal plane.

The field of view of the camera in relation to the ground consists of a triangle with an obtuse angle allocated in the range of $90°$ to $180°$, which can be seen in Fig. 29. In this generalizable scenario, it has as parameters the focal aperture angle of the camera, represented by α, the angle of inclination of the camera in relation to the ground, represented by β, and the altitude of the UAV in relation to the ground, represented by d_z. These parameters are configured beforehand or detected during the execution time of the algorithm.

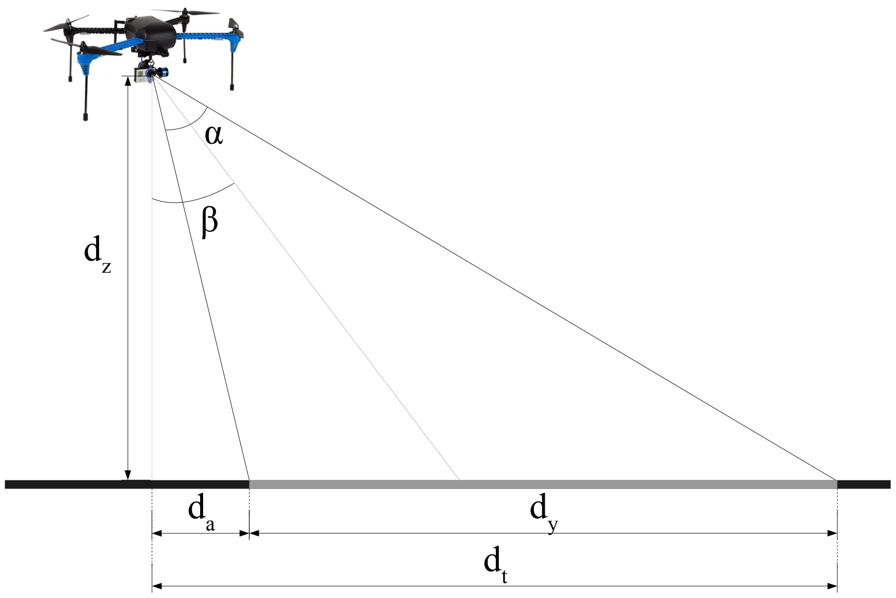


Figura 29: Side view of the focal plane of camera.

To calculate the value of d_y which is the opposite side the camera aperture angle, first the d_t is calculated, using (19), which is represented in Fig. 29, i.e. the total distance from the corresponding point on the ground to the location of the UAV and the end point where the line corresponding to the camera's viewing angle intersects the ground.

$$d_t = tan(\beta + \frac{\alpha}{2}) \cdot d_z \tag{19}$$

With the total distance calculated, d_a is calculated using (20), which represents the distance between the corresponding point on the ground to the location of the UAV and the starting point, corresponding to the point of the image captured closer to the UAV.

$$d_a = tan(\beta - \frac{\alpha}{2}) \cdot d_z \tag{20}$$

Then, knowing the measures of the two adjacent sides to the right angle formed by the axis that defines the position of the UAV relative to the ground, (21) is used to calculate the value of d_y in meters.

$$d_y = d_t - d_a \tag{21}$$

Knowing the measure d_y, (22) is used to calculate p_x which quantitatively represents in meters the equivalence of one pixel in the captured image.

$$p_x = \frac{d_y}{h} \tag{22}$$

The next step is to calculate d_x that corresponds in meters the distance to the horizontal axis of the image using (23).

$$d_x = p_x \cdot w \tag{23}$$

With the distances proportional to the area of the image calculated in meters, the gain vector G^3 is calculated using (24).

$$G = \begin{bmatrix} (\frac{A_x+B_x}{2} - \frac{C_x+D_x}{2}) \cdot p_x \\ \frac{d_y}{2} \\ d_z \end{bmatrix} \tag{24}$$

Where G is the final result of the algorithm, consisting of the gains to be added to the three-dimensional position in which the UAV is in the previous instant, adjusting the position of the UAV over the T line. The angular coefficient of the line T can be used to control the rotation of the UAV around the z axis of the three-dimensional coordinate system, but this step is performed automatically by the flight controller used in this work.

7 IMPLEMENTATION DETAILS

7.1 Hardware Architecture

The hardware architecture is comprised of an embedded hardware integrated to the UAV. The UAV used in this project is a 3DR Iris+ quadrotor (CANO et al., 2017). For the protection of embedded hardware during the experiments flights, a protective enclosure was built. Fig. 30 shows a picture of the hardware mounted on the 3DR Iris+ quadrotor.

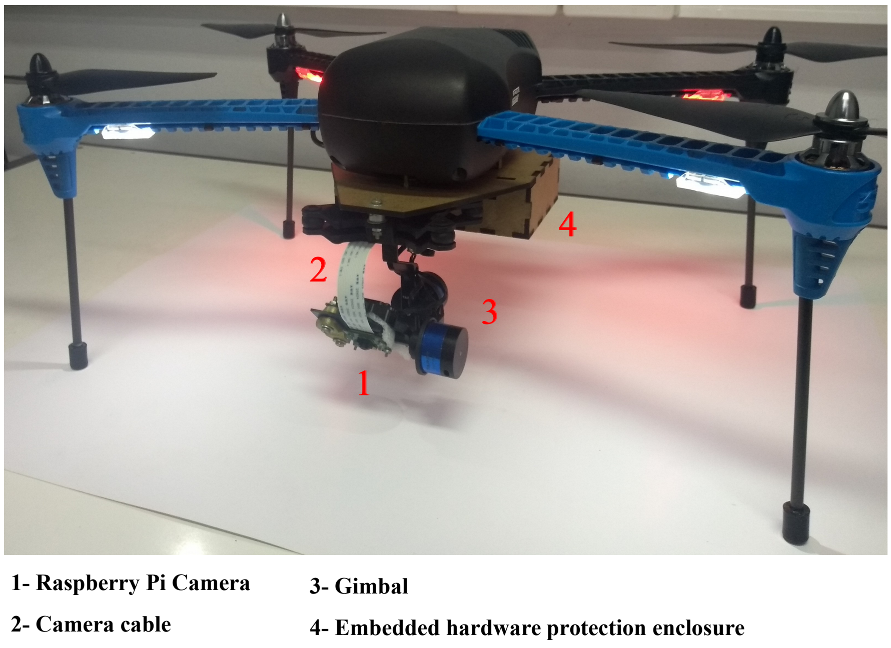


Figura 30: Final assembled hardware.

In this project two different hardwares were used to acquire images. In the case study presented in Section 6.1 which proposes the punctual and self-regulated application of agrochemicals, the Raspberry Pi NOIR V1 Rev 1.3 camera (Fig. 31) was used, that has the same features of a common RGB camera, but does not have the infrared filter that allows capturing this light spectrum channel. For correct acquisition of the infrared images using the Raspberry Pi NoIR camera, the blue gel filter (UPTON, 2013) (which official name is Roscolux #2007 Storaro Blue) was fixed in front of the infrared camera (Fig. 31).

The blue gel filter in conjunction with Pi Noir allows monitoring the health of green plants (UPTON, 2013), using indexes such as NDVI, which is used in this work. The case study presented in Section 6.2 where it is proposed to autonomously guidance the UAV on the plantation use the Raspberry Pi V1 Rev 1.3 RGB camera (Fig. 31) for the video acquisition. To control the camera position, it was attached to the gimbal Tarot T-2D V2. The camera cable was replaced with a long cable that allows the correct operation of the system with the gimbal (Fig. 31).

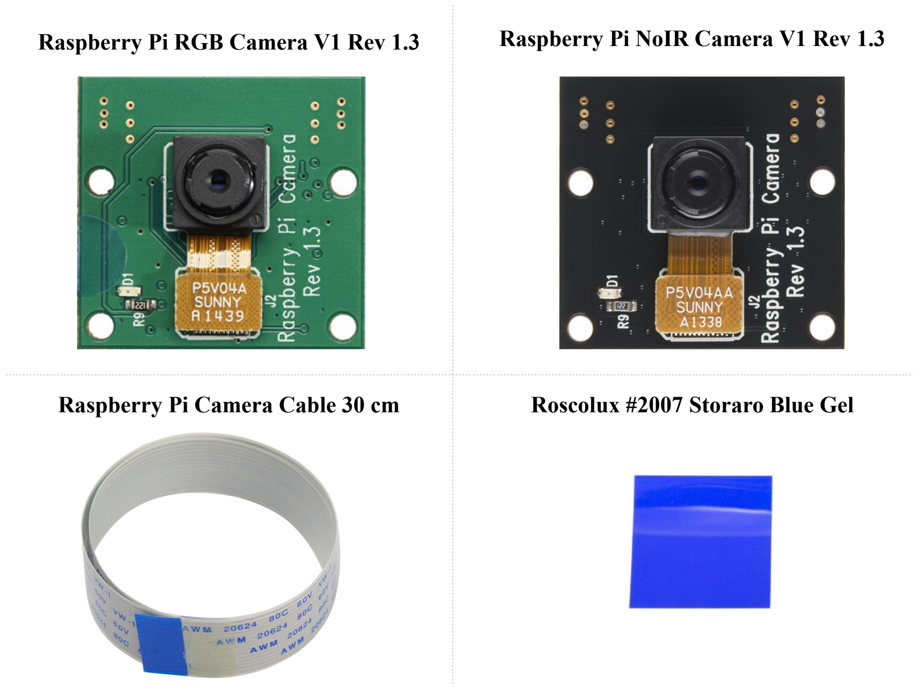

Raspberry Pi RGB Camera V1 Rev 1.3
Raspberry Pi NoIR Camera V1 Rev 1.3
Raspberry Pi Camera Cable 30 cm
Roscolux #2007 Storaro Blue Gel

Figura 31: Camera hardware components.

The Raspberry Pi 3 model B was used as embedded computer, that has a 1.2GHz 64-bit quad-core ARMv8 processor, 1GB of RAM and GPU VideoCore IV 3D graphics core (RICHARDSON; WALLACE, 2012). A SD card class 10 with 32GB is used to store the operating system and the software that implements the developed algorithms. For the integration of all hardware and energy source to the gimbal and the Raspberry Pi, a DC-DC converter circuit was used (MHATRE et al., 2015). A real-time clock (RTC clock) module was coupled to Raspberry Pi to prevent overwriting of recorded log files. The Fig. 32 presents the embedded hardware connections.

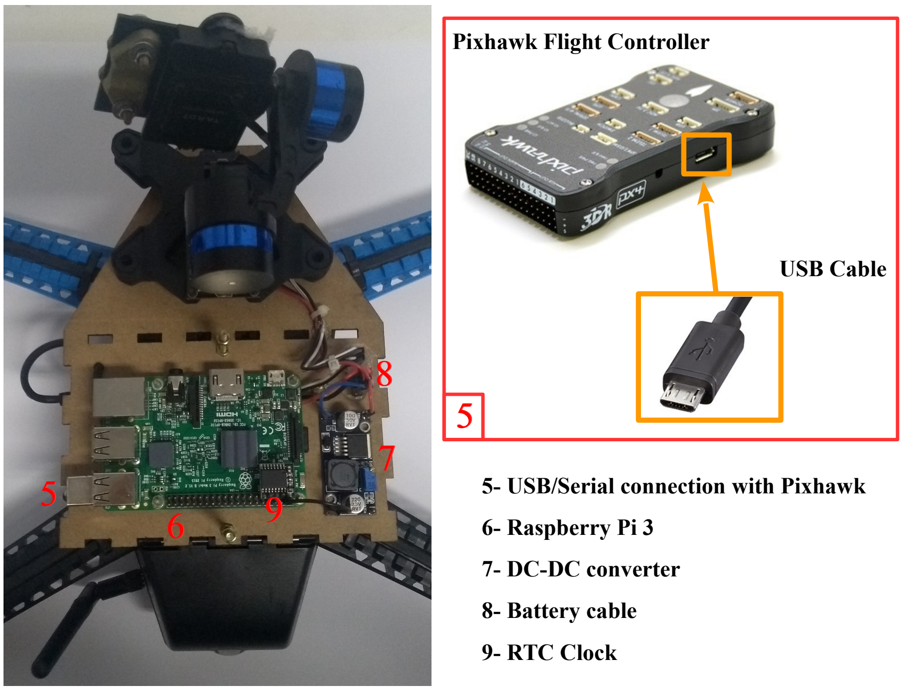


Figura 32: Embedded hardware connections.

The flight controller hardware is responsible for receiving the commands from the proposed framework and then trying to execute them. In this work the flight controller used was the Pixhawk Autopilot (MEIER, 2011). A serial connection with the Pixhawk was configured using the USB cable installed to the Raspberry Pi. The integrated hardware has been tested and configured prior to software installation and testing.

7.2 Software Architecture

The proposed software architecture is divided into four layers, as shown in Fig. 33. The first module, at the bottom of the figure, is the operating system, Raspbian Stretch with GUI interface disabled for better performance. This operating system has been chosen due to the compatibility with the software libraries required for the developed system.

The second layer of software architecture is comprised of software libraries and drivers. The main software libraries used are: the library Socket, used for the connection between the framework and the clients of computer vision; the pySerial library used to perform the serial connection to the vehicle hardware; and the MAVLink, which allows the exchange of messages to the flight controller of the UAVs over an established serial connection.

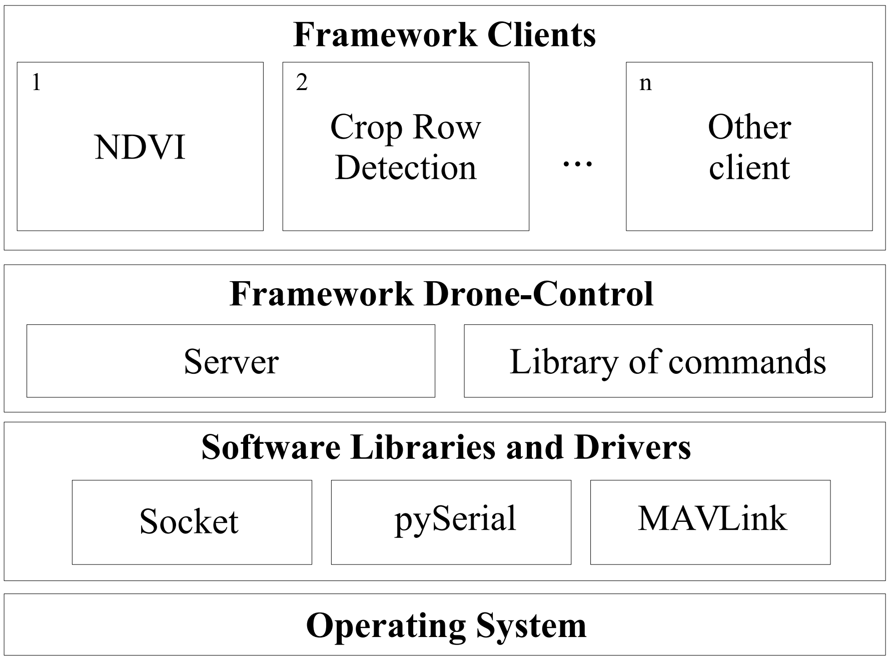


Figura 33: General Software Architecture.

The third layer of software is composed of the proposed Framework, whose structure and functioning are described in Chapter 5. In short, the framework is composed of a server connected to the hardware of a UAV and a library of commands, which the hardware can execute, these commands are previously implemented. The framework is configured for startup along with the startup of the operating system, which in turn is initialized at the time the vehicle is powered on.

Finally, in the last layer are computer vision algorithms, these are called clients and can connect and exchange messages with the hardware through a connection with the framework. These algorithms may be located on the same embedded hardware that the framework is installed, or on a different hardware. The framework has a feature to auto boot these algorithms, if they are installed in the same hardware of the framework. The architecture of the client algorithms developed in the proposed case studies can be observed in the Sections 7.2.1 and 7.2.2.

7.2.1 NDVI Client

The NDVI client software architecture is divided into three main modules as shown in Fig. 34. The first module, at the bottom, on the left, is composed by the software of connection with the proposed framework. This connection software consists of a Socket client library along with an encoder and decoder of messages in the JSON format. From this connection module it is possible to exchange information with the proposed framework through lists of data encoded in textual format.

The second module consists of the software libraries and drivers used in the application. The first library is the Raspicam, which allows the interface between the camera hardware and the developed algorithms. The second library is the Open Source Computer Vision Library (OpenCV) (KAEHLER; BRADSKI, 2016), used in the manipulation of images. The last module is composed of the proposed NDVI algorithm.

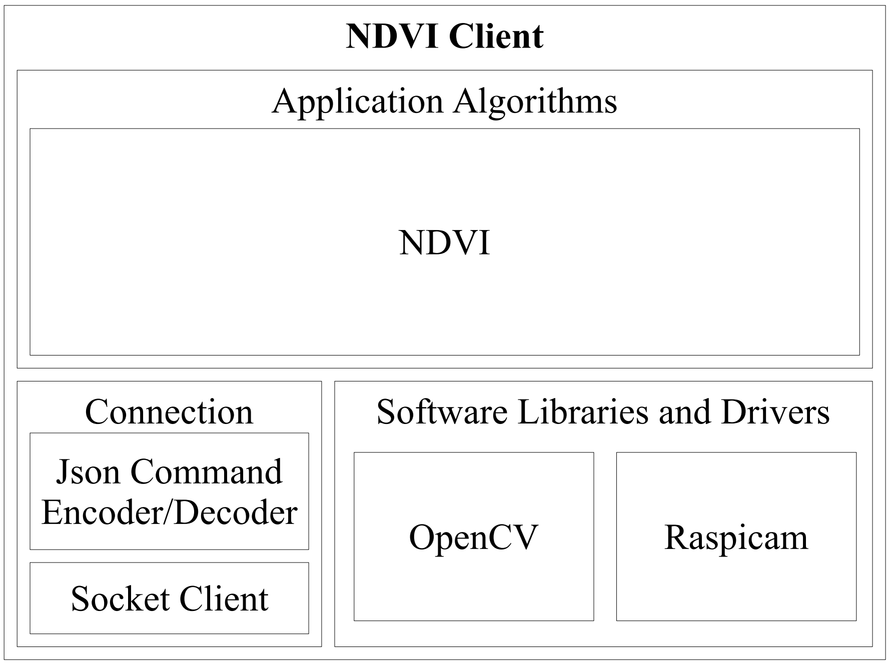


Figura 34: Proposed NDVI software architecture.

For obtaining an algorithm with efficient performance, two different implementations were developed to test the proposed algorithm efficiency in two different programming languages. The first implementation was done in Python and the second implementation was developed in C++ language. To carry out the implementation in both languages it was considered good coding practices for the performances purposes, avoiding "for"structures and avoiding excessive parameters' passing. The use of OpenCV library functions for mathematical operations on matrices was chosen to avoid the "for"structures. Also, OpenCV library was configured to use the GPU hardware via components of Open Computing Language (OpenCL) (MUNSHI, 2009).

The activity flow executed by the algorithm implemented in both languages is shown in Fig. 35. The execution of the algorithm is automatically initialized by the proposed framework, when it is initialized.

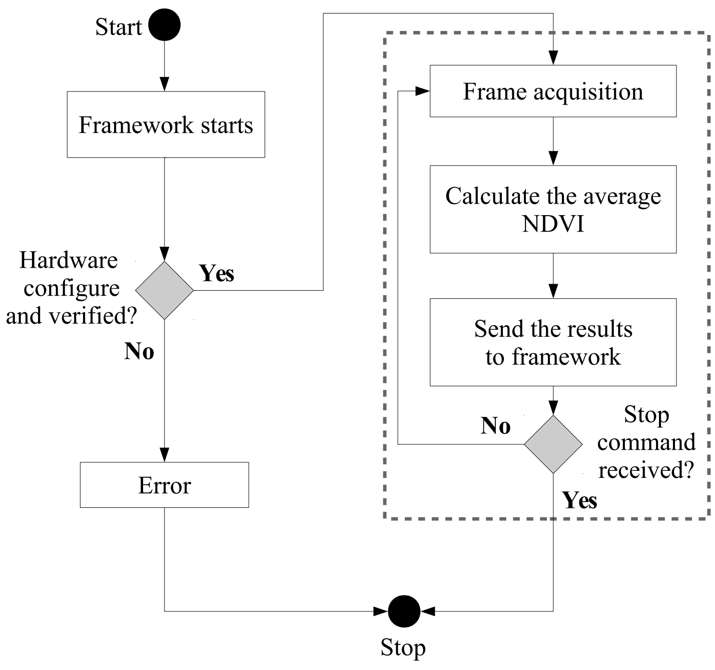

Start
Framework starts
Hardware configure and verified?
Yes
No
Error
Frame acquisition
Calculate the average NDVI
Send the results to framework
No
Stop command received?
Yes
Stop

Figura 35: Proposed NDVI Algorithm flow.

Once initialized, the algorithm defines the appropriate settings in the Raspberry hardware and checks if it is functional, otherwise the algorithm terminates, indicating an error. If everything is functional, the video frames begin to be captured following the predefined parameters. NDVI is calculated for each frame using (7). Then the results are sent to the framework where a command receives the data and writes it to a log file for further analysis of the results. After this procedure, the algorithm verifies if the stop command was sent by the framework, otherwise, it stay in the execution loop. If the stop command is received, the algorithm terminates. The stop command is sent at the moment the UAV back to land.

7.2.2 Crop Row Detection Client

The crop row detection software client architecture is divided into three main modules as shown in Fig. 36. The first module, at the bottom, on the left, is composed by the software of connection with the proposed framework. This connection software consists of a Socket client library along with an encoder and decoder of messages in the JSON format. From this connection module it is possible to exchange information with the proposed framework through lists of data encoded in textual format.

The second module consists of the software libraries and drivers used in the application. The first library is the Raspicam, which allows the interface between the camera hardware and the developed algorithms. The second library is OpenCV, that proves the necessary functions for the implementation of the algorithms. The third and last module is composed by the proposed algorithms, implemented in C++ language for better performance.

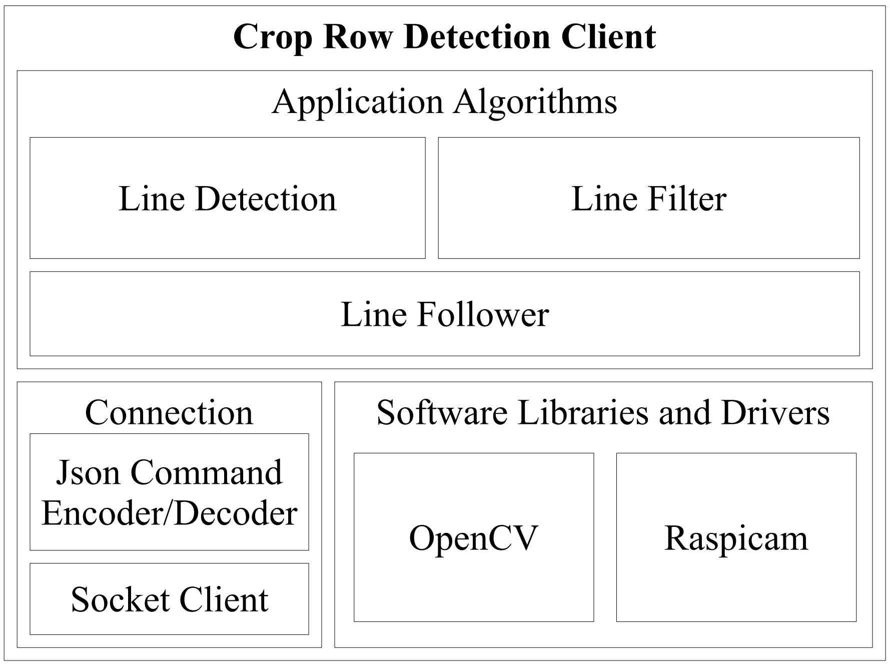


Figura 36: Proposed crop row detection software architecture.

The flow of activities performed by the implemented system is shown activity diagram depicted in Fig. 37. The execution of the algorithm is automatically initialized by the proposed framework, when it is initialized.

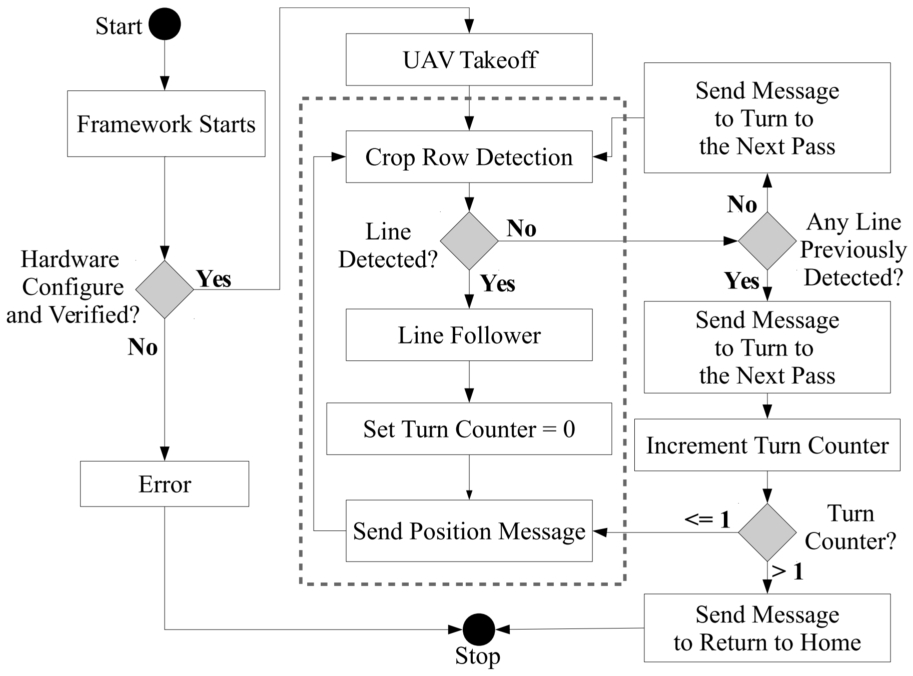


Figura 37: Proposed Crop Row Detection operation flow.

After starting, the software sets the appropriate settings on the Raspberry Pi camera hardware and it verifies that the camera is functional, otherwise the software terminates, indicating an error. If the hardware is properly configured, commands are sent to the framework for the takeoff of the UAV. After taking off the vehicle, its height is adjusted to 2 m from the ground. The Crop Row Detection Algorithm is then initialized by following all the steps described in Section 6.2. The UAV mission path is shown in the Fig. 38. If the lines are not detected, the system starts to send position messages to framework, to moving the UAV forward until the first lines are detected (Fig. 38 dashed line f). When lines have been detected for the first time, this information is passed to the line follower algorithm described in Section 6.2.5 which will in its turn determine the parameters for driving the UAV on the identified lines (Fig. 38 lines indicated by L).

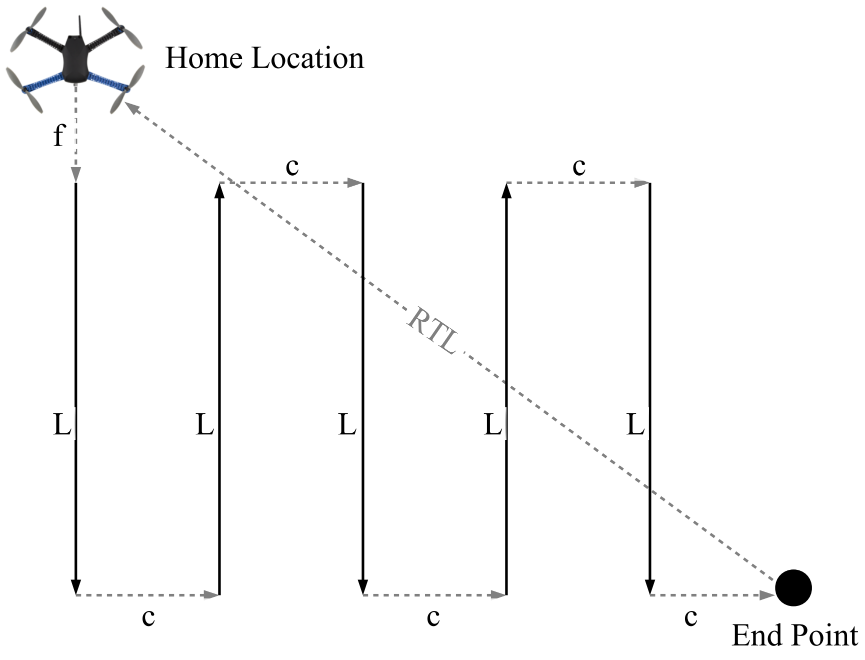


Figura 38: Mission path covered in the execution of the Crop Row Detection algorithm.

When the UAV reaches the end of a planting line, no lines will be detected, this will indicate that the UAV should return to the plantation area in the subsequent lines. To perform this operation, it moves along the path indicated by line c in Fig. 38 and turning its heading to the plantation area. After returning to the plantation area, the UAV will identify the next line and repeat the process until the end of the extension of the plantation area. Whenever the motion indicated by the line c in Fig. 38 is executed, a counter is incremented and if new lines are identified, this counter is decremented. If new lines are not identified this movement is repeated and the counter is incremented again. If the counter value is greater than 1 this will indicate the end of the plantation area, in this case a message indicating to the UAV to go back to the starting point is sent. The return of the UAV to the starting point is demonstrated in line RTL on Fig. 38. The entire process of conducting the UAV is done by proposed framework, that sent the mission control parameters to the UAV's control flight unit. The proposed crop row detection software is responsible for generating the flight runtime orientation parameters that are sent via framework. This mission setup illustrated in 38 was used to perform the tests with the algorithms that are described in Section 8.2.

8 EXPERIMENTS AND RESULTS

The experiments performed with the proposed framework occurred in two different stages. In the first stage, the laboratory tests were performed in order to verify and prove the feasibility of using the framework in the autonomous applications proposed in the case studies. For this, a test client was developed that connects to the framework and sends a series of random commands that compose a test mission. The entire test client simulates the execution of multiple clients through the use of multiple threads. The number of messages generated by each test client of the framework was fixed in 33 messages. All threads were simultaneously created and initialized in order to simulate the execution of the system in an actual application. The number of threads was specified in a range of 1 to 12, each thread being the representation of a client connected directly to the framework.

In each simulated client, before sending the command to the framework, the current time with high precision was recorded inside the message and then it was encoded and sent to the framework. Upon receiving the message the framework interpreted the message generating a command, which when executed, retrieved the time recorded in the message and collected the time interval again. Both temporal measures such as other message identification data were recorded in logs as test results. A total of 12 sets of tests were performed.

The results with the log data obtained in the tests is shown in Fig. 39. In the performed tests, the average time for executing a command is 237.82 ms. This measure comprises from the time the message is prepared for sending on the client side until the time it is decoded and executed in command format by the framework side. Analyzing the results, it can be considered that the framework achieved the desired performance to manage the application commands, this is justified by the fact that the processes executed in its internal flow synchronously but simultaneously, that is, at the same time that a command is being decoded, commands are being received and executed all at the same time.

Another important point to be analyzed is that the average time for processing and executing a command by the proposed framework varies for each mission configuration and number of client algorithms required to execute it. This also depends on how the processes are allocated to run on embedded hardware, having direct connection to the behavior of the process scheduler

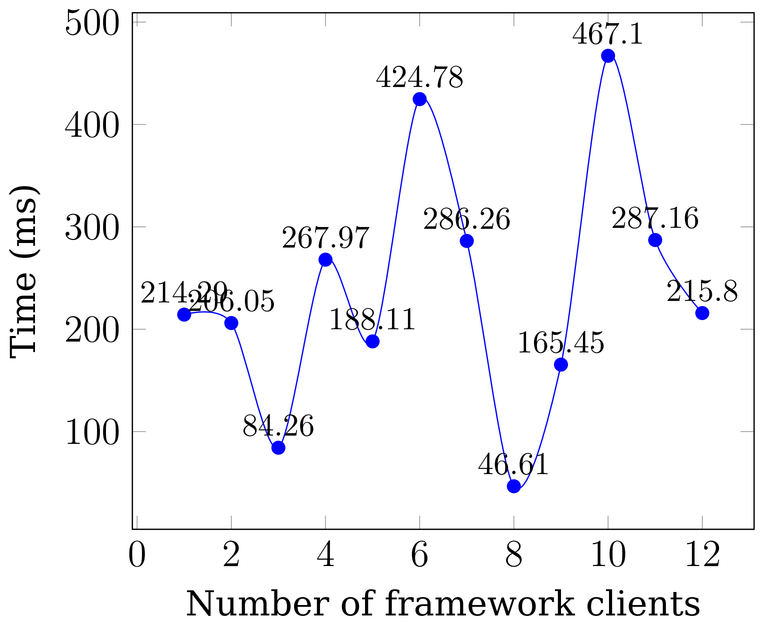
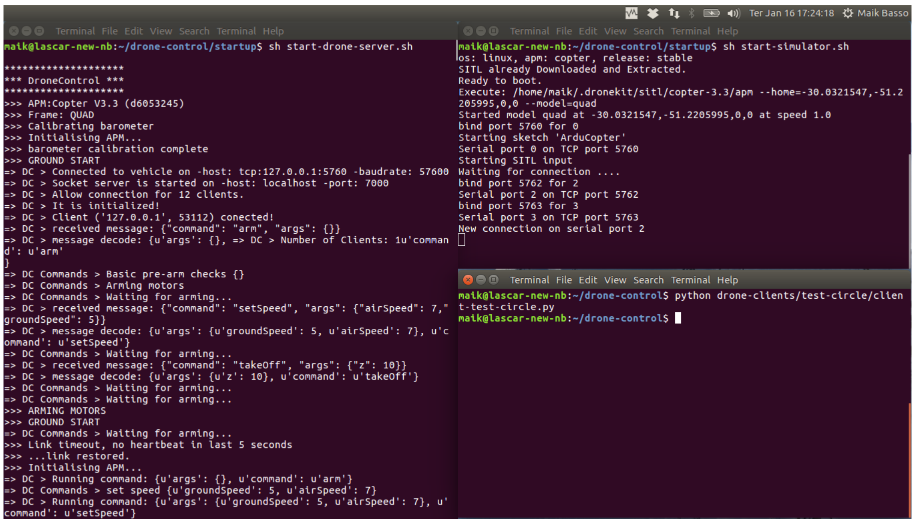


Figura 39: Average time spent executing a command with different numbers of competing clients in the framework.

of the operating system used. During the tests it was possible to verify that in an application running with eight clients the performance is superior to applications running with fewer connected clients.

With the performance of the framework proven on embedded hardware, tests were performed to ascertain the behavior of the framework by treating possible errors that may occur in the execution process. For this, a notebook and software simulator called Dronekit Sitl were used (3DROBOTICS, 2015). The software used is provided by the manufacturer of the flight controller and simulates a vehicle by running the same firmware on the flight controller hardware. In addition to the simulator, the software QGroundControl (MEIER et al., 2010) was used to visualize and monitor the movement of the simulated vehicle. QGroundControl is a flight controller software to be used in base control stations. Fig. 40 demonstrates the log execution of a mission in which the purpose was to perform a flight in circular format. In all simulated tests performed the vehicle behaved in an expected manner by responding to the commands sent and completing the missions successfully. In Fig. 40a the proposed framework log, the simulator log, and the test mission client execution terminal are demonstrated. In the screen represented by Fig. 40b the path of the test mission performed is demonstrated.

In this step, it was also evaluated the operation of the handling of exceptions to the possible problems that can cause in the stop of operation of the proposed framework. Among these problems were tests such as loss of connection of clients, messages with corrupted content, missing commands and loss of connection with the vehicle. In none of the tests performed, the framework had its flow of execution interrupted, dealing with errors efficiently.



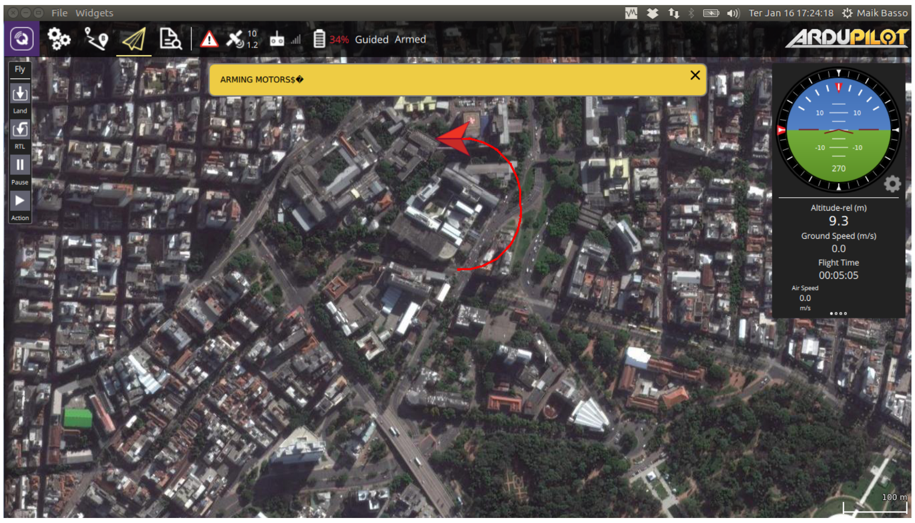
```
maik@lascar-new-nb:~/drone-control/startup$ sh start-drone-server.sh

********************
*** DroneControl ***
********************
>>> APM:Copter V3.3 (d6053245)
>>> Frame: QUAD
>>> Calibrating barometer
>>> Initialising APM...
>>> barometer calibration complete
>>> GROUND START
=> DC > Connected to vehicle on -host: tcp:127.0.0.1:5760 -baudrate: 57600
=> DC > Socket server is started on -host: localhost -port: 7000
=> DC > Allow connection for 12 clients.
=> DC > It is initialized!
=> DC > Client ('127.0.0.1', 53112) conected!
=> DC > received message: {"command": "arm", "args": {}}
=> DC > message decode: {u'args': {}, => DC > Number of Clients: 1u'comman
d': u'arm'
}
=> DC Commands > Basic pre-arm checks {}
=> DC Commands > Arming motors
=> DC Commands > Waiting for arming...
=> DC > received message: {"command": "setSpeed", "args": {"airSpeed": 7,"
groundSpeed": 5}}
=> DC > message decode: {u'args': {u'groundSpeed': 5, u'airSpeed': 7}, u'c
ommand': u'setSpeed'}
=> DC Commands > Waiting for arming...
=> DC > received message: {"command": "takeOff", "args": {"z": 10}}
=> DC > message decode: {u'args': {u'z': 10}, u'command': u'takeOff'}
=> DC Commands > Waiting for arming...
=> DC Commands > Waiting for arming...
>>> ARMING MOTORS
>>> GROUND START
=> DC Commands > Waiting for arming...
>>> Link timeout, no heartbeat in last 5 seconds
>>> ...link restored.
>>> Initialising APM...
=> DC > Running command: {u'args': {}, u'command': u'arm'}
=> DC Commands > set speed {u'groundSpeed': 5, u'airSpeed': 7}
=> DC > Running command: {u'args': {u'groundSpeed': 5, u'airSpeed': 7}, u'
command': u'setSpeed'}
```

```
maik@lascar-new-nb:~/drone-control/startup$ sh start-simulator.sh
os: linux, apm: copter, release: stable
SITL already Downloaded and Extracted.
Ready to boot.
Execute: /home/maik/.dronekit/sitl/copter-3.3/apm --home=-30.0321547,-51.2
205995,0,0 --model=quad
Started model quad at -30.0321547,-51.2205995,0,0 at speed 1.0
bind port 5760 for 0
Starting sketch 'ArduCopter'
Serial port 0 on TCP port 5760
Starting SITL input
Waiting for connection ....
bind port 5762 for 2
Serial port 2 on TCP port 5762
bind port 5763 for 3
Serial port 3 on TCP port 5763
New connection on serial port 2
```

```
maik@lascar-new-nb:~/drone-control$ python drone-clients/test-circle/clien
t-test-circle.py
maik@lascar-new-nb:~/drone-control$
```

(a) On the left, the framework log. On the right at the top, the log presented by the simulator. On the right at the bottom, it is the command used to initialize the client responsible for generating the mission commands.



(b) QGroundControl mission log.

Figura 40: Framework experimental tests on simulator.

The second set of tests with the framework was performed in the field and served as a test base for the clients of the proposed framework for the implementation of the case study in precision agriculture. In these experiments the client algorithms were run on embedded hardware and its activities were supported by the integration with the proposed framework. During the case study presented in Section 8.1 on self-regulated application of agrochemicals the client NDVI performed the acquisition of the indexes that were stored in a log by decoding the messages sent to the framework. Already in the case study experiments presented in Section 8.2, the proposed algorithms generated the matrices of gains responsible for adjusting the position of the UAV on the plantation lines. In this case, the framework received these parameters in the form of messages and decoded them by sending commands directly to the flight controller, thus doing the autonomous guidance of the UAV on the plantation.

8.1 NDVI Part of the Case Study

Two types of experiments were carried out in this part of the case study. First, the performance of the developed algorithm was evaluated in the laboratory with the hardware up and running. Similar tests were conducted for both implementations in Python and C++. The tests were conducted for nine different resolutions of images acquired in real time by the Raspberry Pi NoIR camera. The highest resolution was 1920 pixels width by 1080 pixels height, and the lowest resolution was 133 pixels width by 100 height. The measured variables for the performance tests were: time to process a frame (TPF), in seconds; FPS; percentage of CPU utilization; RAM memory usage, in MB; and virtual memory usage (SWAP).

The second stage of tests carried out on the project took place at the experimental agronomic farm belonging to UFRGS, in the city of Eldorado do Sul, state of Rio Grande do Sul, Brazil. As an example of the collected images, Fig. 41 shows an infrared image captured by the Raspberry Pi NoIR camera during the UAV flight over the crops, following the scenario specifications contained in the Chapter 6.

There, test flights were carried out in an experimental corn field, with the algorithms running and their results are sending to the developed framework which generated the results logs. In this part of the experiment the NDVI levels were obtained for each image resolution at one point of the field, following the same procedures for both implementations of the algorithm. At the same moment of obtaining the results of the NDVI algorithm, readings were performed using the Trimble GreenSeeker handheld crop sensor (GOVAERTS; VERHULST, 2010), a widely used equipment in precision agriculture for acquiring NDVI readings.

To facilitate understanding, the obtained results in both experiments were separated by implementation. The results of the implementation in Python can be seen in Table 1, while the results for implementation in C++ can be seen in Table 2.

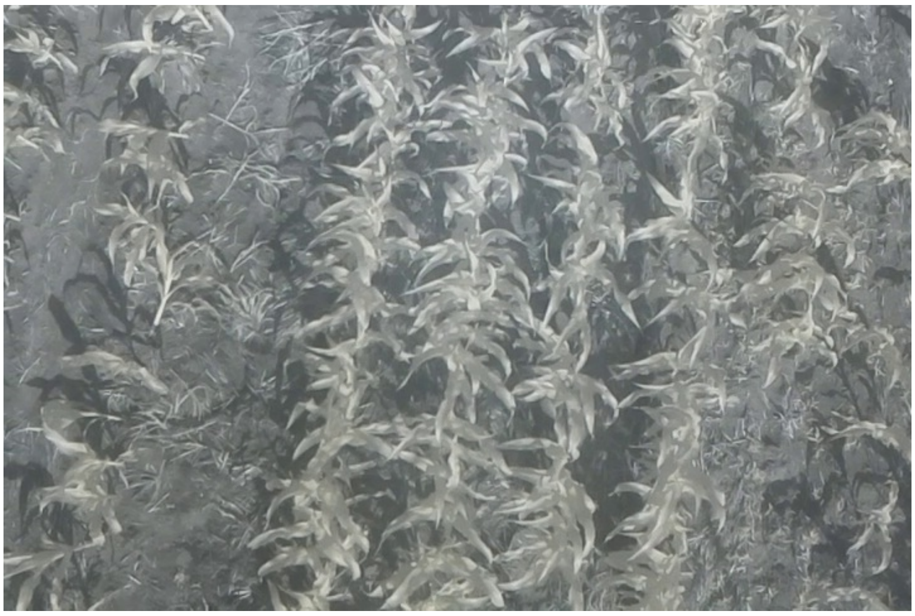


Figura 41: Infrared image captured by Raspberry Pi NoIR camera in the test scenario.

Tabela 1: Python results

Resolution	**TPF** (s)	**FPS**	**CPU** (%)	**RAM** (MB)	**SWAP** (MB)	**NDVI**$_{avg}$
1920 x 1080	0.6	1.64	20	122	240.5	0.59
1336 x 768	0.29	3.4	18	65.7	179.5	0.61
1280 x 720	0.25	3.91	16	55.5	173.6	0.6
1024 x 768	0.24	4.1	15	49.5	127.8	0.59
800 x 600	0.16	5.99	12	46.9	168.1	0.57
640 x 480	0.12	7.82	11	41.9	164.8	0.56
320 x 240	0.083	12.03	7	39.1	160.1	0.55
160 x 120	0.075	13.29	5	37.1	159.6	0.54
133 x 100	0.021	45.98	4	36.8	159.4	0.59

Analyzing the performance of both implementations of the proposed algorithm, it was assessed that both presented low CPU utilization, around 20 % for image resolutions in high definition (HD) and low memory consumption, in average 70 MB. It is also possible to observe that the performance achieved with the implementation in C++ is significantly higher than the one in Python, up to 100 % more efficient.

Considering the relationship between the speed of UAV movement and FPS detailed in Chapter 2, it is possible to state that both implementations met the real time requirements of the project, and leaving available resources for the execution of the other algorithms of image processing that may be deployed in the system.

Another important observation is that the resolution of the processed image does not interfere in the average level of NDVI for the same region of the plantation field. The small variation of NDVI in the different tested resolutions contained in the results refers to luminance variation in the image of the scene during the period of testing. The main luminance the scene was provided by the sun light. From this experiment it was shown that it is possible

Tabela 2: C++ results

Resolution	**TPF** (s)	**FPS**	**CPU** (%)	**RAM** (MB)	**SWAP** (MB)	**NDVI**$_{avg}$
1920 x 1080	0.3	3.28	23	65.7	132.3	0.61
1336 x 768	0.15	6.59	22	39.6	111.3	0.6
1280 x 720	0.14	6.8	20	30.3	101.2	0.58
1024 x 768	0.12	8.29	21	30.5	99.7	0.578
800 x 600	0.083	12.021	17	28.7	97.6	0.58
640 x 480	0.059	16.92	17	24.6	94.5	0.58
320 x 240	0.024	41.51	9	21.7	91.5	0.59
160 x 120	0.023	43.028	5	20.2	90.9	0.6
133 x 100	0.022	44.21	4	20.3	91.1	0.59

to use lower resolution to capture the average NDVI, increasing the performance and consequently the availability of resources for the implementation of other algorithms to make the system more "intelligent"or being able to perform other applications concurrently.

To validate the developed algorithm, the NDVI levels readings acquired with its usage were compared to the NDVI levels acquired by the GreenSeeker. The readings were taken at the same time and location; otherwise, possible luminance differences could result in great variations. A set of 120 of samples were measured for the same points to perform this comparison, and the obtained results can be observed in the chart presented in Fig. 42.

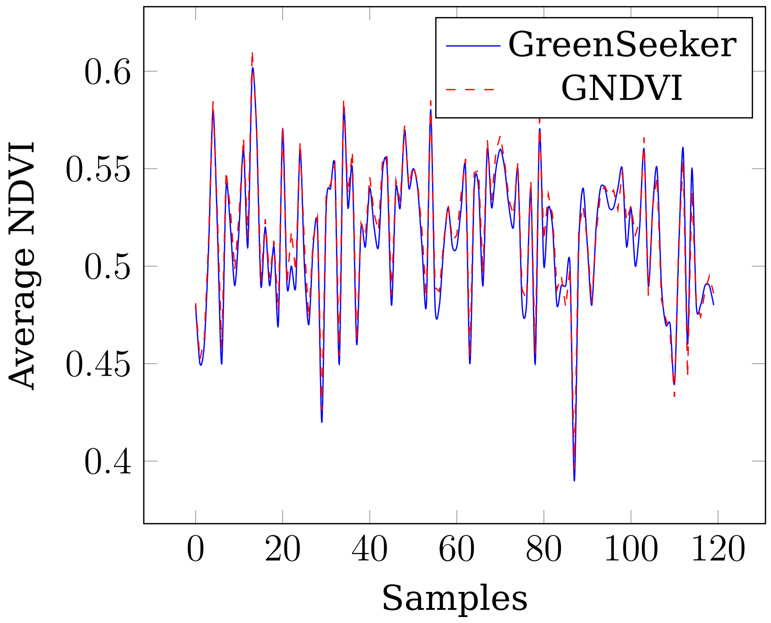


Figura 42: Comparison between NDVI acquired by the proposed embedded GNDVI algorithm and the COTS GreenSeeker device.

Whereas the GreenSeeker uses the red visible channel to calculate the NDVI, as shown in (1), the developed algorithm uses the visible green channel as shown in (6), due to the unavailability of the hardware resource of the Raspberry Pi NoIR camera. In this context, a small variation may occur in achieved steady levels regarding the diversity of the used bands, which was noticed in the performed tests and shown in the Fig. 42. However, this fact does not present any problem in using the indexes to estimate the health of the crop (WANG et al., 2007). What is important to remark is that the difference between the GreenSeeker and the GNDVI results are very small, as can be observed in Fig. 42, as the two graphs almost overlap each other. For the set of 120 samples collected during the performed experiment, the average different between the GNDVI and GreenSeeker was $0,0055416667$ with an standard deviation of $\pm 0,0033378296$.

8.2 Crop Row Detection Part of the Case Study

In this part of the case study, two types of experiments were performed in order to prove the feasibility and correct operation of the proposed system. The first type of experiment was carried out in the laboratory and consists of analyzing the performance and functioning of the developed software through different input parameters. It is considered as input parameters different spatial resolutions of the image captured by the camera, being $160\text{x}120$ the lowest resolution and $1920\text{x}1080$ the highest resolution used in the tests. The internal parameters of the algorithm such as thresholds have been adjusted once since the general proposal of the system is to work for different resolutions of images and variations of luminance and noise in the background of the images without the need to change them.

A scenario of test was set up to simulate the flight in the plantation and to perform the results acquisition in the laboratory. The first step is pre-loading the test images acquired with the proposed hardware in the flight conditions for spraying. Test images were available in all spatial resolutions used in laboratory tests. After the Raspberry Pi camera image acquisition step, the previously loaded test images are used in the pretreatment step then the frame captured by the Raspberry Pi camera is discarded. Only this alteration in the algorithm was necessary for the achievement of the laboratory experiments. The benchmarks were made using the *chrono* library available in C++ (VAN WEERT; GREGOIRE, 2016). The start and end times of the execution of each step of the developed algorithms were stored in CSV files for later analysis.

In this experiment, 1000 tests were performed in the laboratory using some test videos, for each of the eight spatial resolutions evaluated. The average time for the algorithm to perform all proposed steps on an image captured by the camera and generate the gain matrix (24) can be seen in Fig. 43. The algorithm averaged processing time was of 15.13 ms for the lowest image resolution and 620.09 ms for the highest resolution images. As expected, the processing time increases considerably when the spatial resolution increases and consequently it increases the amount of pixels to be processed.

The FPS rate obtained in the experiments can be seen in Fig. 44. Reaching 93.62 FPS for the lowest resolution and 1.63 FPS for the highest resolution and considering the use of embedded commercial hardware of low performance it is considered that the algorithm presented sufficient performance to conduct the UAV respecting its relation between speed and detachment on the

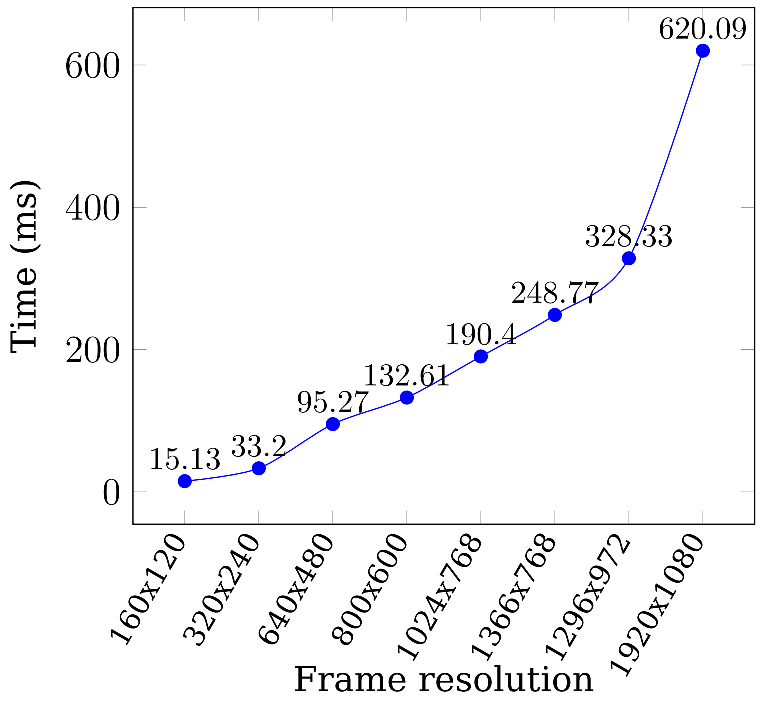


Figura 43: Average time spent to processing a frame at different resolutions.

identified lines on the spraying scenario.

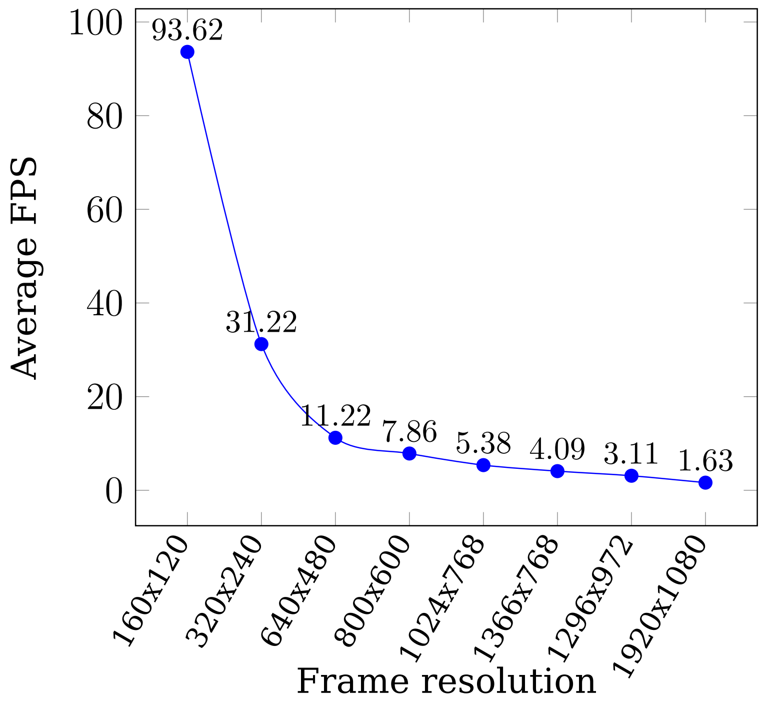


Figura 44: Average FPS at different resolutions.

In addition, the average time spent by each step of the algorithm is observed in Fig. 45. Considering that the image acquisition stage is responsible for about 21.25% of the time spent in processing a frame, it is concluded that making the change of the image acquisition hardware would improve the performance. The performance of the algorithm is limited by the acquisition rate of the used camera. The high cost of the pre-treatment and filtering stages of the lines were expected and compensated by the optimization of the time consumed by the Hough Transform, which in turn has high processing time exponentially increasing with the increase of noise in the image input.

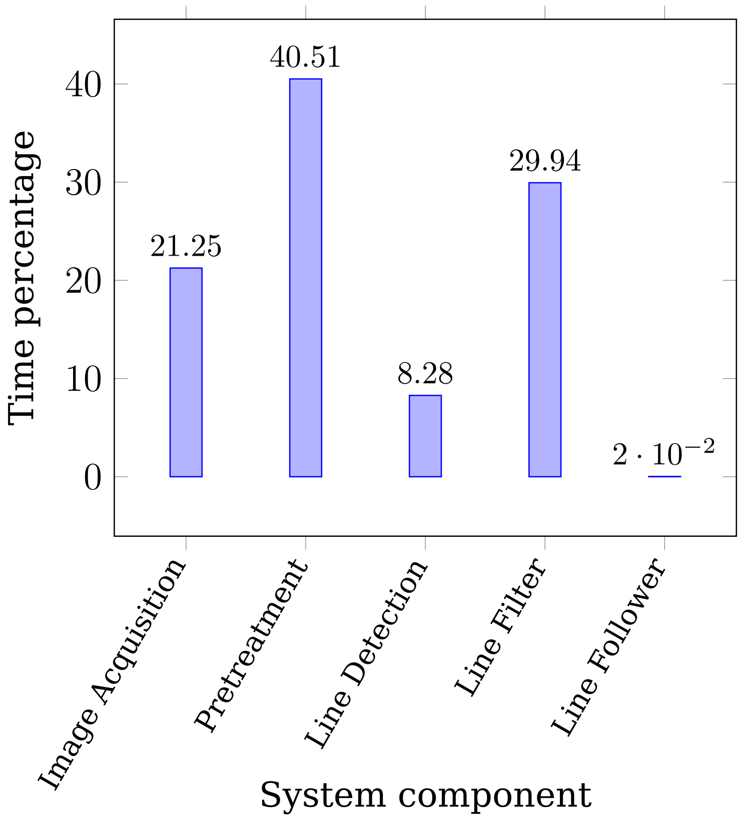


Figura 45: Distribuiton of time to processing of each of the modules of proposed system.

However, in addition to evaluating performance, it is necessary to evaluate the precision of the algorithm in detecting all crop rows. For this, the average number of lines detected for each resolution used in the experiments can be seen in Fig. 46. These results show only the detection step not covering the filtering lines step. The images used in these tests had between eight and twelve lines to be identified.

It is possible to notice that for the resolution 160x120 the loss of information was considerably large making the algorithm incapable of performing the detection of all crop rows. This can be observed since the input images had on average 8 detectable lines. Already for resolutions above 320x240 all lines were detected. Analyzing these results it is also noticed that for resolutions above 320x240 the level of detail of the images increased and consequently the background noise level that went through the pretreatment was greater, thus causing the identification of false positives.

In almost 100% of the cases, the false positives were caused by weeds present between the crop rows. The presence of these false positives did not influence the result of the calculation of the gain matrix in any of the performed tests, because the line filter algorithm was also efficient in eliminating these false positives. However, for the resolution 160x120 that presented a loss of information, then this resolution should not be used.

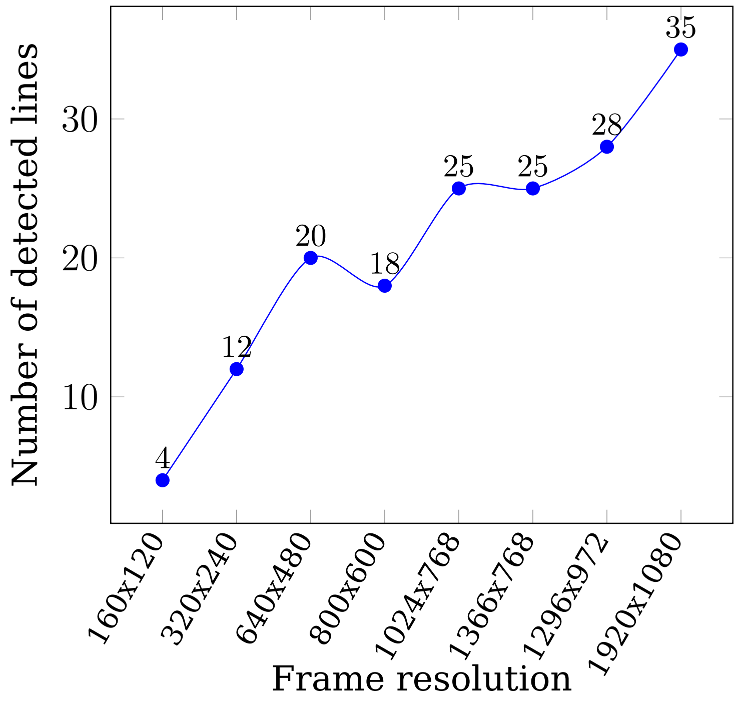


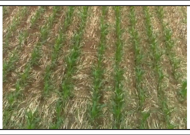
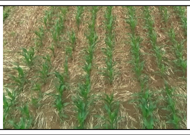
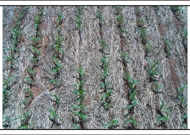
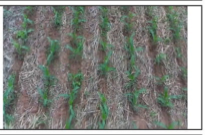
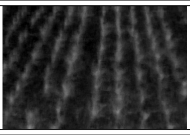
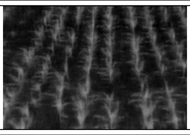
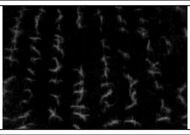
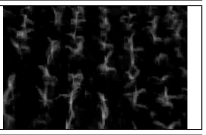
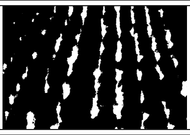
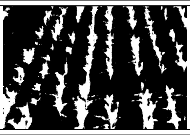
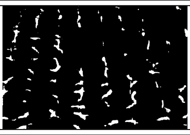
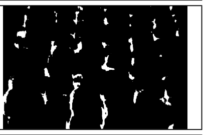
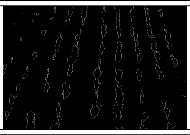
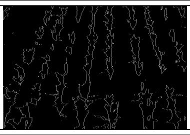
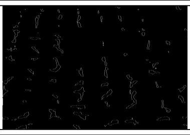
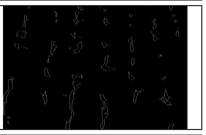
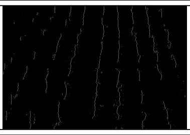
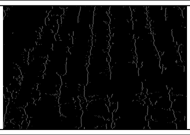
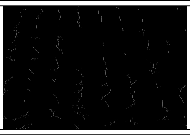
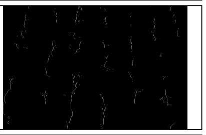
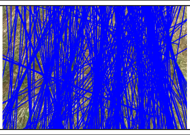
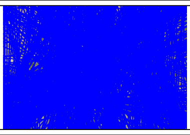
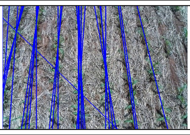
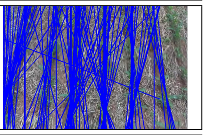
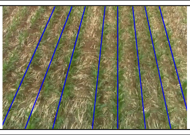
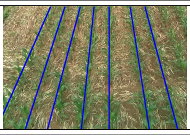
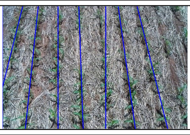
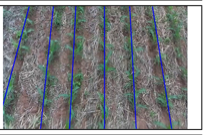
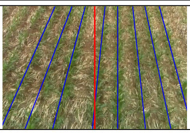
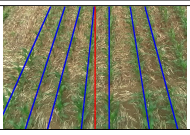
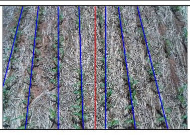
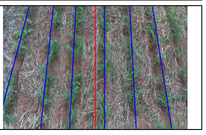
Figura 46: Accuracy of the algorithm in detecting lines at different resolutions.

Analyzing all the conclusions obtained in the laboratory experiments, the best image acquisition resolution to be used in the proposed system is 320x240. This resolution got the best performance and precision in detection of all lines. In addition, this resolution was the one that presented less noise caused by the level of detail of the images. As a consequence, the resolution 320x240 was selected for the realization of the second stage of the experiments.

The second stage of the experiment was performed at the Experimental Agronomic Farm of the Federal University of Rio Grande do Sul. For this experiment, it was chosen a maize crop, because in precision agriculture this type of crop is grown by creating the lines characteristics identified by the proposed algorithm. The first experiment performed in the field was the acquisition of images to assemble the image database used in the development of this work. The flights to mount the database took place on November 18th, 2016, where the acquisition of videos in two schedules, with different speed configurations and with a fixed height of 2 m of the ground were realized.

The first flight was performed at a speed of 2 m/s, which presented good image quality. For the 5 m/s flight, the images presented a blurry effect caused by the high speed compared to the low rate of acquisition of the Raspberry Pi camera. From this database it was developed the algorithms and the whole proposed system. Experiments performed in the field can be seen in the Table 3.

Tabela 3: Accuracy of the system in identifying crop rows in different scenario settings.

Date	11-18-2016		11-09-2017	
Hour	02:19 p.m.	03:24 p.m.	01:32 p.m.	03:36 p.m.
UAV Speed	2 m/s	5 m/s	2 m/s	5 m/s
UAV Altitude	2 m	2 m	2 m	2 m
Captured Image				
2G-R-B Transform				
Threshold Filter				
Edge Detection				
Average Edges				
Hough Transform				
Line Filter				
Line Follower				

In November 9th, 2017, new tests were performed, but this time with the embedded system running in the developed UAV hardware platform with the proposed algorithms. This stage of testing was carried out in the field. The precision agriculture plantation of the corn crop had a characteristic shape as shown in Fig. 38. In this day, two flights with the same mission settings were performed with the UAV speed configured to 2 m/s and then to 5 m/s.

The video with the image captured by the UAV was recorded at the same time that the algorithm used these images to conduct the UAV on the plantation. The flights took place perfectly considering the rectangular shape of the plantation. A more specific treatment to make the curves at the end of each crop row is necessary, but this is not the focus of the present work. Another relevant point is that the algorithm reached the expected results, with higher performance than the related works, thus making it possible to identify the crop rows in the initial phase of growth and to conduct the UAV even when the captured images had abrupt changes in luminosity.

This phenomenon can be clearly seen in the Table 3. Another problem that the algorithm could undergo would be background noise caused by weeds and vegetation at the edges of the plantation. Because it is an experimental plantation of precision agriculture, these last two problems were not identified.

9 CONCLUSIONS

This work presented the development of a framework that allows the exchange of messages/commands of guidance and control of mission between UAVs and embedded computer vision algorithms, providing autonomy to these equipments.

All the proposed software was implemented in low cost embedded hardware architecture. In the performed experiments, the framework presented satisfactory performance in the exchange and in the decoding of messages, and also in the execution of the abstracted messages commands. With the proposed framework, new ways were opened for the easy integration of visual computing algorithms into the flight controllers of the UAVs, creating tools for the integration of several algorithms and/or several vehicles into a single autonomous system. The proposed solution is scalable, allowing the multiplicity of the system both in the number of UAVs and in the number of algorithms that control them.

The next step of this work would be the development of a management interface that allows control of all the resources made available to the independent system created by the proposed framework. With this interface, it would be easy to make changes in the course of the missions at run time. As another future work stands the installation and testing of the proposed system in a UAV that has the spraying hardware, thus testing the feasibility of the system working with all parts together.

In Section 9.1 the conclusions obtained specifically with the development of the part of case study related to the punctual and autoregulated application of agrochemicals are presented. In the section 9.2 the specific conclusions obtained with the development of the part of case study related to the autonomous guidance of UAVs on precision agriculture plantations are presented.

9.1 NDVI Part of the Case Study

This part of the case study reports the design and development of an embedded algorithm to support automatic agrochemical spraying. The complete solution comprises of a software system running in an embedded computing hardware integrated into a COTS UAV platform.

The performed tests showed that the implemented algorithms are computationally efficient, taking into account the time requirements of the applications, and effective because similar levels of NDVI measurements with constant variation were obtained compared to those calculated by a GreenSeeker COTS sensor. The tests showed that the resolution of the processed image does not interfere in the average NDVI obtained for each frame, due to the fact that the index be more related to the proportionality of the elements that compose the image than to its resolution. The use of lower resolutions can increase performance and consequently the availability of resources, allowing the simultaneous execution of other applications.

The next steps of this project are the implementation of a classifier algorithm that makes the adjustment of the actuators for the regulated application of fertilizers and pesticides through the developed algorithm.

9.2 Crop Row Detection Part of the Case Study

This part of the case study presented the proposal of an embedded guiding system for UAVs on precision agriculture plantations in an initial state of plant growth. A crop row detection algorithm and an algorithm that is responsible for generating the necessary parameters to control the UAV through a connection between the embedded hardware and the flight controller have been proposed.

From the obtained results it can be observed that the proposed system achieved satisfactory performance compared to the related works found in the literature. In addition, the proposed pre and post processing algorithms were fundamental to achieve enhanced performance in the Hough Transform that composes the crop row detection algorithm. Another important point that can be verified with the results of the performed tests is that the proposed system was able to correctly conduct the UAV even when the crop rows appeared in a curve in the image captured by the camera. This is due to the fact that the proposed algorithm divides the curves into segments of a line. In the performed field experiments using the proposed hardware, the detection algorithm achieved a detection rate of 100 % of the crop rows for images with resolutions above 320x240. The system performance was also measured in laboratory experiments and reached 31.22 FPS for images with smaller resolution, 320x240, and 1.63 FPS for the higher resolution images, 1920x1080. The experiments carried out proven the feasibility of using the proposed system for the autonomous guidance of UAVs for precision agriculture applications.

As future work for this part of the work, the plan is to further develop the system including an artificial intelligence module capable of conducting the UAV more accurately into the area of the planting after the end of each line, thus enabling the flight in plantations with irregular scenarios, or important discontinuations. Using these algorithms, it is possible to train it to recognize certain patterns of the path and to increase the abilities of the autonomous navigation algorithm, so that the system is able to handle these irregularities of the plantation.

REFERÊNCIAS

3DROBOTICS. **Setting up a Simulated Vehicle (SITL)**. Available at: <http://python.dronekit.io/develop/sitl_setup.html>, Accessed in: 23/10/2018.

AKRAM, T. et al. Towards real-time crops surveillance for disease classification: exploiting parallelism in computer vision. **Computers & Electrical Engineering**, Amsterdam, v.59, p.15 – 26, 2017.

AL-KAFF, A. et al. Survey of computer vision algorithms and applications for unmanned aerial vehicles. **Expert Systems with Applications**, Amsterdam, v.92, p.447 – 463, 2018.

APVRILLE, L.; TANZI, T.; DUGELAY, J. L. Autonomous drones for assisting rescue services within the context of natural disasters. In: URSI GENERAL ASSEMBLY AND SCIENTIFIC SYMPOSIUM (URSI GASS), 31., 2014, Beijing, China. **Proceedings. . .** New Jersey:IEEE, 2014. n.31, p.1–4.

ARAúJO, P. et al. Air-SSLAM: a visual stereo indoor slam for aerial quadrotors. **IEEE Geoscience and Remote Sensing Letters**, New Jersey, USA, v.14, n.9, p.1643–1647, Sept 2017.

BASSO, B. et al. Spatial validation of crop models for precision agriculture. **Agricultural Systems**, Amsterdam, v.68, n.2, p.97 – 112, 2001.

BILLS, C.; CHEN, J.; SAXENA, A. Autonomous MAV flight in indoor environments using single image perspective cues. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2011, Shanghai, China. **Proceedings. . .** New Jersey:IEEE, 2011. p.5776–5783.

BOYKOV, Y.; VEKSLER, O.; ZABIH, R. Fast approximate energy minimization via graph cuts. **IEEE Transactions on pattern analysis and machine intelligence**, New Jersey, USA, v.23, n.11, p.1222–1239, 2001.

BRAY, T. **The javascript object notation (json) data interchange format**. Available at: <http://www.rfc-editor.org/info/rfc7159>, Accessed in: 01-04-2017.

CAMPOY, P. et al. Computer Vision Onboard UAVs for Civilian Tasks. **Journal of Intelligent and Robotic Systems**, Amsterdam, v.54, n.1, p.105–135, Aug 2008.

CANO, E. et al. Comparison of Small Unmanned Aerial Vehicles Performance Using Image Processing. **Journal of Imaging**, Basel, Switzerland, v.3, n.1, p.4, 2017.

CHEN, S.; GUO, S.; LI, Y. Real-time tracking a ground moving target in complex indoor and outdoor environments with UAV. In: IEEE INTERNATIONAL CONFERENCE ON INFORMATION AND AUTOMATION (ICIA)., 2016, Ningbo, China. **Proceedings. . .** New Jersey:IEEE, 2016. p.362–367.

CHENG, H. et al. An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2017, Vancouver, BC, Canada. **Proceedings. . .** New Jersey:IEEE, 2017. p.1732–1738.

CHOI, H. et al. Open source computer-vision based guidance system for UAVs on-board decision making. In: IEEE AEROSPACE CONFERENCE, 2016, Big Sky, MT, USA. **Proceedings. . .** New Jersey:IEEE, 2016. p.1–5.

CHOWDHARY, G. et al. GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft. **Journal of Field Robotics**, Hoboken, EUA, v.30, n.3, p.415–438, 2013.

CONTICELLI, F.; ALLOTTA, B. Two-level visual control of dynamic look-and-move systems. In: MILLENNIUM CONFERENCE. IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION. SYMPOSIA PROCEEDINGS (CAT. NO.00CH37065), 2000, San Francisco, CA, USA. **Proceedings. . .** New Jersey:IEEE, 2000. v.4, p.3784–3789 vol.4.

DAWEI, Y. **The Rise of China’s Drones**. Available at: <http://www.slate.com/articles/technology/caixin/2015/07/drones_in_china_can_the_country_s_industry_for_uavs_bloom.html>, Accessed in: 19/01/2018.

DUDA, R. O.; HART, P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. **Commun. ACM**, New York, NY, USA, v.15, n.1, p.11–15, Jan. 1972.

FAIÇAL, B. S. et al. The use of unmanned aerial vehicles and wireless sensor networks for spraying pesticides. **Journal of Systems Architecture**, Amsterdam, v.60, n.4, p.393–404, 2014.

FAIçAL, B. S. et al. Fine-Tuning of UAV Control Rules for Spraying Pesticides on Crop Fields. In: IEEE INTERNATIONAL CONFERENCE ON TOOLS WITH

ARTIFICIAL INTELLIGENCE, 26., 2014, Limassol, Cyprus. **Proceedings. . .** New Jersey:IEEE, 2014. p.527–533.

FOROUZAN, B. A. **TCP/IP Protocol Suite**. 2.ed. New York, NY, USA: McGraw-Hill, Inc., 2002.

FREITAS, E. P. et al. Real Time Embedded Image Processing System for Points of Interest Detection for Autonomous Unmanned Aerial Vehicles. In: AEROSPACE TECHNOLOGY CONGRESS, 2016, Solna, Stockholm, Sweden. **Proceedings. . .** Sweden: FTF, 2016. p.1–13.

GARCíA-SANTILLáN, I. D. et al. Automatic detection of curved and straight crop rows from images in maize fields. **Biosystems Engineering**, Amsterdam, v.156, n.Supplement C, p.61 – 79, 2017.

GONZALEZ, R.; WINTZ, P. **Digital image processing**. Massachusetts, EUA: Addison-Wesley, 1977.

GOVAERTS, B.; VERHULST, N. **The normalized difference vegetation index (NDVI) Greenseeker(TM) handheld sensor**: toward the integrated evaluation of crop management. part a - concepts and case studies. Mexico: CIMMYT, 2010.

GUERRERO, J. et al. Automatic expert system based on images for accuracy crop row detection in maize fields. **Expert Systems with Applications**, Amsterdam, v.40, n.2, p.656 – 664, 2013.

HOUGH, P. **Method and means for recognizing complex patterns**. Patent Number US 3069654 A.

HUANG, T. Computer Vision: evolution and promise. **19th CERN School of Computing**, Egmond aan Zee, The Netherlands, p.21–25, 1996.

HUH, S.; SHIM, D. H.; KIM, J. Integrated navigation system using camera and gimbaled laser scanner for indoor and outdoor autonomous flight of UAVs. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 2013, Tokyo, Japan. **Proceedings. . .** New Jersey:IEEE, 2013. p.3158–3163.

ILLINGWORTH, J.; KITTLER, J. A survey of the hough transform. **Computer Vision, Graphics, and Image Processing**, Amsterdam, v.44, n.1, p.87 – 116, 1988.

JI, R.; QI, L. Crop-row detection algorithm based on Random Hough Transformation. **Mathematical and Computer Modelling**, Amsterdam, v.54, n.3, p.1016–1020, 2011.

JIANG, G. Q.; ZHAO, C. J.; SI, Y. S. A machine vision based crop rows detection for agricultural robots. In: INTERNATIONAL CONFERENCE ON WAVELET ANALYSIS AND PATTERN RECOGNITION, 2010, Qingdao, China. **Proceedings. . .** New Jersey:IEEE, 2010. p.114–118.

JIANG, G.; WANG, Z.; LIU, H. Automatic detection of crop rows based on multi-ROIs. **Expert Systems with Applications**, Amsterdam, v.42, n.5, p.2429 – 2441, 2015.

JINLIN, X.; WEIPING, J. Vision-based guidance line detection in row crop fields. In: INTELLIGENT COMPUTATION TECHNOLOGY AND AUTOMATION (ICICTA), 2010, Qingdao, China. **Proceedings. . .** New Jersey:IEEE, 2010. v.3, p.1140–1143.

JUNG, Y.; BANG, H.; LEE, D. Robust marker tracking algorithm for precise UAV vision-based autonomous landing. In: INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION AND SYSTEMS (ICCAS), 15., 2015, Beijing, China. **Proceedings. . .** New Jersey:IEEE, 2015. p.443–446.

KAEHLER, A.; BRADSKI, G. **Learning OpenCV 3**: computer vision in c++ with the opencv library. Massachusetts, USA: O'Reilly Media, Inc., 2016.

KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. **Transactions of the ASME–Journal of Basic Engineering**, USA, v.82, n.Series D, p.35–45, 1960.

KANELLAKIS, C.; NIKOLAKOPOULOS, G. Survey on Computer Vision for UAVs: current developments and trends. **Journal of Intelligent & Robotic Systems**, Amsterdam, v.87, n.1, p.141–168, July 2017.

KN, R. et al. Automatic detection of powerlines in UAV remote sensed images. In: INTERNATIONAL CONFERENCE ON CONDITION ASSESSMENT TECHNIQUES IN ELECTRICAL SYSTEMS (CATCON), 2015, Bangalore, India. **Proceedings. . .** New Jersey:IEEE, 2015. p.17–21.

LEE, H.; JUNG, S.; SHIM, D. H. Vision-based UAV landing on the moving vehicle. In: INTERNATIONAL CONFERENCE ON UNMANNED AIRCRAFT SYSTEMS (ICUAS), 2016, Arlington, VA, USA. **Proceedings. . .** New Jersey:IEEE, 2016. p.1–7.

LEIRA, F. S.; JOHANSEN, T. A.; FOSSEN, T. I. A UAV ice tracking framework for autonomous sea ice management. In: INTERNATIONAL CONFERENCE ON UNMANNED AIRCRAFT SYSTEMS (ICUAS), 2017, Miami, FL, USA. **Proceedings. . .** New Jersey:IEEE, 2017. p.581–590.

LI, X. et al. Visible defects detection based on UAV-based inspection in large-scale photovoltaic systems. **IET Renewable Power Generation**, United Kingdom, v.11, n.10, p.1234–1244, 2017.

LIU, X.; GUO, B.; MENG, C. A method of simultaneous location and mapping based on RGB-D cameras. In: INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION, ROBOTICS AND VISION (ICARCV), 14., 2016, Phuket, Thailand. **Proceedings. . .** New Jersey:IEEE, 2016. p.1–5.

LIU, Y. et al. A Novel Trail Detection and Scene Understanding Framework for a Quadrotor UAV With Monocular Vision. **IEEE Sensors Journal**, New Jersey, EUA, v.17, n.20, p.6778–6787, Oct 2017.

LUO, H. et al. Optimization of Pesticide Spraying Tasks via Multi-UAVs Using Genetic Algorithm. **Mathematical Problems in Engineering**, Cairo, Egypt, v.2017, 2017.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: FIFTH BERKELEY SYMPOSIUM ON MATHEMATICAL STATISTICS AND PROBABILITY, VOLUME 1: STATISTICS, 1967, Berkeley, Calif. **Proceedings...** Berkeley:University of California Press, 1967. p.281–297.

MAGREE, D.; JOHNSON, E. N. Combined laser and vision-aided inertial navigation for an indoor unmanned aerial vehicle. In: AMERICAN CONTROL CONFERENCE, 2014, Portland, OR, USA. **Proceedings...** New Jersey:IEEE, 2014. p.1900–1905.

MALEK, S. et al. Efficient Framework for Palm Tree Detection in UAV Images. **IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing**, New Jersey, EUA, v.7, n.12, p.4692–4703, Dec 2014.

MEIER, L. **Pixhawk Flight Controller Hardware Project**. Available at: <https://pixhawk.org/>, Accessed in: 04/01/2018.

MEIER, L. et al. **QGroundControl**: ground control station for small air-land-water autonomous unmanned systems. Available at: <http://qgroundcontrol.com/>, Accessed in: 04/01/2018.

MEIER, L. et al. **Mavlink**: micro air vehicle communication protocol. Available at: <http://qgroundcontrol.org/mavlink/start>, Accessed in: 04/01/2018.

MHATRE, V. et al. Embedded video processing and data acquisition for unmanned aerial vehicle. In: INTERNATIONAL CONFERENCE ON COMPUTERS, COMMUNICATIONS, AND SYSTEMS (ICCCS), 2015, Kanyakumari, India. **Proceedings...** New Jersey:IEEE, 2015. p.141–145.

MINAEIAN, S.; LIU, J.; SON, Y. J. Vision-Based Target Detection and Localization via a Team of Cooperative UAV and UGVs. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, New Jersey, EUA, v.46, n.7, p.1005–1016, July 2016.

MOHAMED, M. K.; PATRA, S.; LANZON, A. Designing simple indoor navigation system for UAVs. In: MEDITERRANEAN CONFERENCE ON CONTROL AUTOMATION (MED), 19., 2011, Corfu, Greece. **Proceedings...** New Jersey:IEEE, 2011. p.1223–1228.

MORANDUZZO, T.; MELGANI, F. Automatic Car Counting Method for Unmanned Aerial Vehicle Images. **IEEE Transactions on Geoscience and Remote Sensing**, New Jersey, EUA, v.52, n.3, p.1635–1647, Mar. 2014.

MUNSHI, A. The opencl specification. In: IEEE HOT CHIPS SYMPOSIUM (HCS), 21., 2009, Stanford, CA, USA. **Proceedings. . .** New Jersey:IEEE, 2009. p.1–314.

MYNENI, R. B. et al. The interpretation of spectral vegetation indexes. **IEEE Transactions on Geoscience and Remote Sensing**, New Jersey, USA, v.33, n.2, p.481–486, 1995.

O. DEMERECI M. VARUL, N. S.; ODABAS, M. S. Plant counting with low altitude image processing. In: SIGNAL PROCESSING AND COMMUNICATIONS APPLICATIONS CONFERENCE (SIU), 23., 2015, Malatya, Turkey. **Proceedings. . .** New Jersey:IEEE, 2015. p.2266–2269.

PESTANA, J. et al. Computer vision based general object following for GPS-denied multirotor unmanned vehicles. In: AMERICAN CONTROL CONFERENCE, 2014, Portland, OR, USA. **Proceedings. . .** New Jersey:IEEE, 2014. p.1886–1891.

PESTANA, J. et al. A Vision-based Quadrotor Multi-robot Solution for the Indoor Autonomy Challenge of the 2013 International Micro Air Vehicle Competition. **Journal of Intelligent & Robotic Systems**, Amsterdam, v.84, n.1, p.601–620, Dec 2016.

PHANG, S. K. et al. Autonomous Mini-UAV for indoor flight with embedded on-board vision processing as navigation system. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTATIONAL TECHNOLOGIES IN ELECTRICAL AND ELECTRONICS ENGINEERING (SIBIRCON), 8., 2010, Listvyanka, Russia. **Proceedings. . .** New Jersey:IEEE, 2010. p.722–727.

PRASAD, M. **Basic Concepts of Computer Vision**. Available at: <http://maxembedded.com/2012/12/basic-concepts-of-computer-vision/>, Accessed in: 10/01/2018.

PURI, V.; NAYYAR, A.; RAJA, L. Agriculture drones: a modern breakthrough in precision agriculture. **Journal of Statistics and Management Systems**, London, United Kingdom, v.20, n.4, p.507–518, 2017.

RAJA, V. m. Vision based landing for unmanned aerial vehicle. In: AEROSPACE CONFERENCE, 2011, Big Sky, MT, USA. **Proceedings. . .** New Jersey:IEEE, 2011. p.1–8.

RICHARDSON, M.; WALLACE, S. **Getting started with raspberry PI**. California, EUA: O'Reilly Media, Inc., 2012.

SENTHILNATH, J. et al. A novel hierarchical clustering technique based on splitting and merging. **International Journal of Image and Data Fusion**, London, United Kingdom, v.7, n.1, p.19–41, 2016.

SHAH, U.; KHAWAD, R.; KRISHNA, K. M. Detecting, localizing, and recognizing trees with a monocular MAV: towards preventing deforestation. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2017, Singapore. **Proceedings. . .** New Jersey:IEEE, 2017. p.1982–1987.

SOBEL, I.; FELDMAN, G. A 3x3 isotropic gradient operator for image processing. **Pattern classification and scene analysis**, New York, p.271–272, 1968. Presented talk at the Stanford Artificial Project.

SUWANSRIKHAM, P.; SINGKHAMFU, P. Indoor vision based guidance system for autonomous drone and control application. In: INTERNATIONAL CONFERENCE ON DIGITAL ARTS, MEDIA AND TECHNOLOGY (ICDAMT), 2017, Chiang Mai, Thailand. **Proceedings. . .** New Jersey:IEEE, 2017. p.110–114.

TOMIC, T. et al. Toward a Fully Autonomous UAV: research platform for indoor and outdoor urban search and rescue. **IEEE Robotics Automation Magazine**, Nova Jersey, EUA, v.19, n.3, p.46–56, Sept 2012.

TORRES-SáNCHEZ, J.; LóPEZ-GRANADOS, F.; PEñA, J. An automatic object-based method for optimal thresholding in UAV images: application for vegetation detection in herbaceous crops. **Computers and Electronics in Agriculture**, Amsterdam, v.114, p.43 – 52, 2015.

UPTON, L. **What's that blue thing doing here?** Available at: <https://www.raspberrypi.org/blog/whats-that-blue-thing-doing-here/>, Accessed in: 24/01/2018.

VAN WEERT, P.; GREGOIRE, M. **C++ Standard Library Quick Reference**. New York, EUA: Apress, 2016.

VIDOVIĆ, I.; CUPEC, R.; HOCENSKI, Ž. Crop row detection by global energy minimization. **Pattern Recognition**, Osijek, Croatia, v.55, p.68–86, 2016.

WANG, F.-M. et al. New vegetation index and its application in estimating leaf area index of rice. **Rice Science**, China, v.14, n.3, p.195–203, 2007.

WARD, S. et al. Autonomous UAVs wildlife detection using thermal imaging, predictive navigation and computer vision. In: IEEE AEROSPACE CONFERENCE, 2016, Big Sky, MT, USA. **Proceedings. . .** New Jersey:IEEE, 2016. p.1–8.

WOEBBECKE, D. et al. Color indices for weed identification under various soil, residue, and lighting conditions. **Transactions of the ASAE-American Society of Agricultural Engineers**, Saint Joseph, EUA, v.38, n.1, p.259–270, 1995.

WU, C.-D. et al. Linking Student Performance in Massachusetts Elementary Schools with the "Greenness" of School Surroundings Using Remote Sensing. **PLOS ONE**, San Francisco, California, US, v.9, n.10, p.1–9, 10 2014.

WU, K. J. et al. Development of an indoor guidance system for unmanned aerial vehicles with power industry applications. **IET Radar, Sonar Navigation**, Georgia, USA, v.11, n.1, p.212–218, 2017.

WU, Y.; SUI, Y.; WANG, G. Vision-Based Real-Time Aerial Object Localization and Tracking for UAV Sensing System. **IEEE Access**, USA, v.5, p.23969–23978, 2017.

YIN, Y. et al. Robust Visual Detection–Learning–Tracking Framework for Autonomous Aerial Refueling of UAVs. **IEEE Transactions on Instrumentation and Measurement**, Beijing, China, v.65, n.3, p.510–521, March 2016.

YUAN, C. et al. Unmanned aerial vehicle based forest fire monitoring and detection using image processing technique. In: IEEE CHINESE GUIDANCE, NAVIGATION AND CONTROL CONFERENCE (CGNCC), 2016, Nanjing, China. **Proceedings. . .** New Jersey:IEEE, 2016. p.1870–1875.

YUAN, C.; LIU, Z.; ZHANG, Y. Vision-based forest fire detection in aerial images for firefighting using UAVs. In: INTERNATIONAL CONFERENCE ON UNMANNED AIRCRAFT SYSTEMS (ICUAS), 2016, Arlington, VA, USA. **Proceedings. . .** New Jersey:IEEE, 2016. p.1200–1205.

ZHANG, C. et al. An UAV navigation aided with computer vision. In: CHINESE CONTROL AND DECISION CONFERENCE, 26., 2014, Changsha, China. **Proceedings. . .** New Jersey:IEEE, 2014. v.1, n.26, p.5297–5301.

ZHANG, J.; LIU, W.; WU, Y. Novel Technique for Vision-Based UAV Navigation. **IEEE Transactions on Aerospace and Electronic Systems**, Provo, USA, v.47, n.4, p.2731–2741, Oct 2011.

ZHOU, H. et al. Efficient Road Detection and Tracking for Unmanned Aerial Vehicle. **IEEE Transactions on Intelligent Transportation Systems**, Los Angeles, USA, v.16, n.1, p.297–309, Feb 2015.