Fall 2019

Development of an Autonomous Aerial Toolset for Agricultural Applications

Terrance Life

DEVELOPMENT OF AN AUTONOMOUS AERIAL TOOLSET FOR AGRICULTURAL APPLICATIONS

A Capstone Project Presented in Partial Fulfillment
of the Requirements for the Degree Bachelor of Electrical Engineering
with Honors College Graduate Distinction at
Western Kentucky University

Author: **Terrance Life**
November 26, 2019

CE/T Committee
Dr. Farhad Ashrafzadeh, Chair
Professor Ali Buendia
Professor Joel Lenoir

Dedication

For my family that always encouraged and supported me.

Acknowledgements

I would like to express my very great appreciation to Mr. Ali Buendia for his valuable and constructive suggestions during the planning and development of this research work. His willingness to give his time so generously has been very much appreciated.

I would also like to thank Dr. Farhad Ashrafzadeh and the Center for Energy Systems for providing the space and opportunity to complete this project.

Abstract

According to the United Nations, the world population is expected to grow from its current 7 billion to 9.7 billion by the year 2050. During this time, global food demand is also expected to increase by between 59% and 98% due to the population increase, accompanied by an increasing demand for protein due to a rising standard of living throughout developing countries. [1] Meeting this increase in required food production using present agricultural practices would necessitate a similar increase in farmland; a resource which does not exist in abundance. Therefore, in order to meet growing food demands, new methods will need to be developed to increase the efficiency of farming, thereby increasing yield from the present land.

One way in which this problem can be solved is through the usage of autonomous aerial systems to scout for problems which could potentially affect the crop yield – such as nutrient deficiency, water stress, or diseases. Once located, this data can be used to determine the proper treatment for the field to alleviate the problem. Through this process, resources can be reduced to the required minimum, while problems affecting the crop yield will still be corrected, allowing greater production with a lower amount of resources.

This project on the application of Unmanned Aerial Vehicles (UAV's) to the field of agriculture consisted of two phases. First, a study was conducted on the required background to define the problem statement and what solutions were available for this application. This consisted of first defining the operations within agriculture where UAV's could be used to increase efficiency, and then the sensors, hardware, and software these operations would require. The remainder of the project consisted of evaluating the tools which could be utilized to develop such a solution. Primarily, the project focused on software tools – programming software, simulation environments, and machine learning algorithms – which could be utilized by future students to develop a functional hardware and software toolchain for the research of autonomous systems for agricultural applications.

After analyzing these development solutions, a set of tools was selected which showed promise in the creation of a functional solution. It was demonstrated that the core functions required for a UAV-based agricultural solution – navigation, perception, and feature detection – could be implemented within these systems, implying that they could be integrated into a full solution. As the tools were selected to ensure the developed algorithms would be transferable to physical platforms, this additionally supports a physical system could also be developed. The present work is part of the Autonomous Systems Lab which belongs to the WKU Center for Energy Systems. The author hopes that this project contributes to the advancement of the curriculum within the engineering department and serves as a foundation for future students developing autonomous systems, perception, and applied artificial intelligence at WKU.

Vita

EDUCATION

Western Kentucky University – Bowling Green, KY December 2019
B.S. in Electrical Engineering – Mahurin Honors
College Graduate Honors Capstone: *Development of an Autonomous Aerial Vehicle for Agricultural Applications*

Carol Martin Gatton Academy of Mathematics and Science, May 2016
Bowling Green, KY

PROFESSIONAL EXPERIENCE

The Center for Energy Systems, WKU October 2017-Present
Research Associate

AWARDS & HONORS

Summa Cum Laude, WKU, December 2019

PROFESSIONAL MEMBERSHIPS

Tau Beta Pi (TBP)

PRESENTATIONS

Life, T. (2019, March). *Development of an Advanced Research Platform for Aerial Autonomous Navigation.* Poster presented at the WKU Student Research Conference. Bowling Green, KY.

Contents

List of Figures

List of Tables

Chapter 1: Introduction to Precision Agriculture with UAV's

One method to increase the efficiency of farmland and reduce resources such as fertilizer, pesticide, herbicide, and water, is through the application of precision farming techniques. These techniques are fundamentally concerned with applying resources strategically where required rather than uniformly across an entire field. [1] This process is carried out in three distinct stages. First using sensors to monitor the field and report on various metrics which might affect the yield on the crops in regions of the field. After which, these metrics are inspected by professional analysts who can recommend a plan for an efficient application of resources to maximize yield without expending unnecessary materials which might cause environmental or commercial damage. For example, by analyzing the soil at specific points throughout a field, a map can be made of the field to strategically apply fertilizer in appropriate amounts over the field. This ensures that the varied sections of the field get the fertilizer they need while not placing more fertilizer than is required on already fertile sections; reducing costs and environmental impacts. This variable application plan is then carried out using intelligent machinery in conjunction with positioning equipment.

These three stages can be implemented with a multitude of systems, from measuring hand-extracted soil samples to purchasing satellite images of the field. The application which will be explored in this paper is the use of UAV's in this process, specifically the data acquisition and treatment application stages. Many commercially available UAV's are designed to carry some form of camera or payload while in flight. While this camera is often one intended to operate over the visible light spectrum, it is possible to substitute a different sensor intended to observe a different frequency (such as an infrared sensor) or to more accurately observe a designated color (such as the red-green spectrum). This enables the UAV's to be deployed over a defined area, over which they will capture images of the field which can be used to assess a variety of metrics to determine crops health and predict yield.

Why UAV's?

During the image acquisition phase, there are two broad categories from which data is collected, sampling and remote sensing. Sampling, covering methods where samples of the material to be measured are collected and tested, is both labor and time intensive and could potentially affect the crop if samples must be taken directly from the plants. The alternative method, remote data acquisition, has been around for multiple years but some of the more common methods – manned aircraft and satellite imaging – are rarely used outside of government locations and the largest of farms. Due to these systems' operating height, the detail with which they can photograph is limited by image resolution. On the contrary, UAV's operate at a far lower height than manned aircraft and satellites, leading

to shorter sensing distances, which provides numerous advantages in data acquisition frequency and quality. Due to the lower flight altitude and distances, the UAV's are less sensitive to the effects of cloud cover as opposed to manned aircraft and satellites, which enables more frequent covering of the field as well as faster turnaround times and higher image quality. This higher image quality allows greater accuracy in differentiating elements of the image during analysis, which in turn enables a level of detail about the field closer to that of sampling which can be used for a variety of metrics about the crop. The benefit to this being that the data collection was non-invasive and nondestructive [2].

Scouting for Problems

Three of the most significant factors which limit the growth of crops are weeds, pests, and diseases. Early detection of these factors so that appropriate action can be taken can improve the health, and thereby yield, of the crop. Through routine monitoring and identification of these issues, the problems can be handled faster and more efficiently. This increases yield through reducing the total damage done by these limiting factors, while also saving costs by enabling herbicides and pesticides to be applied locally, rather than sprayed uniformly.

Weed Detection

Weeds can be classified as any plant which grows in the field other than the crop which was intentionally planted there. These plants consume resources which could have been used by the intended crop, thereby restricting the growth and development of these plants; reducing the final yield. Monitoring the field allows these invasive plants to be identified and handled before significant resources are consumed, which serves to increase yield. Since weeds are often found in sparse clumps, at least during the beginnings of their growth, short sample distances are needed to obtain the necessary data for detection and differentiation [2]. As previously mentioned, UAV's are a system which can facilitate these lower sample distances, allowing weeds to be located during the analysis phase. This increased detail also allows differentiation between plant species, both between crops and weeds as well as between different species of weed. This rapid detection and classification enable specialized management methods to be utilized, such as local application of a targeted herbicide [3]. By decreasing the total amount of herbicide needing to be used for weed-management both costs for herbicide and the environmental impact from runoff and contamination can be lessened [2].

Pest and Disease Detection

The most common method of treating for pests and diseases is to uniformly spray pesticide across the entire crop [3]. Much like with the application of herbicide to limit the spread of weeds within the field, precision agriculture techniques can also be applied to the spraying of pesticides to strategically locate and eradicate pests when infestations occur. Psirofonia, Samaritakis, Eliopoulos, and Potamitis outline such a method through a case

study on the use of UAV-based observation of palm trees infested with red palm weevils. These insects lay their eggs in the crowns of palm trees after which the larvae begin to work their way down the tree, consuming it. Infested palms show a steady loss of leaves which can be noticed within 3 months of infestation. Through the use of UAV-collected images, the crowns of trees can be inspected to locate suspicious trees which can then be investigated more closely. If an infestation is found, pesticide can be applied, or the tree removed to limit the spread of the weevils [4]. Through this process, environmental impact and financial costs can be mitigated.

A similar methodology can be applied to the detection and management of diseased plants. Another case study summarized by Psirofonia, et. Al detailed the usage in UAV imaging to detect the presence of the Xylella fastidiosa in olive trees. By flying over the olive grove and taking images, trees were located which had discolored and dried leaves; marking them as suspicious and potential candidates for infestation. Analysts were then able to bring the UAV back through to the suspicious trees to obtain better data. Ultimately, the discoloration was determined to be the result of a recent fire, however the process for detecting abnormalities in the canopy was found to be successful; indicating there is promise with the process [4]. In a similar application, digital cameras were utilized with Google's Tensorflow system to detect diseases within the cassava plant in Swahili (fig. 1).

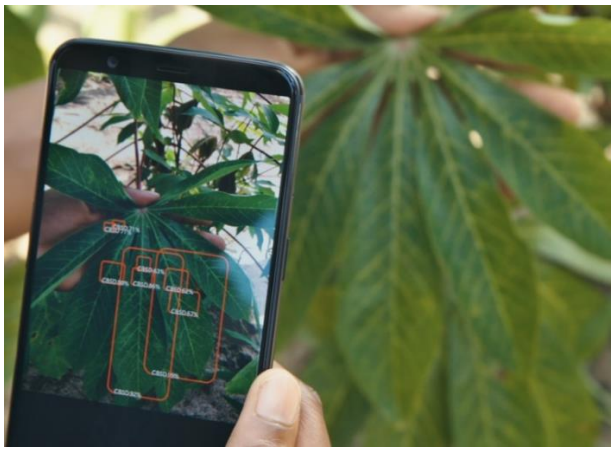


Figure 1: Nuru, a phone-based app which uses similar technology to detect diseases in the cassava plant created using Google Tensorflow. https://www.blog.google/technology/ai/ai-takes-root-helping-farmers-identity-diseased-plants/

Nutrient Stress

While not an invasive external factor, the lack of nutrients, such as nitrogen, in the soil can be just as significant in reducing yields. By utilizing UAV-based sensors, it is possible to assess the chemical composition of the plant and determine these deficiencies so that management practices can be used to correct them [2]. Conversely, the same sensing algorithm can detect areas where sufficient quantities of this nutrient exist, and no action needs to be taken (fig. 2). Using these techniques, fertilizers can be selectively applied to the areas where they would be beneficial while refraining from fertilizing the

areas which are abundant in these elements. The process saves cost and provides environmental benefit; as excess nitrogen in the soil can be converted into the greenhouse gas nitrous oxide by soil dwelling bacteria [2].

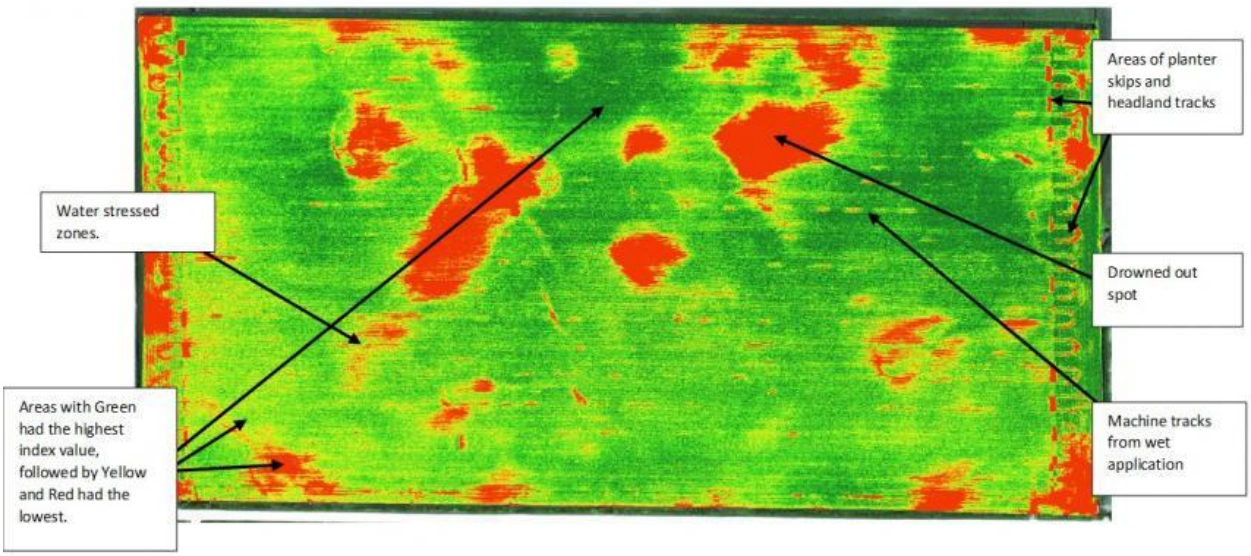


Figure 2: Map of field which has been analyzed for water stress. https://event38.com/news/color-nir-and-ndvi-imagery-according-to-iowa-state/

Monitoring to Prevent Yield loss

While performing scouting missions to determine potential issues can assist in preventing and managing threats to crop yield, another factor is to frequently check throughout the growing cycle to ensure proper management. This continual monitoring allows for the effects of management techniques to observed so that further management can be optimized. To facilitate this, the field is organized into management zones, areas of the field which have similar soil properties, which receive treatment tailored to their specific needs. Such management could also factor in other needs of the plants such as the irrigation. Using thermal sensors, the rate of transpiration in plants can be determined, which is proportional to the water the plants are receiving. This allows water stress or abundance to be calculated throughout the field, allowing irrigation to be improved [2]. Pantazi, Moshou, Alexadridis, Whetton, and Mouazen provide an example which shows the combination of these two measures, where one section of a field had high water content due to poor drainage, water logging the section and thereby decreasing the yield [5]. Through continual management, the irrigation pattern within the field could be determined over multiple planting cycles; allowing these areas to be either untended or the local irrigation improved in order to make the land conducive to growing crops.

Crop Management

After data is gathered and analyzed to determine management zones and the overall management plan, the next step is to implement the strategic applications determined by the analysis. Using positioning equipment, properly equipped UAV's, and a treatment plan, it is possible to automate this process; applying the appropriate treatment when and where it is needed to efficiently manage the detected problems. As discussed earlier, these treatments can involve the local application of specified chemicals such as pesticides, herbicides, and fertilizers.

A case study which outlines this functionality was covered by Psirofonia, et. Al concerning the operation of UAV's in conjunction with e-traps. The e-trap simulates a smart insect capturing device which monitored the number of insects it had captured over a period; sending a notification if it passed a defined threshold. This triggered a UAV to deploy and spray pesticides within the area surrounding the trap [4]. This process could be implemented with other requirements for crop management, such as the application of herbicides or fertilizers. Since the UAV's payload can be changed as required, the same UAV's could be used for all necessary treatment applications. Additionally, changing payloads could enable the same UAV's which performed the image-collection to apply the treatment; further raising the value of an investment in UAV-based crop management.

Chapter 2: UAV's and Required Technologies

Types of UAV's

Multiple styles of UAV exist on the market today, including fixed wing aircraft, multirotor UAV's, and single-rotor helicopters. Depending on the location where the UAV needs to operate and the function which it needs to complete, different systems might be more beneficial than other systems. The flight characteristics, maximum mission duration, maneuverability, and payload limit all need to be considered to select the optimal UAV for an application. The two primary distinctions in UAV systems are between fixed-wing aircraft and rotorcraft, as these distinctions most fundamentally affect the flight characteristics and maneuvers the UAV can perform.

Fixed-Wing Aircraft

Fixed-wing aircraft produce lift through the pressure difference caused by the flow of air over the wings of the aircraft. Due to this, as long as the aircraft remains moving at a specific minimum velocity, the system will remain in the air. This greatly increases the efficiency of powering the system, as power can be concentrated on horizontal propulsion as opposed to split between thrust and lift. Consequently, these systems can perform longer and faster flights than rotorcraft systems, allowing significantly more ground to be covered per flight. Such efficiency comes at drawbacks to its maneuverability and sensor detail however, as these systems must continue moving at a defined speed to remain aloft. This

restricts them from being able to hover and mandates the UAV fly above the canopy level to prevent crashes. Therefore, close inspections of a limited area cannot be carried out with this system, nor can it be used to apply treatment to a limited area. Additionally, many of these systems require a significant area for take-off or landing, much like an airplane requires a runway. In recent years however, some systems have been developed which enable vertical take-off and landing while maintaining the fixed-wing flight mechanism. While this provides the flexibility of landing in a smaller area, design changes to enable vertical take-off can reduce the efficiency of forward flight [6].

Rotorcraft

The lift for rotorcraft is derived from the moving air generated from the movement of the rotors (fig. 3). While this requires more power to stay aloft and move than fixed-wing systems, this source of lift independent of the thrust enables rotorcraft to hover in a single place for data collection and treatment operations. The other key benefit from this is their capacity to take-off, operate, and then land in a confined area. These capabilities allow the system to be used for more precise data collection and treatment operations than their fixed-wing counterparts. These capabilities come at the cost of shorter flight times and slower flight speeds, which restrict the operating range of these systems. In addition, these systems are often lighter than fixed-wing UAV's, giving them lower weight limits for their payloads and more sensitivity to weather. The magnitude of this sensitivity varies depending on the exact nature of the rotorcraft. Multi-rotor UAV's are most sensitive but are also capable of operating in a much smaller area than single rotor helicopters, which trade precision in movement for increased endurance and less weather sensitivity. Additionally, helicopters require more care to ensure safe operation as the larger rotors have the potential to inflict more severe injuries when rotating [6].

Figure 3: Worker using a UAV for pesticide application http://www.chinadaily.com.cn/business/tech/2016-11/25/content_27481877.htm

Data Collection and Interpretation Technologies

While the use of UAV's as a platform for image capture and data acquisition is a critical portion of applying precision agriculture techniques, it is not a monolithic solution.

Other technologies are required to facilitate the process of collecting data, processing the data, and generating a treatment plan. These include the requisite cameras to collect the variety data from the aerial platform as well as software to convert the acquired data into a usable form. These include such technologies as image meshing software which coordinates images with the positions they were taken and combine the captured images into a cohesive whole, allowing for analysis and location to be interfaced [7]. The final category of these auxiliary technologies is machine learning algorithms, software which while not strictly required for data analysis, can improve the process.

Computer vision systems

For the UAV to collect the required data about a field, a variety of image acquisition equipment must be utilized. The most basic of these sensors are digital cameras affixed to the UAV's, which provide similar data as visually inspecting the field. In precision agriculture, however, these systems often are insufficient to provide the detail needed to make decisions about the health of crops. For this reason, an alternative to cameras, the spectrometer, can be used. Unlike cameras which record only three spectral bands (Red, Green, and Blue), spectrometers record additional spectral bands; allowing them to differentiate between colors which might appear identical with only 3 bands [2]. Two primary versions of the spectrometer are used: hyperspectral sensors which feature numerous overlapping bands over a 10-20 nm range and multispectral sensors which can feature bands in ranges of 50-100 nm (fig. 4). By examining specific frequencies, it is possible to visually determine parameters about the crop, including chlorophyll content [2] and grain properties [8], allowing the health and yield of a crop to be assessed. Plant health can also be assessed through the use of thermal imaging to estimate the transpiration rates of vegetation within the scanning area. As transpiration rises, the temperature of the canopy falls due to the energy leaving the plant as water vapor. Thus, by indirectly locating areas of lower transpiration, these systems can determine which areas of the field are experiencing water stress [2]. By analyzing this data, treatment plans can be generated to provide areas of the field with the exact chemicals the crops require and to effectively irrigate the field.

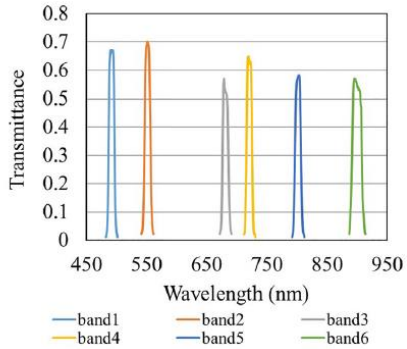


Figure 4: Graph of the color bands in an example spectrometer. [9]

Machine Learning

In precision agriculture practices, for gathered data to be used to develop management plans it must be interpreted; a service generally performed by specialized users [4]. Much like how the process of scouting the field and gathering data was automated using UAV's, this process can also be automated through the use of machine learning algorithms. These algorithms seek to enable a machine to accept a defined form of input (for example a picture of a field taken by a UAV) and then perform a desired function; either classifying the images into defined categories (such as infected or non-infected plants) or clustering the data into sets based on patterns found in the images (for instance, images of vegetation versus soil). By providing the machine with a large set of data and instructing the machine to classify or cluster the data repeatedly, then providing feedback on the successfulness of the endeavor, the process can be steadily improved [3]. These trained systems can then be used to analyze new data, allowing them to be applied to multiple operations in agriculture.

The application for these systems which most closely follows from the UAV applications discussed in this paper is utilizing these systems to detect problems scouted by UAV's. For instance, Liakos, Patrizia, Busato, Moshou, and Pearson cite a study where multispectral images were captured in a field and then given to a machine learning algorithm which detected weeds within the crop with 53-94% accuracy [3]. Similarly, these same processes were used for disease detection by differentiating between healthy wheat plants and those infested with Yellow Rust and Septoria tritici with accuracies of 99.83% and 98.7% respectively [3]. These detection methods could be used in conjunction with the automated scouting and treatment application techniques to apply herbicide and pesticide strategically; effectively automating most of the problem management.

Additionally, these systems can be applied to yield prediction and quality; enabling the system to classify and quantify the product which can be expected from a given field. For instance, Patrício and Rieder outlined the benefits of using a machine learning algorithm for the process of classifying wheat grains. Wheat grain classification is often fraught with misclassifications due to human error and fatigue, as well as the process being dependent on human opinion and thus varies between analysts. For these reasons, the use of machine learning algorithms can improve this industry, allowing the grains to be judged by a standard set of parameters and performed without the effect of fatigue [8]. Patrício and Rieder also outlined examples where this process is applied to rice grains, detecting fungus and variation within the harvested crop in order to ensure that quality product is delivered to consumers [8]. Before the product is harvested, similar systems can also be employed to predict the yield of the crop as outlined by Pantazi, Et. Al. in an experiment to predict the yield of wheat crop with soil parameters. These systems achieved close to 92% accuracy in predicting areas of low yield, 70.5-85% for medium yield areas, and 75-83% accuracy for areas of high yield [5]. This can be utilized to provide better focus for crop management techniques throughout the growing cycle and estimate the results of these techniques when applied.

Commercial UAV Solutions

As a result of the previously established benefits of UAV assisted farming, several manufacturers have capitalized on the technology. They have created commercial platforms tailored for usage in agricultural data collection; ranging from add-ons for other UAV's to entire systems designed from the ground up for the farming application. These systems include both fixed-wing and rotor-based platforms from a variety of manufacturers, ranging in cost from $2000 to $25000. [10]

Fixed Wing Solutions

As mentioned previously, the main advantages for fixed-wing packages are their relatively larger coverage ranges, being able to cover 500-600 acres in a single run [11,12]. Sensefly, a subsidiary of Parrot, has released a line of commercial mapping UAV's called the eBee series; with the eBee SQ (fig. 5) being developed specifically for agricultural use. This system enables the user to survey approximately 500 acres per flight, while mapping with a precision of up to 1.5in/pixel with the included Parrot Sequoia multispectral camera. This is further supported by the eMotion Ag flight planning software, which facilitates 3-dimensional flight plans alongside hand launching and automatic landing [11].

Figure 5: eBee SQ Platform, https://www.sensefly.com/drone/ebee-sq-agriculture-drone/

For users who require more customizable sensor another fixed-wing platform, the FireFLY6 PRO (fig. 6), is available for agricultural use through PrecisionHawk. This UAV produced by BirdsEyeView Aerobotics is capable of vertical takeoff and surveying 600 acres in a single flight. In addition to the digital camera included with the product by the manufacturer, the PrecisionHawk package allows the user to select additional sensors – including thermal, multispectral, and LIDAR – as well as their PrecisionAnalytics Agriculture package. This gives users the ability to customize the UAV to fit their needs, as well as access to a robust analysis framework to extract data from the collected images [12].

Figure 6: FireFLY6 Platform, https://www.precisionhawk.com/drones/birdseyeview-firefly6-pro

Rotorcraft Solutions

In addition to their fixed-wing packages, PrecisionHawk also provides modifications for DJI's Matrice 200 and 210 rotorcraft (fig. 7). These platforms, like all rotorcraft, have shorter flight times and coverages than fixed-wing systems, but add the ability to hover and allow for closer scouting ranges. The 200 series UAV's allow for 30 minutes of flight time, with protection against rain and dust, while also permitting flight during high winds and sub-zero temperatures. The platform also features obstacle avoidance and cooperative UAV sensing systems to assist with flight safety and permit the simultaneous operation of multiple UAV's [12].

Figure 7: DJI Matrice 200 Platform, https://www.dji.com/matrice-200-series

In addition to data acquisition, similar commercial platforms can be applied to automate treatment operations. DJI's Agras MG-1 (fig. 8) provides a platform for variable application of pesticides, herbicides, and fertilizer allowing a single system to deliver the necessary treatment to the appropriate areas of the field. Without any payload, the UAV can fly for 24 minutes, with the 10-liter payload, this UAV has only 10 minutes of flight time in which to perform the spraying operation, spraying approximately 2 hectares per deployment [13].

Figure 8: DJI Agras MG-1, https://www.amazon.com/DJI-Intelligent-Scorpion-Drones-Dealer/dp/B074CJCJ2Y

Chapter 3: Introduction to UAV Control

According to a 2018 study, over three-quarters of farmers in the United States use UAV's on a daily or weekly basis [14]. This, combined with the urgent need to begin increasing food production in tandem with the growing world population, reinforces the importance of developments within this industry. In order to achieve the desired increase in production, more than an increased adoption of precision agriculture techniques will be required. An increase in the overall efficiency in these practices will be required as well to unilaterally raise productivity. Additionally, approximately fifty percent of these farmers who utilized UAV technologies contracted with an outside agency in order to perform the analyses [14]. This reinforces the need for smarter systems to increase the efficiency of the data interpretation to inform users on what actions to take to improve crop productivity.

The process of developing an autonomous aerial system which could, after several iterations of development, be deployed to improve agricultural efficiency was comprised of several stages. First, the requisite tools for UAV development – platform, sensors, and development environment – were specified, evaluated, and selected for future development. From this point, work moved to the implementation of these tools to demonstrate their application in navigation methods and object perception. Once this was completed, the final stage was the development of machine learning algorithms to interpret this data to provide usable information. These subsystems could then be combined to form a single deployable system.

UAV Tools

To utilize autonomous UAV's in agricultural functions, several interconnected technologies will be required. First and foremost, the operation of the physical UAV will

need to be considered during development. The way in which the UAV can move and the operations the motors must undertake to do so will be a deciding factor in how the monitoring algorithm would be programmed. This will be further informed by the navigational and data collection sensors which are deployed on the UAV, as these subsystems will determine the methods which can be used for navigation as well as the requirements to collect data. Once these systems are determined, the method of programming the UAV can be determined; alongside an environment to simulate the UAV to allow for more expedient search pattern development.

Governing Equations

The process of navigating with a given quadcopter begins with the equations which describe the motion of the system in 3-dimensional space [15]. These equations describe how changes to the orientation and position of the system can be generated by the thrust from the four rotors; allowing commands to move the quadcopter to be expressed in terms of these four actuators. In the present thesis, the author is not concerned with the development of control systems. However, a brief overview of the dynamic system is included for completeness. These equations can be divided into two basic categories: angular and linear motion.

Angular Motion

The most fundamental motion for a quadcopter is rotation about its x, y, and z axes. This allows the UAV to change the orientation of the rotors, enabling lateral and longitudinal motion. The absolute angular position of the system is described by the vector η, listing the angles about the x, y, and z axes in that order. These define the differences in orientation between the absolute axes (motion and forces relative to the world) and the body reference frame (motion and forces relative to the body of the UAV). The corresponding velocities relative to the body reference frame are also recorded by the vector *v*.

$$\eta = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} \quad v = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

These three angular positions in η can be used to create the rotational frame between these two reference frames.

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\varphi - S_\psi C_\varphi & C_\psi S_\theta C_\varphi + S_\psi S_\varphi \\ S_\psi C_\theta & S_\psi S_\theta S_\varphi + C_\psi C_\varphi & S_\psi S_\theta C_\varphi - C_\psi S_\varphi \\ -S_\theta & C_\theta S_\varphi & C_\theta C_\varphi \end{bmatrix}$$

Like all rotational matrices, this can be used to form the transformation matrices between the body and absolute frames.

$$\dot{\eta} = W_\eta^{-1} v, \qquad \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & T_\theta S_\varphi & T_\theta C_\varphi \\ 0 & C_\varphi & -S_\varphi \\ 0 & \frac{S_\varphi}{C_\theta} & \frac{C_\varphi}{C_\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

$$v = W_\eta \dot{\eta}, \ \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\varphi & C_\theta S_\varphi \\ 0 & -S_\theta & C_\theta C_\varphi \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

The angular velocity each rotor i which can be denoted ω_i create both an upward force f_i as well as a torque τ_i about its axis of rotation. This can be modelled:

$$f_i = k\omega_i^2, \quad \tau_i = b\omega_i^2 + I_i \dot{\omega}_i^2$$

In which k, b, and I_i *represent* the lift coefficient, drag coefficient, and moment of inertia respectively. Together, these rotor forces generate a thrust T in the direction of the body frame's z-axis and torques τ_φ, τ_θ, and τ_ψ about their respective axes. This can be written:

$$T = \sum_{i=1}^{4} f_i = k \sum_{i=1}^{4} \omega_i^2$$

$$\tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(\omega_4^2 - \omega_2^2) \\ lk(\omega_3^2 - \omega_1^2) \\ \sum_{i=1}^{4} \tau_i \end{bmatrix}$$

Where l is the distance between the rotor and the centroid of the quadcopter.

Linear Motion

Once the orientation of the quadcopter is known, these angles can be used to convert the rotor thrusts into the absolute reference frame. This can be combined with the angular position vector η into the vector q, which contains all 6 positions in the universal reference frame.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad q = \begin{bmatrix} \xi \\ \eta \end{bmatrix}$$

Using the previous rotation matrix, the body and inertial frame equations can be related and converted. In the inertial reference frame, the total force $m\ddot{\xi}$ is equal to the gravity force $\boldsymbol{G}$ and the total thrust of the rotors $\boldsymbol{RT}$.

$$m\ddot{\xi} = G + RT$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} C_\psi S_\theta C_\varphi + S_\psi S_\varphi \\ S_\psi S_\theta C_\varphi - C_\psi S_\varphi \\ C_\theta C_\varphi \end{bmatrix}$$

When combined, these equations define the dynamic model of the control system for the quadcopter. By controlling the commands send to the four motors, the rotor velocities can be adjusted to affect the desired linear and angular response (fig. 9).

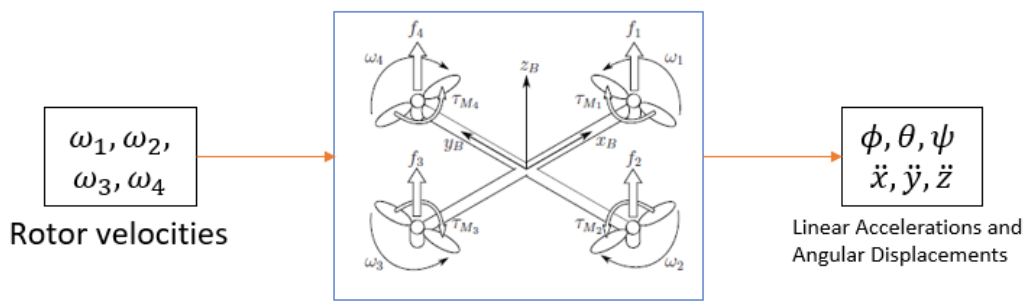


Figure 9: UAV Control System Diagram. UAV model from: [15]

Required Sensors for Agriculture

In order to enable a quadcopter to navigate autonomously and collect the needed information during a mission, several sensors must be integrated into the system. To begin, the UAV must have a method of determining its position, orientation, and the relative direction it is moving. A common method for establishing this information, and the one utilized during development, is using accelerometers and gyroscopes mounted within the quadcopter. With just these sensors, navigation will be performed relative to the origin point of the system, with the sensors being used to calculate changes from this point. The internal gyroscopes will allow the system to measure orientation changes from the starting position, usually parallel to the ground. As described above, this is used to control the necessary thrust to keep the UAV aloft and to control the direction of travel. From the accelerometers, it is possible to derive an object's velocity and relative position from its point of origin; enabling navigation using this reference frame. If the UAV is in an area open to the air, GPS signals can be used to align this reference frame with the global latitude-longitude system, providing an additional method of specifying movement and position. These two systems can be utilized in conjunction to provide a more accurate navigation algorithm; estimating position with the accelerometers and checking with GPS readings periodically. This allows the UAV to continue operating while unable to

communicate with satellites, but to navigate more precisely when this information is available.

Vision Sensors

Once navigation is successfully implemented, the other key system to enable data acquisition using a UAV is the onboard vision system. These sensors can be differentiated by the number of "spectral bands" the system captures; values which represent the magnitude of the radiation in a specified wavelength. These sensors range from relatively cheap, consumer grade cameras to professional grade cameras with fine control over the image. The simplest sensor which can be used for gathering information is a simple digital camera. Though simple, providing only 3 spectral bands, this information is sufficient for some applications with proper filtering systems. For other applications, more precise spectral measurements or systems which can look at wavelengths not read by a digital camera may be necessary. In these cases, a spectrometer can be utilized.

Spectrometers, specifically multispectral and hyperspectral sensors, capture more range bands than are acquired with an RGB camera. This enables the sensors to be utilized in two broad uses. The first is to capture additional wavelengths than are possible with just the RGB ranges of a camera (fig. 10). This enables the system to consider ranges such as the near-infrared (NIR) range, in which reflectance can be correlated to chlorophyll content. Alternatively, these bands can be concentrated in a smaller wavelength range, facilitating the differentiation of colors which would appear identical in the wider spectral bands. This functionality can be used to focus the wavelength analysis on the specific wavelengths around which the chlorophyll will reflect, allowing slight changes to be detected and used to determine the foliage density and chlorophyll content of the crops [9].

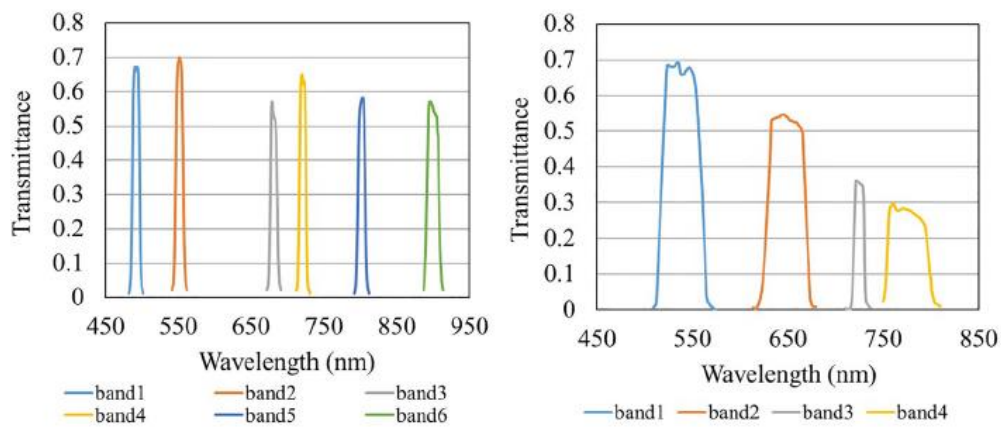


Figure 10: IDK Spectral band graph for example spectrometers [9]

Another significant set of sensors which can be utilized within UAV-based data acquisition are thermal sensors. Unlike previous sensors which use colors of the foliage to determine crop health, thermal sensors can be utilized to estimate water stress in regions

of a field. One way in which plants remove heat from themselves is through transpiration, the evaporating of water from within the plant, much like how human sweat works. In order for this process to occur, sufficient water must exist within the plant to be evaporated. Therefore, plants which are experiencing water stress will have higher foliage temperatures, enabling these areas to be located using thermal imaging.

Tools for System Programming and Simulation

Using both vision and positioning sensors, it is possible for a UAV to collect extensive data whilst coordinating that acquisition with the physical location where it was collected. In order to perform this task, however, functions are needed to pilot the UAV along a planned path and to control the acquisition of information during this flight. To eliminate the necessity for a licensed operator to provide this control, the MATLAB and ROS platforms were explored to provide this control autonomously. Within these systems, the processes of flight control and image acquisition were implemented using simulated environments; eliminating the cost of failed runs and thereby accelerating the pace of algorithm development. Once this implementation was finished, the two systems were compared and the tool which would be used for future developments was selected.

MATLAB and Simulink

The first system which was utilized for algorithm development was Mathworks' MATLAB and Simulink software. MATLAB is a text-based system built specifically for usage in engineering applications ranging from linear algebra to frequency analysis. Its sister product Simulink is designed to mimic the block diagrams used for control systems, being designed for usage in data acquisition, signal processing, and system control applications. Both products come with a wide variety of available add-ons to facilitate the program's usage in specific applications. In this application, the Aerospace Blockset, Simulink 3D simulator, and image processing toolbox were utilized extensively.

Each of these three toolboxes was instrumental in one of the core requirements for the development of the system. The Aerospace toolbox provided systems to simulate the dynamics of an aerial vehicle, allowing both the UAV to internally estimate its position but also facilitating motion within the Simulink 3D simulation. This simulation allows the UAV to be flown within software, and facilitates the viewing of this flight, thereby enabling the development of control systems. This includes modelling a realistic environment with gravity, air pressure, and collision detection; ensuring the programs developed within the simulation are transferable to a physical UAV. In addition to the system dynamics, the simulation allows the views from the UAV to be utilized within the Simulink model, modelling the on-board camera and providing data for image analysis (fig. 11).

Figure 11: Parrot UAV within Simulink 3D

The final subsystem which is central to this application is the image processing toolbox, which facilitates the development of processes to extract information from simulated images and video feeds. These include the location and number of items within the image, such as furrows in a field or the location of cows in an example image (fig. 12).

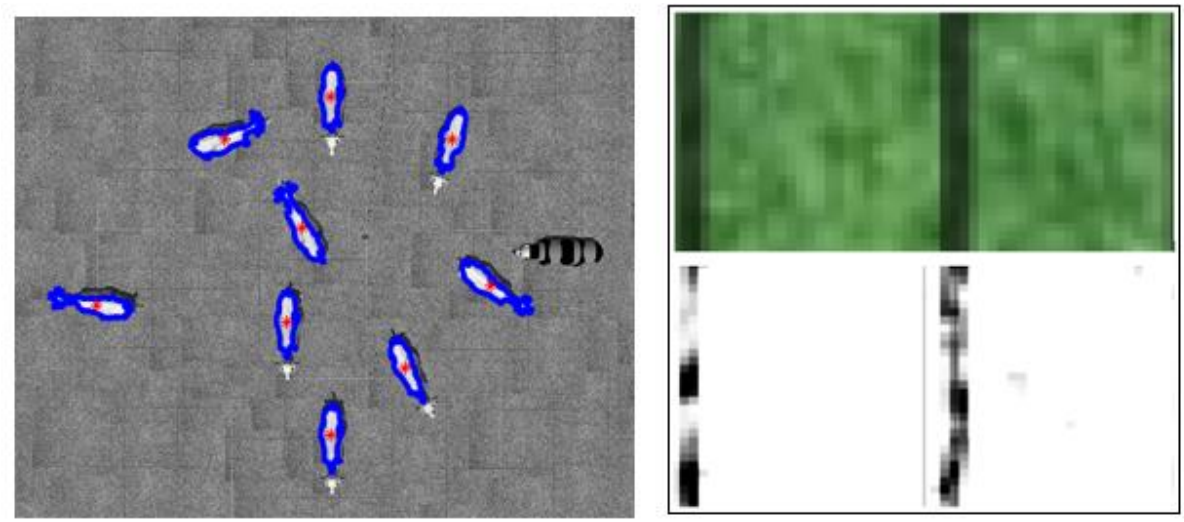


Figure 12: Feature recognition from images using MATLAB for cows (a) and furrows (b)

World Creation in Simulink 3D

Development of a UAV flight algorithm began with the creation of the world in which the UAV would operate. For the project, the original demonstration world was modified from a small set of buildings to a large field with high-contrast furrows. This was performed through the Simulink 3D world editor, where the building models were

deleted, and the ground was changed to an image of farmland in which the furrows had been enhanced through the application of dark lines to the image using Microsoft Paint. Finally, the UAV model was positioned within the world to fly over the field capturing data. This setup eliminated unnecessary information from the world and provided a clear area in which to begin developing feature detection systems (fig. 13).

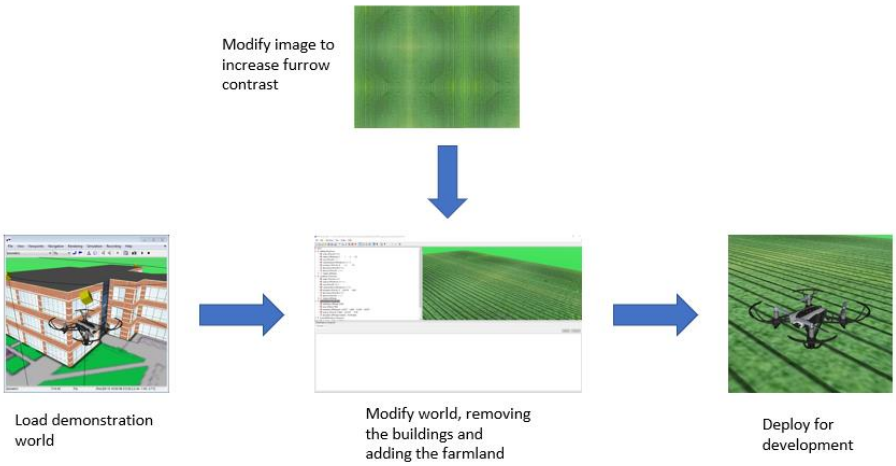


Figure 13: World Creation and Deployment in MATLAB

Flight and Process

Once the world and tools were set up, the process of developing algorithms to control the UAV, acquire data, and process the data commenced. The initial objective was to create a flight path and program the UAV to follow this path while collecting data. After this functionality was established, work began on developing a system which would take the input from the lower camera and detect the furrows the UAV passes over. Finally, a separate algorithm was developed to analyze images of cattle in a field, determining the number of cattle within the image while ignoring any fake cattle.

Olympe and Sphinx

Partway through algorithm development, the Parrot minidrone used by the MATALB support package was discontinued by the manufacturer. As one of the goals for this project was to pave the way for later students to develop systems to control these UAV's, the process of finding another system to develop these algorithms began. A new drone was selected for the next stage of development which utilized a combination of Parrot's Sphinx and Olympe software. These systems use python commands to control the robot while allowing the system to interface with the Gazebo simulator, a physics simulator available on Linux machines. With these, similar projects were carried out, with the potential for future students to continue the development.

Olympe is a python-based system designed to enable the development of control algorithms for Parrot UAV's. This system permits connecting to these UAV's, after which the critical functions of the UAV: position, orientation, camera movement, and video streaming can be controlled programmatically. Though the Olympe control program can be used to control physical UAV through the use of a ground-based control station, the system is intended to be used with the Sphinx simulation package. Sphinx facilitates the simulation of Parrot UAV's within the Gazebo environment, and the visualization of system parameters during the run. This combination allows the user to simulate a Parrot UAV within a custom environment which imitates physical interactions such as gravity, air movement, and global coordinates (fig. 14).

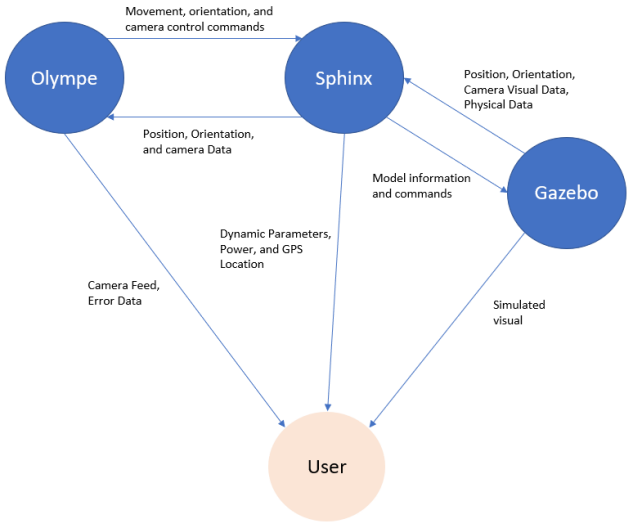


Figure 14: Diagram of Information Flow within Olympe Implementation

World Creation in Gazebo

World creation within Gazebo is carried out in a modular form; adding different pieces to the world until the world has all components which are required. These pieces take the form of models, files which define the three-dimensional objects which are to be placed in the world and all the relevant parameters which affect these items. Such parameters can include whether the object can move, if it is affected by environmental factors such as wind and gravity, and the texture of the object. For this project, worlds were generated in two ways. A simple world depicting cows in a field was developed by hand by introducing items into the world until all desired features had been included (fig. 15). Beginning with an empty world, a grass plane was added and then models of cows downloaded from 3dwarehouse.sketchup.com were inserted into the world and moved into their current positions (fig. 17).

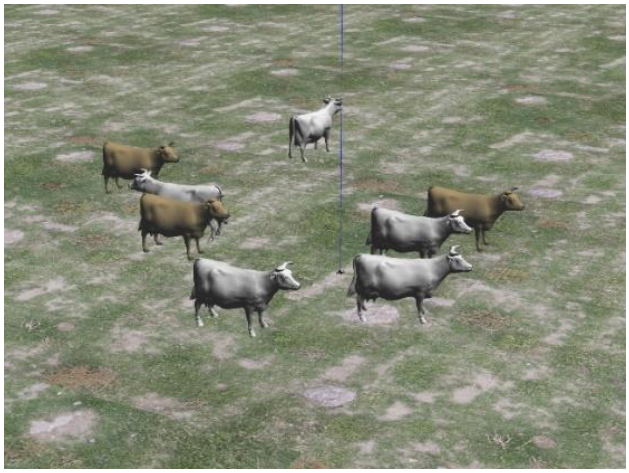


Figure 15: Image of World Populated with Cow Models

For the model depicting the WKU campus, a different method was used. Using a program called Open Street Viewer, a three-dimensional map was downloaded onto the computer. Here, unnecessary features such as sidewalks, side-streets, and features close to the edge of the map (beyond where the UAV was to be operated) were removed. This simplified the world file, enabling the simulation to be operated more smoothly on the machine while having minimal impact on the overall results (fig. 17). What remained after this process were models of all buildings on the WKU campus (fig. 16) in their proper locations and orientations without them needing to be input manually.

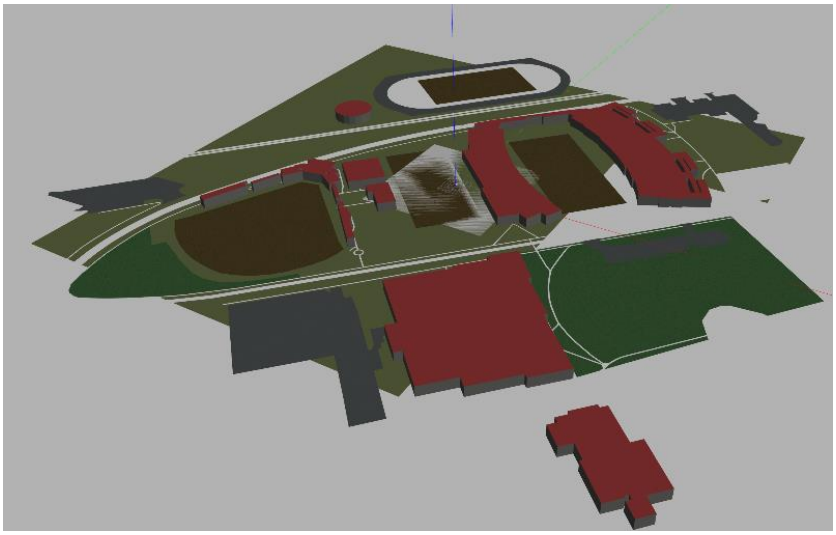


Figure 16: Section of Gazebo World of WKU Campus

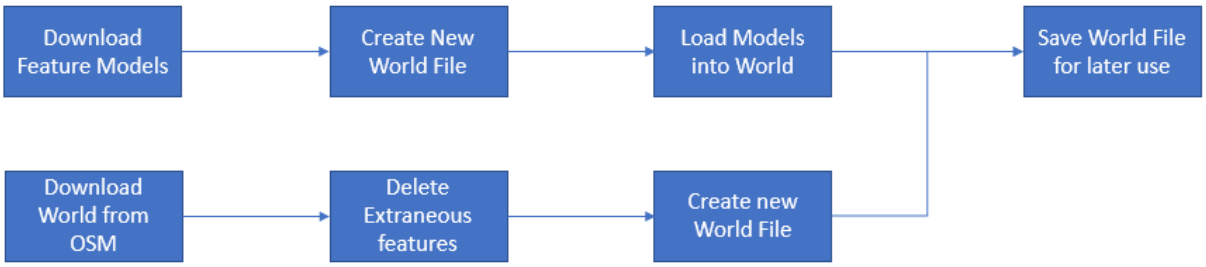


Figure 17: Diagram of World Creation in Gazebo

GPS Integration

One additional feature the Gazebo worlds implemented as opposed to the Simulink 3D worlds was the inclusion of the GPS coordinates for the world. This allowed for an additional level of path planning for the simulated UAV, navigation by GPS. Through the use of online maps, the coordinates for WKU were found and utilized, enabling the simulation of not just the buildings and layout, but also a potential form of navigation. In addition to the world, the online Sphinx Dashboard provided a map tool which utilized the UAV's simulated position to display the UAV and its flight path in their equivalent real-world location.

Flight and Process

Once the worlds were generated, the work progressed in two ways; creation of an algorithm to detect cows within the simulated world and navigation of the simulated WKU campus by GPS. The beginning for both outcomes was simple navigation using distance commands and determining how to control the camera on the UAV. After determining these controls and creating the WKU world, work began on developing a method to move to a specified GPS coordinate without the robot timing out. Soon afterwards, the video stream from the onboard UAV camera was analyzed and work began on the best parameters to detect the cows. This culminated in a system which could count the cattle from a predetermined height by finding the area of objects detected in the images.

Tool Selection

Once a working program was developed in both the Simulink and Olympe systems, the two systems were compared to assess which would be most beneficial to future development. The MATLAB implementation was noted to have a more transparent implementation of the control system. All subfunctions within the simulation can be opened, permitting the user to view any section of the model and modify the parameters if they wish. Additionally, MATLAB features a window where any variable involved with the simulation can be viewed, granting the user in-depth insight into the model of the system. This additionally permits efficient troubleshooting should the system behave in an

unexpected manner as by monitoring intermediate variables the location of the error can be swiftly determined and corrected. These benefits, however, came with the significant drawback that Mathworks had discontinued support for the Parrot UAV. Therefore, the hardware support package would not be updated for new product releases and no support would be given for connecting physical UAV's to the system, which would limit the range of applications for the system.

Conversely, the Olympe implementation was developed by Parrot, and planned to continue supporting their UAV's. Additionally, this system utilized the Gazebo simulation environment, a far more robust physics simulator than the Simulink 3D environment used by MATLAB. Though the effects of factors such as wind, air pressure, and gravity could be modelled in the MATLAB environment, the user would be required to manually include these as inputs to the airframe. In Gazebo on the other hand, these factors could be included in the world, and the simulator would account for these forces on any object (such as the UAV) which the user defined as being affected by them. This added functionality comes at the expense of additional complexity, as the Sphinx implementation involves multiple interacting systems, and errors in one subsystem may be caused by another program, increasing the complexity of troubleshooting.

Table 1: Comparison between MATLAB and Olympe

	MATLAB	Olympe
Pros	• Can view all system variables • All functions within the control system and dynamic model are readily accessible • More experience with the system	• More robust simulation system (Gazebo) which can simulate physical interactions • Open source • Has active support for the hardware
Cons	• The supported drone has been discontinued • Simulink 3D is a visualization platform, not a physics simulator	• Error handling occasionally lacking • Less experience with the program in the organization.

After evaluating the advantages of both implementations, the decision was made to continue with the Olympe implementation in future iterations of the project. The discontinuing of the drone utilized by the MATLAB system, though not an insurmountable obstacle, would have required extensive work to circumvent. As one of the primary goals of the project was to develop a system which future students could use for development, a supported platform was deemed as a critical component which could not be negotiated.

Navigation

One of the core subsystems which is required for the UAV to operate autonomously is the navigation system. Without this system, the UAV must be controlled by a human operator, necessitating an experienced user and thereby limiting the customer base who can utilize the system. With this system however, it might be possible for this system to be utilized to increase agricultural productivity across a larger percentage of the farming industry. Two primary methods have been explored to perform this navigation: navigation using predefined distances from the UAV's current location and GPS navigation.

Navigation by Predetermined Distances

The simplest manner of navigating with autonomous systems is by instructing the system to move a specific distance and direction from its present location. Though the principle remains the same, this is implemented in different ways between the two systems utilized with this navigation method. Within the Olympe system, this form of navigation is performed with the command:

moveBy(*dX*, *dY*, *dZ*, *dPsi*)

which moves the UAV to a location which is distances dX, dY, dZ relative to the UAV's front, right, and bottom respectively. Additionally, it allows the heading of the UAV to be rotated dPsi radians by the time it reaches the specified location.

Alternatively, the MATLAB system implemented this control not in the form of motion from the UAV's position at the beginning of the command, but from the initial launch point. Throughout the flight, the UAV determines its current position relative to the point it launched from and accepts movement commands in the form of a position vector relative to the original starting point (fig. 18).

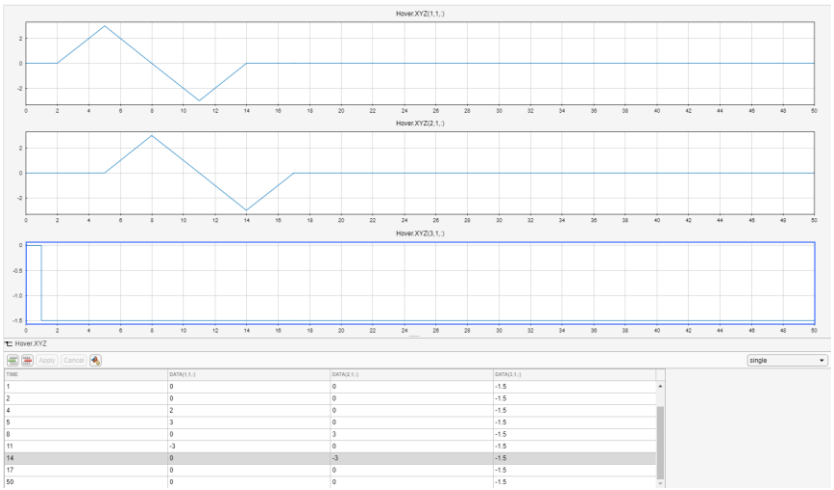


Figure 18: Signal editor for UAV path planning in MATLAB

This carries the benefit that unlike the Olympe version, this control scheme allows the user to define waypoints the UAV should visit rather than needing to define a list of directional movements. This is significant as in many remote scouting applications, this is how the operator is likely to conceptualize the path, as moving between points in the field (fig. 19).

Figure 19: Example UAV path https://bestdroneforthejob.com/drone-buying-guides/agriculture-drone-buyers-guide/

While this control scheme is relatively simple, both conceptually and in the sensors required to carry out the commands, it comes with a significant disadvantage. Due to the location of the UAV being determined internally from its motion, the UAV has no ability to correct itself should the prediction be inaccurate. This requires the UAV to be placed at the exact orientation and position for which it was programmed each time it is run, otherwise the positions will be shifted and turned by the offset from the expected value. For smaller applications, this might be within acceptable limits, for larger applications however, a more precise method might be required.

GPS Navigation

An alternative to determining position internally is to use the Global Positioning System (GPS) coordinates to define the waypoint locations. The process of defining how the UAV needs to move would be much like the MATLAB implementation described previously, except that the locations would be relative to the Earth as opposed to the UAV's launch point. This ensures the UAV's path will not be as significantly influenced by errors in location or orientation during placement at the beginning of the mission.

What is GPS?

The Global Positioning System is a network of satellites which allow a system with the proper receiver to determine its location on Earth. By connecting to these satellites, the GPS device will receive transmissions detailing the satellites' locations and the time they sent their transmissions. Using this information, the device can determine its displacement from these known points in space. After connecting to three or more of these satellites, these displacements are sufficient to define a singular location where the device must be located as illustrated in figure 20. This location can then be converted into the global latitude-longitude system; permitting navigation by this system if periodic updates are calculated during operation.

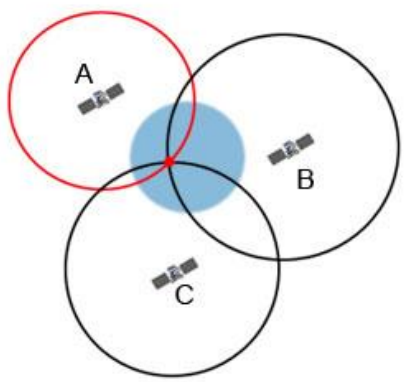


Figure 20: GPS location triangulation https://satmo.co.uk/latest/what-is-gps-tracking/

GPS Emulation

Within the Gazebo worlds used by the Sphinx and Olympe system to simulate the Parrot UAV, a latitude-longitude location can be specified for the center of the world. Using this process, navigation by GPS commands can be emulated within the world with the command:

moveTo(*latitude*, *longitude*, *altitude*, *orientation_mode*, *heading*)

which moves the UAV to the specified latitude-longitude location at the given altitude and heading. Initially, this line was sufficient, however it was eventually discovered that there was an upper limit to the timespan the UAV could move before the command timed out. This would trigger the virtual "ground base" to immediately send the next command to the UAV, believing the UAV was unable to execute the movement. This required additional work to keep the UAV moving to the given location without the system believing the UAV was unable to complete the task and therefore timing out. To accomplish this, a loop was implemented:

```
drone(    moveTo(36.985924,-86.455330, 10, 0, 0)).wait()
```

```
while(drone.get_state(FlyingStateChanged)["state"]is not FlyingStateChanged_State.hovering):

  drone.get_state(FlyingStateChanged)
```

This loop instructs the program to continue checking the state of the UAV until it has reached its destination, keeping the system from timing out before the UAV has reached its final location. This allowed later demos, such as the UAV flight from the Center for Energy Systems to the Houchen's Stadium to be completed without the system crashing.

Sphinx Dashboard Map

In addition to Gazebo worlds, Sphinx provides another method to simulate the motion of the UAV in the world; as well as provide critical information to the user during the virtual flight. By logging onto a system port during the simulation which was specified during the launch, a webpage will open, allowing the user to view graphs of system data – such as three-dimensional acceleration, velocity, and the UAV's current battery life. Additionally, a map widget can be opened, showing a street map with the UAV's present location marked (fig. 21). As the UAV moves, its path is drawn behind it. This facilitated GPS navigation development while the WKU campus map was being constructed.

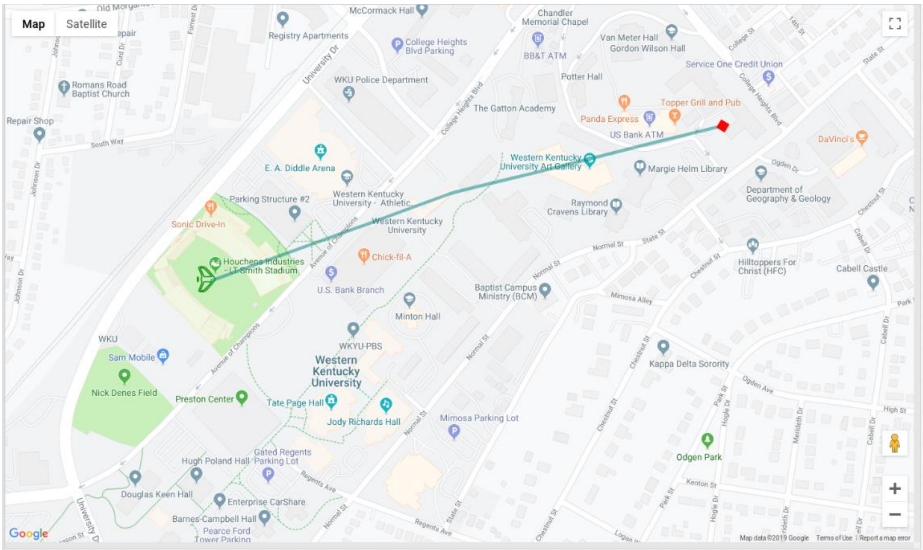


Figure 21: Map of the simulated GPS motion of the UAV in Sphinx

Chapter 4: Perception

The other critical subsystem for UAV sensing applications is the data acquisition and interpretation system. The process of extracting information from images is focused around finding patterns within the image which can be interpreted. Therefore, a key action in this process is in developing algorithms to eliminate unnecessary information and

interpret the remaining information. Broadly speaking, there are two methods to accomplish this task. Traditional methods which consist of defining rules for the computer to impose in the decision-making process and machine learning algorithms which determine the rules from pre-existing data.

What is Perception?

According to the Oxford English Dictionary, perception is "the ability to see, hear, or become aware of something through the senses." The ability for a system to receive information from its environment is only the first part of this process. After receiving the information, the system must be able to interpret the signals such that it can understand what was detected. The human brain is optimized for this process, being able to distinguish between a variety of objects without regard for orientation. For machines, however, the issue of perception is far from trivial; requiring extensive rules to be defined to perform the simplest of classification problems.

One of the first instances of machine perception was the robot Shakey developed by Stanford professor Charles Rosen. This system utilized a rule-based algorithm to interpret information obtained from its camera; permitting it to operate within a simple environment. Every image the robot sees is processed, simplifying the image into basic shapes, after which the resulting, simplified, image is classified using a set of rules determined by the designer before operation. While this rule-based system permits the efficient functioning of a real-time detection system, slight changes to the input conditions can result in incorrect classifications. One method of alleviating these rule-based detection issues is through the use of machine learning algorithms for feature detection. By training the network on images in all orientations and conditions, the system can attain a more general ruleset for object classification.

Two approaches were implemented to detect objects within a captured image. First, systems utilizing traditional, rule-based methods were implemented for both furrow and cattle detection and counting. These algorithms utilized a series of filters and intensity threshold functions to separate the desired objects from the background; enabling the detection of these objects. Additionally, machine learning techniques were implemented for image classification and cattle detection. These systems utilized a set of images to develop a set of parameters which could be used to classify images, permitting more complex classification systems to be developed, permitting these system to correctly operate on images with higher variability than traditional methods.

Image Processing

Traditional image processing methods consist of the programmer determining rules which can be used for analyzing the image and then writing a system to apply them. Such techniques include applying filters to the image to highlight specific features – such as vertical lines or edges between contrasting colors – and developing new images focused

on a single quality – such as the brightness of the pixels. Assuming well defined rules can be determined for this process, these methods can be applied to a variety of applications. In this project, these processes were applied to the detection of furrows between stands of crops and cattle in simulated fields.

Two programs were employed to perform image processing throughout this project. Within the MATLAB UAV algorithm, the Image Processing Toolbox was utilized to perform filtering, thresholding, and image transformation processes to detect features within the images. For the Olympe implementation, the OpenCV (Open Computer Vision) program was used. This is an open-source image processing system developed for use with computer vision and machine learning systems which includes algorithms for image preprocessing and object detection. Both performed comparably during testing, with the selection of the Olympe system for UAV development, however, it was decided to continue with image processing development using the OpenCV system due to it being an open-source program; making it more accessible to other students.

Furrow Detection and Counting

To assist with crop irrigation, furrows are often utilized within farmland. These channels between crop rows assist with guiding the flow of water through the field, as well as for flow control on sloped land. Due to the size of these features – in addition to their stark contrast with the crop rows – furrows were selected for the development of an initial feature detection and counting system within MATLAB. While the primary purpose of this application was to demonstrate the feature recognition process; practical applications for this information exist as well. After locating and counting furrows, this information could be matched with data on water stress within the crops to determine if the density and direction of the furrows are impacting crop growth. From this, decisions can be made on field management from this knowledge on water flow and saturation. For example, if the plants in one section of a field are water stressed, while another section downhill along the furrows is water-logged, it might indicate run-off is a significant problem. This could influence how treatment is applied to these sections of the field, and if severe enough, might suggest a change is needed to improve water flow within the field.

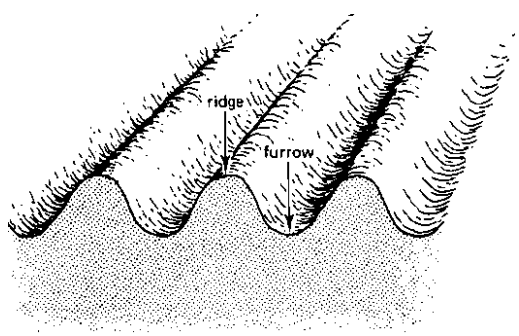


Figure 22: Diagram of furrows within a field , http://www.fao.org/3/S8684E/s8684e04.htm

Within the MATLAB simulation, this process is executed by the UAV as it flies over the simulated farmland. As the UAV traverses the field, furrows between stands of crops can be detected by searching for areas which form straight lines across the image. The methodology to detect these lines is through the application of a high-pass filter. This will result in an output image in which areas of dramatic change between pixels will be shown clearly while areas of gradual change will not appear. This filter was implemented through the process of 2-dimensional convolution between the image taken from the UAV and the filter matrix:

$$filter = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

This process is accomplished through applying the convolutional filter matrix to sections of the input matrix and performing a two-step operation within the bounds of the input matrix. First, elementwise multiplication is performed over the relevant locations within the input matrix and the filter followed by summing the results of this multiplication. This value is then placed into the appropriate location of the output matrix and the process repeats for a new section of the input (fig. 22).

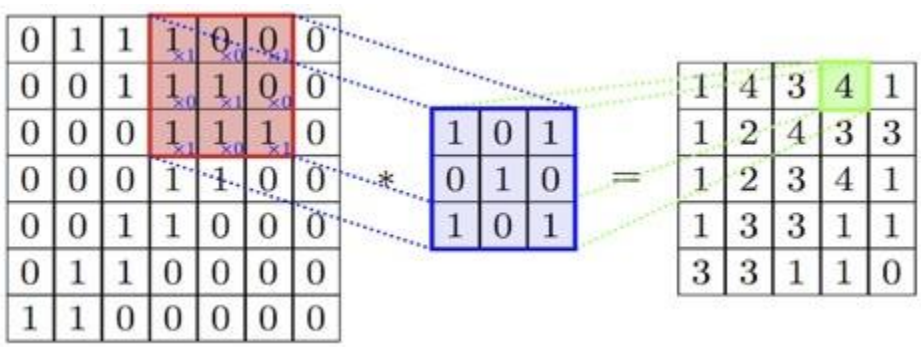


Figure 23: Example of convolutional filter, https://insuranalytics.ai/general/different-kinds-convolutional-filters/

By utilizing this filter, vertical lines within the image are emphasized; the direction the furrows will be oriented within the video stream when the UAV is flying perpendicular to them. This permits more focused algorithms to be developed for feature detection. After using this filter, the image was gray-scaled and had its contrast enhanced, highlighting the detected rows. The center columns of this image were then analyzed, with the average value of the area being taken. If the average was beyond a certain threshold, it could be assumed the UAV had passed over a furrow, which was subsequently counted (fig. 23). The total values for each pass were recorded and their median taken to produce an estimate of the final number of rows the UAV surveyed. The explored tools included libraries to perform these operations behind the scenes; freeing users to focus on tuning filters to solve detection problems rather than implementing matrix multiplication.

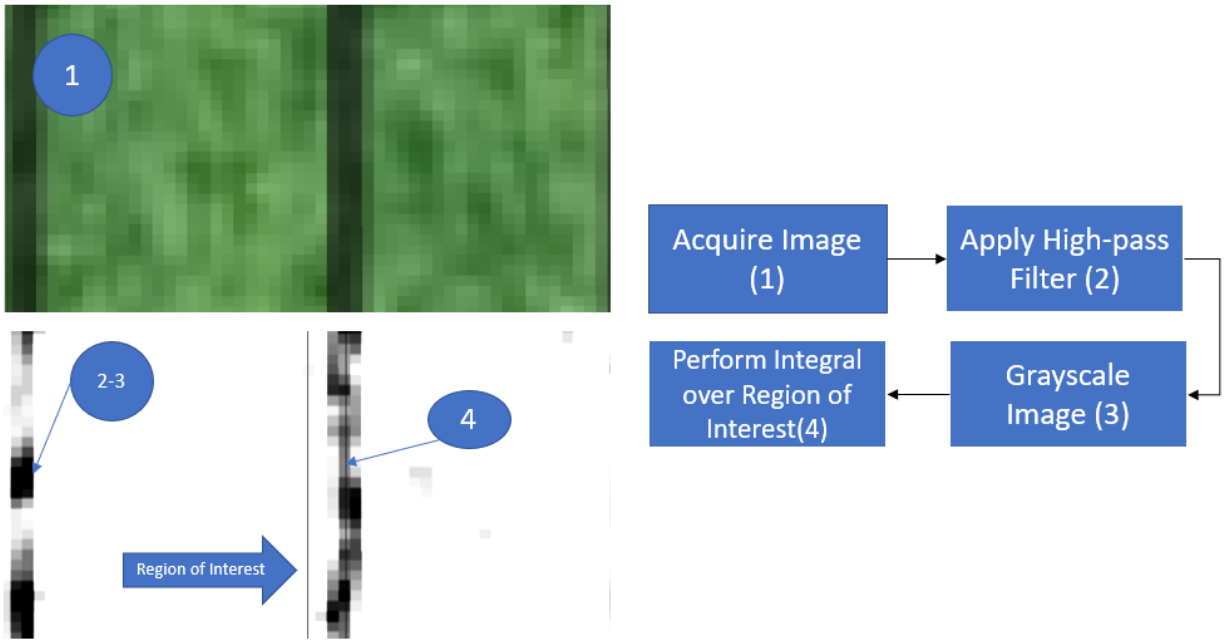


Figure 24: Diagram of Furrow Detection in MATLAB

Bovine Inventory System

Within both the MATLAB and OpenCV systems, processes for determining the number of cows contained in a simulated field were implemented. Both utilized a method of comparing the pixels of a gray-scaled image against a threshold to differentiate between areas which contained cows and those which did not. Once these regions were generated, a second algorithm calculated the properties of these regions: centers, areas, and circularity. This enabled the regions to be categorized as to whether they met the standards which identified the area as a cow.

Threshold Math

Thresholding describes a group of algorithms which compare pixel magnitudes with a given value and then change the pixel's magnitude depending on its relationship to that value. For the Bovine Inventory Algorithms, the method used is referred to as a Threshold Binary method (fig. 24). As shown below, if the magnitude of a pixels is above the given threshold, the system sets that pixel to the maximum value whereas all pixels below are set to 0.

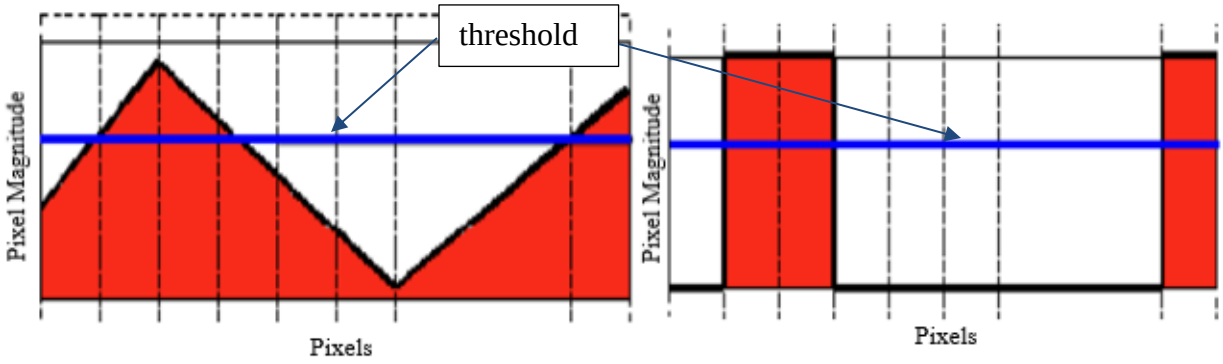


Figure 25: Example of Binary Thresholding https://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html

This method produces an image whose only features are the locations of regions above the threshold. To eliminate every region except for the cows within the image, filters were applied to remove noise. For the MATLAB system, the filtering occurs after the thresholding. The filter fills in regions with areas below a certain value, removing extraneous details and leaving only the regions representing cows in the final image (fig. 25).

Figure 26: Filtering of binary image in MATLAB

The OpenCV algorithm is instead filtered ahead of time; blurring all items within the image and leaving only the cows as recognizable objects (fig. 26). Therefore, when the image has thresholds applied, only the cows remain visible.

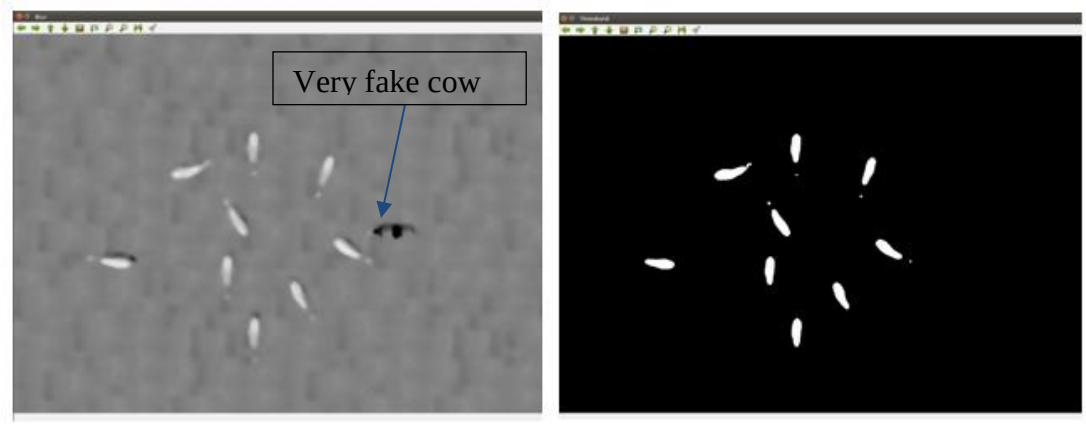


Figure 27: Binary threshold of filtered image in OpenCV

Blob Detection

After the image has had a threshold applied and been filtered, the system begins the process of detecting blobs within the image. Blob detection is the process of finding areas of the image, in these cases white pixels, which are connected into a contiguous region. These shapes can then be screened by a variety of factors (fig. 27) such as area, circularity (how close the shape is to being a circle), and inertia (the ratio between the minor and major axes).

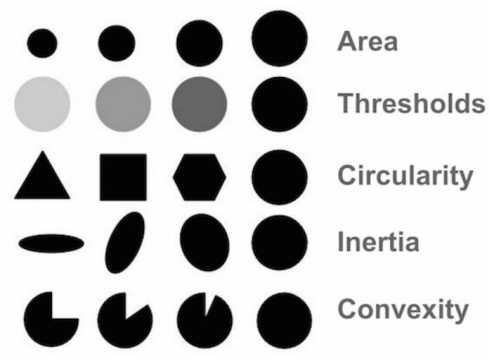


Figure 28: Examples of blob parameters https://www.learnopencv.com/blob-detection-using-opencv-python-c/

After repeated trials with changing the blob detection parameters, the correct set of rules can be determined which allows the system to correctly locate and count the number of cows in the image. In both the OpenCV and the MATLAB implementations (fig. 28) the system can correctly identify the realistic cows within the image while ignoring the fake cow.

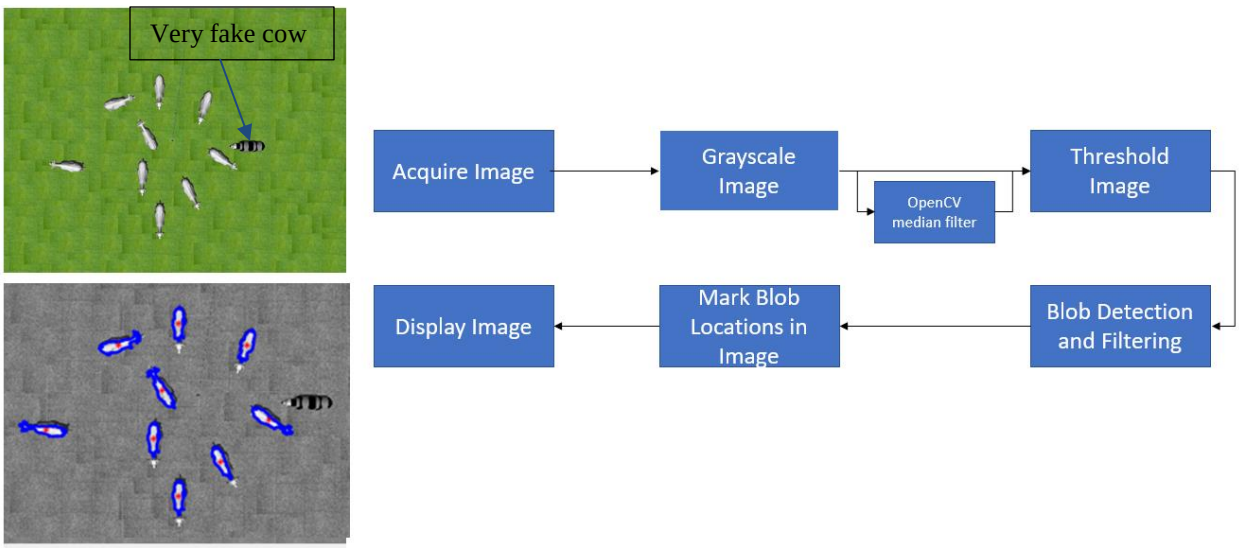


Figure 29: Diagram of Blob Detection Methods in OpenCV and MATLAB

Within the Olympe implementation, this functionality was extending to performing real-time object detection during simulated fights. During the flight, the video stream from the UAV was routed to the OpenCV analysis script before being displayed. This permitted the system to perform the previously detailed processing functions – such as applying filters, thresholding operations, and blob detection – as the UAV was flying. This processed data was then displayed onscreen to the user, indicating the locations of cows within the frame. In a production system, on the other hand, this could be utilized to determine if another function should be performed – spraying crops, incrementing an animal inventory count, or applying pesticides for instance.

This success, however, is limited. The rules to detect these objects are specific to the images they have been designed for; any change to the situation would result in unsuccessful results. If during the in-flight detection system, for example, the height of the UAV was to change or either the cows or background were to be a different color, the system might not produce accurate results. For this reason, a more robust method would be required in an actual application, leading to the decision to investigate machine learning algorithms.

Machine Learning

In order to remove the necessity for a user to determine the appropriate parameters for each image taken by a UAV; development began on a machine learning algorithm to perform this detection. Unlike previously discussed method of identifying objects in an image, the system is not given a set of properties to filter by and functions to perform in order to detect objects within an image. Instead, the machine determines these features itself from a series of training images with the appropriate classifications given. This method was utilized through multiple stages of development to create a system able to

differentiate between bovines and other objects; eventually culminating in a system which could differentiate between images of cattle, cars, and airplanes.

Fundamentals

The core objective of a machine learning algorithm is for the computer to discern underlying patterns within a set of data; thereby allowing it to correctly interpret future, similar, data. This process is motivated by two core reasons. Once the algorithm's structure and set of underlying data classification rules, the model, have been developed, the finished product can be deployed as-built on other machines for similar applications with comparably less time and cost. Additionally, this method can discern patterns within data which might be tedious, conceptually complex or computationally intensive to implement in a traditional manner.

Within machine learning algorithms, there are a variety of common methods which can be leveraged to develop these models; including clustering, Bayesian models, decision trees, and artificial neural networks [3]. Within the scope of this project, the development was focused on the use of neural networks. These systems are designed to mimic the behavior of a brain; with an architecture made of layers of "neurons", called nodes, each of which sends a signal to the next layer based on a function of its inputs. Within a neural network, these layers fit into three broad categories: the input layer which feeds the initial data into the network, the output layer which gives the result determined by the network, and any number of hidden layers which perform operations on the data (fig. 29).

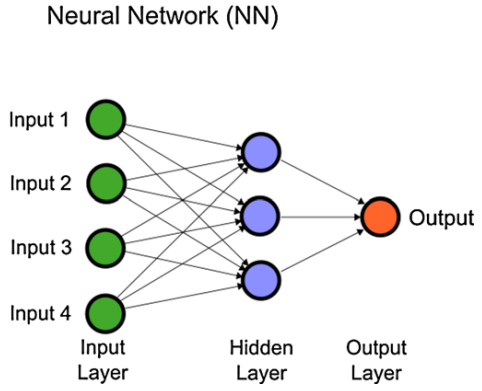


Figure 30: Example of a Simple Neural Network [16]

The size and number of these layers depend on the application for which the system has been developed, ranging from a single layer in some simpler applications to several layers for more complex operations. For input and output layers, the requirements are well defined according to the application. Input layers are required to have a number of nodes equal to the datapoints which will be entered for analysis (e.g. the number of pixels in an image) while the output nodes must be equivalent to the number of labels into which the data is being classified.

For the implementation of the machine learning algorithm, a more complex form of neural network, the convolutional neural network (CNN), was utilized. In addition to the layers described previously, CNN's utilize additional functions to assist in processing the data prior to analysis. The first of these functions is the titular convolution, which applies a filtering matrix to the input by performing element-wise multiplication to a section of the input matrix and summing the results into one element of an output matrix. This function transforms the input matrix into a smaller matrix, lowering the amount of required input nodes, and thereby the complexity of the required model.

After the convolution function is applied to the data, the result processed through the second function, a pooling layer. This function analyzes a section of the input matrix and outputs a value based upon the inspected area into the output matrix. In the application developed for this project, a maximum pooling algorithm was utilized, outputting the maximum of the analyzed region (fig. 30). This combination of functions, convolution and max-pooling, serves to reduce the number of weights which must be fitted to the data, while preserving the key features of the data, permitting classification of a variety of data with reasonable accuracy.

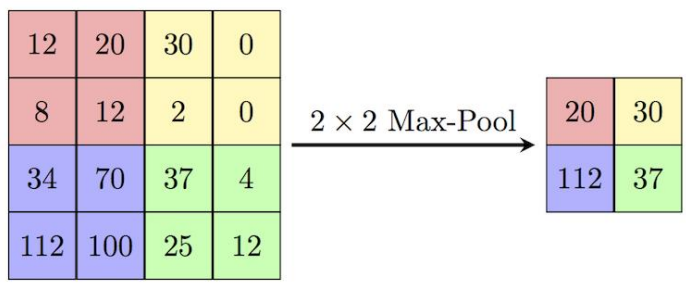


Figure 31: Maximum Pooling Example, https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

These additional processes in CNN's result in increased performance compared to other neural networks. While non-convolutional neural networks experience a performance ceiling after they have been trained on a certain amount of data, for CNN's this value is much higher; permitting better performance to be constructed from the wealth of data available today. For this reason, this project will focus on CNN's.

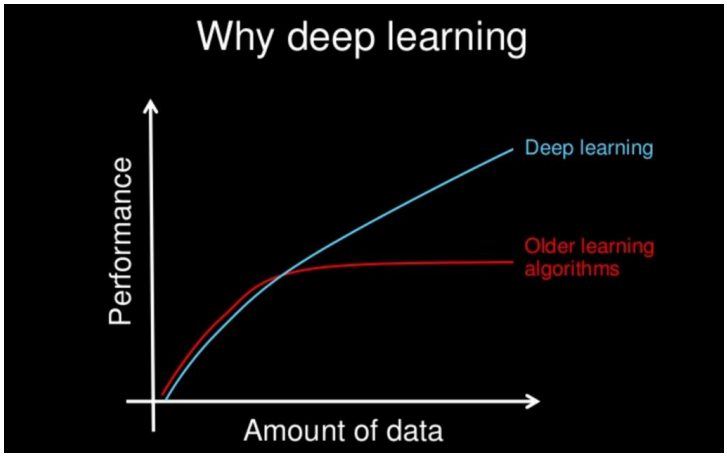


Figure 32:Performance graph of deep learning versus other learning algorithms, https://machinelearningmastery.com/what-is-deep-learning/

Operation of Machine Learning System

This system was implemented through the program Tensorflow with a Keras front end. Tensorflow is a neural network development system created by Google which consists of a set of functions, libraries, and resources enabling the construction of neural networks. Through these functions, the process of setting up the neural network is accelerated as the system can define neurons programmatically using inputs from the user about layers which need to be created. This permits the system to be developed at the layer level, drastically improving the speed at which these programs can be developed. Additionally, these functions were accessed through the Keras API, a high-level interface which further simplifies the design by using common parameter settings. Tensorflow is an adaptable interface, within which custom activation functions and complex systems distributed across multiple processors can be developed. The Keras API removes this complexity using common design parameters, lowering the skill floor required to develop these systems. Though this removed some of the customizability of the system, the time required to build the network was lowered to a manageable level, allowing development to move into the training phase.

Training of the neural network is performed using large, labelled, datasets which are fed to the network and classified (fig. 31). After each classification is performed, the result is compared to the correct label, and the weights within the network are changed according to the success of the system. Once all training data has been analyzed, a second, smaller set of data is utilized to test the success of the system at categorizing novel data. This process is carried out iteratively until the system has completed a specified number of training periods or has achieved a desired accuracy. The selection of this desired number of iterations or target accuracy is crucial to maintain a balance between accuracy and generalized results. If the system is not trained long enough, the system will be underfitted and the results may not be at the accuracy level required by the application it has been designed for. Alternatively, if the system is trained for too long the training can result in overfitting. In this case the model achieves incredible accuracy in predicting the training

set, at the cost of noticeably lower accuracy for data outside this training set. Since in the majority of applications the system is intended to be used on novel data, overfitting is generally undesirable.

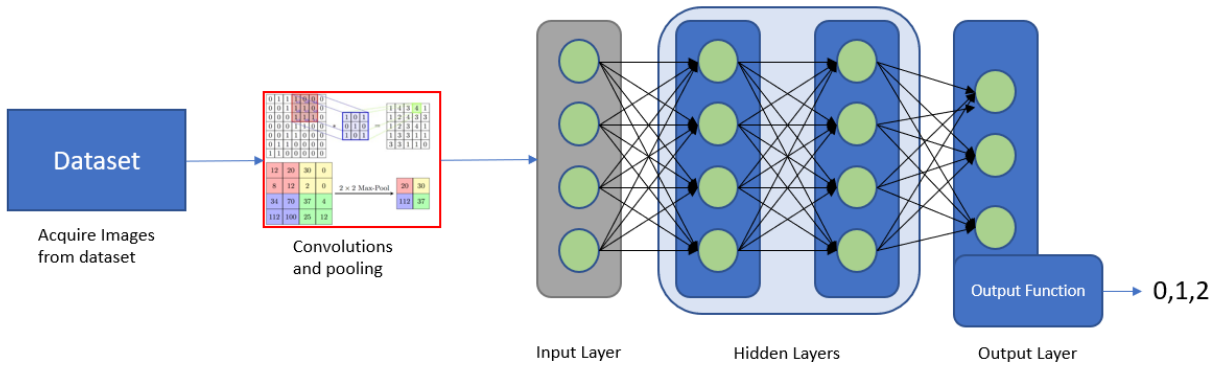


Figure 33: Image Classification with Convolutional Neural Network

Binary Detection

The process of developing an image classifying neural network began with the exploration of existing demonstrations and systems. The first explored were Tensorflow demonstrations and tutorials created by Google, the owners of the system. These demonstrations provided a platform to learn about the process of training a neural network, how increased training cycles increase the accuracy of the system, and the system behavior of an overfitted model. In the Google tutorials, however, the model creation and utilized dataset were pre-generated by the software architects; ensuring the tutorials functioned but providing little to no experience in designing a working model or dataset. For this, another demonstration was found on www.geeksforgeeks.org/python-image-classification-using-keras/ which ran on the local machine and permitted the user to explore the model architecture and structure of the dataset. Using this, experience was obtained in designing the system and reshaping data to be evaluated by the system (e.g. resizing images to the expected dimensions).

Real and Fake Bovine Classifier

From this experience, the first classification system to be implemented was a system to distinguish between images of real and fake bovines. For this application, 115 images of cattle models, both realistic models and clearly unrealistic models, were collected from 3dwarehouse.com, a repository for three-dimensional simulation models. For each of these objects, images were captured of the object from various orientations. Images of these models were chosen for testing in lieu of images of actual cows in the interest of creating a network which could be applied to the simulated UAV flights to classify models detected by the UAV's camera. Once collected, the images were

categorized into “real” and “fake” training subsets and a percentage of the images were separated to be used for validation (fig. 32).

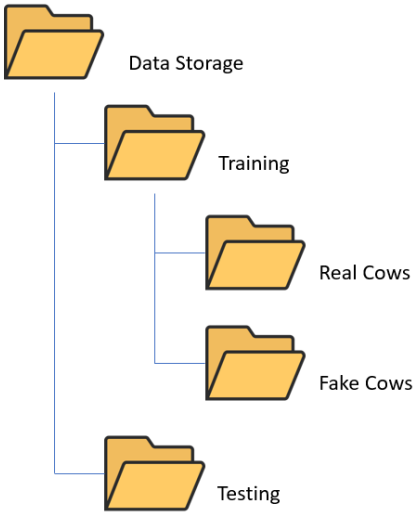


Figure 34: Image Storage Hierarchy

From here, training commenced on the dataset and the results were analyzed through multiple training operation with increasing numbers of training periods. Throughout the process, the system was unable to correctly identify the training images when requested (fig. 33), even when the training indicated accuracies in excess of 98%. After reconsidering the conditions of the dataset, it was concluded that the subsets of data were too similar, leaving the system unable to derive a clear set of rules to distinguish between the data subsets.

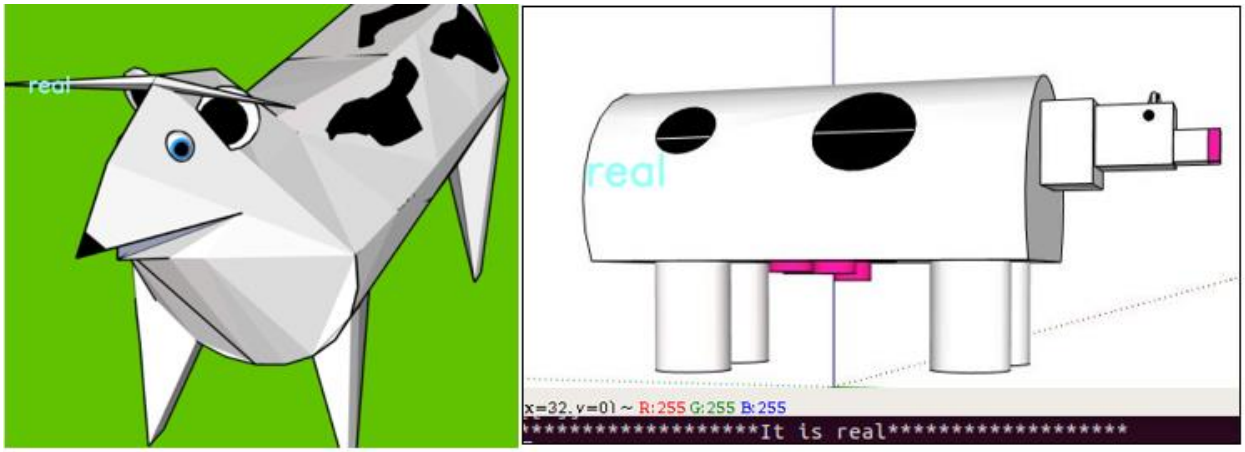


Figure 35: Example of Incorrectly Labelled Images

Bovine Image Detector

In order to provide the system with a set of categories which were clearly differentiable, the "real" and "fake" cattle images were combined into a single set of images and placed into the dataset from the www.geeksforgeeks.org CNN demo. As before, training was performed using the created dataset with the intention that the machine could distinguish between the three categories of images. After training, the system was unsuccessful at image classification in both training – achieving only 35% accuracy – and in manual tests (fig. 34). Upon inspection of the network, it was determined that the output layer of the system was a single node with a binary output. Therefore, the system could only determine if an image belonged to the first class of images within the dataset, allowing it to function for a two-class problem, but not for three. To avoid this issue and verify the bovine dataset was sufficient for training, the subset of data featuring planes was removed and the model was re-trained to distinguish automobiles from cows. In this implementation, the system was able to perform the classification successfully (fig. 35).

Figure 36: Incorrectly Labelled Images from Initial Binary Classification

Figure 37: Correctly Classified Images from Binary Network

Multi-output classification

After confirming the system could successfully distinguish between bovines and automobiles, the CNN was changed to accommodate multiple categories. This entailed changing the output layer of the network to having three nodes, one for each category, and

changing the function in the final layer from a sigmoid activation function (binary output) to a softmax function (fig. 36). This generates a decimal probability for each class proportional to the value of the node, then sends the number of the node with the highest probability as an output. This allows the system to distinguish between multiple classes, permitting the plane subset to be included before the network was trained and tested.

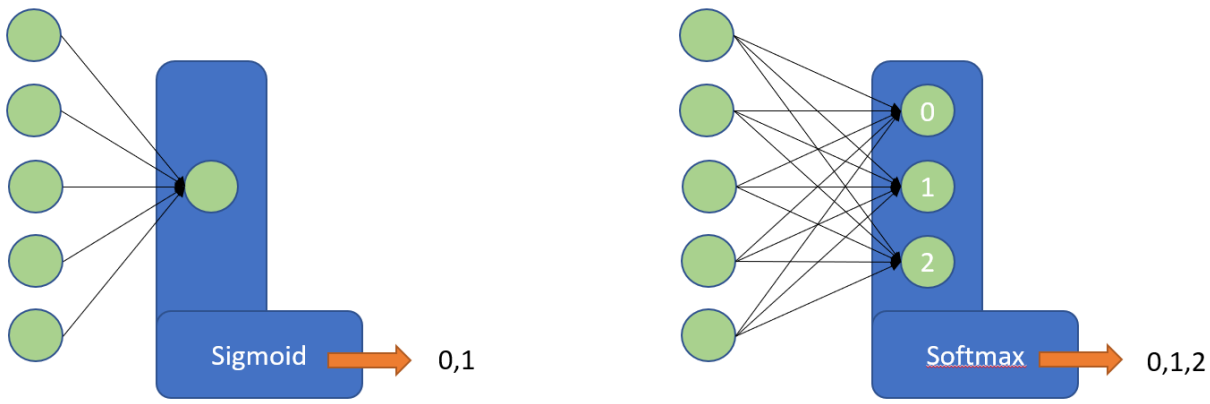


Figure 38: Change in the Architecture to Facilitate Multiple Classifications

Figure 39: Correctly Classified Images from Multi-Category Classifier

After training, the accuracy of the system was confirmed; reaching 98% during training. To fully affirm the accuracy of the network, truly novel images were given to the network to classify, and similar accuracy was observed (fig. 37).

Functional Diagram for Full System Integration

Once all these systems were developed, they could be combined to form a single, composite, system capable of scouting over a field, and locating a desired feature (e.g. cattle, areas needing irrigation, etc.) within the images captured (fig. 38). The system would then perform some operation should the feature be detected (e.g. inventory of the cattle, designating that the area required water, etc.) performing no operation should the feature not be detected. The operation begins by training the neural network to distinguish between

images containing these features and those which do not. This process would be carried out iteratively, repeated as more images are gathered to increase the accuracy of the detection algorithm. Simultaneously, the mission would be defined, specifying the area the UAV would traverse, the data it would be required to collect, and if any operation should be carried out should a specified feature be identified within the data. These parameters would then serve as the guidelines for constructing the path the UAV will traverse and what hardware (sensors, additional storage or processors, chemical applicators, etc.) will be required by the system. Once this system has been developed, it can be deployed on the UAV.

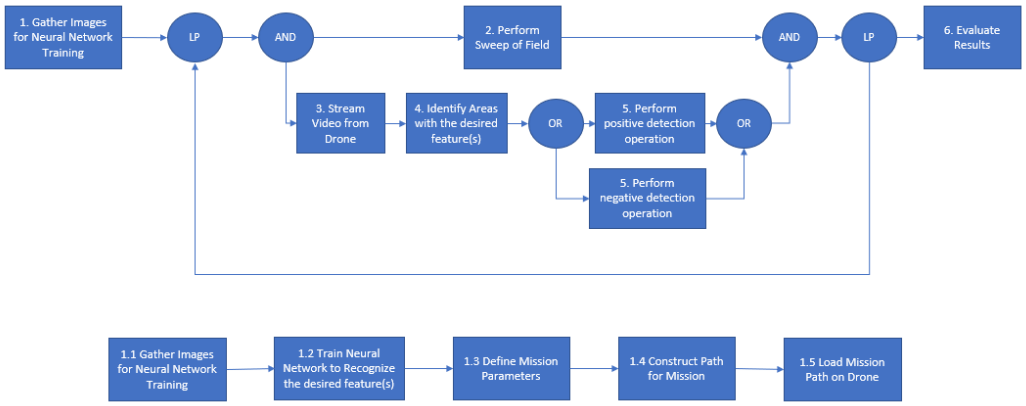


Figure 40: Block Diagram Representation of the mission execution process.

With the system deployed, the UAV will proceed to carry out its mission. This entails the operation of two parallel processes, the flight of the UAV over the field, and the acquisition of data. As the UAV traverses the field, the onboard sensors collect data about the UAV's location and the conditions of the section of field over which the UAV is located. This allows the UAV to accurately follow the path defined for the mission, ensuring the collected field data is referenced to the proper location or treatments are applied in the correct regions. Throughout this flight, limited processing of the data can be carried out to select between operations to perform, including counting features within the field or applying a treatment to a section of field. Once the flight is completed, further, more intensive, processing can be performed on the collected data; using processes which are not feasible for real-time analysis (e.g. resource intensive operations which require more time, processing power, or on-hand data than is available during operation). This analysis can then be used to refine the neural networks and guide the definition of future missions.

Future Work

From this work, future development can continue through several avenues. To begin, the current machine learning algorithm will be further refined to improve the performance of its current functions and add new functions to the system. The projected

next development for this system is a process to implement the detection, locating, and summation of objects detected using the current algorithms. As with the system using traditional image processing, this would enable inventory and feature detection systems using the image of a field. The system could then be expanded with improved data to provide more detailed information of the detected objects. For instance, the bovine detection system could be adapted to provide information on the breeds of cows or if any of them were showing signs of poor health. Though this function would require more data than is presently available to the system to differentiate between similar objects; the benefit to the user would be greatly increased. With sufficient information, a combination of these systems could be generated, permitting the location, number, and status of a herd of cattle to be evaluated.

In addition to further development in the classification algorithm, the system will be implemented on a physical UAV. Due to the existing models within simulation, the current algorithms can be transferred with only minor changes to account for variation in the physical motors. Implementing the system on a physical UAV will confirm its functionality and allow the system to be tested under real flight conditions. As the system matures, it can be applied to an increasing number of operations, potentially through cooperative projects with other departments. A functional classification system and aerial platform could be utilized for an array of potential projects, enabling the work to be efficiently utilized through its multi-faceted application.

The penultimate goal, however, is to create a smart agriculture system which can increase the productivity of arable land. Ideally, this system could be utilized to provide value to the state of Kentucky. Though agriculture comprises only a small percentage of the state gross domestic product (GDP) – 1.1% according to a 2018 study by the University of Kentucky – it is critical to local and regional economies [17]. For this reason, an increase in the efficiency of production would be beneficial both to local and broader markets.

Conclusion

Through the use of UAV's and related technologies in the processes of crop monitoring, data analysis, and crop management; the agricultural process can be made more efficient. By continually monitoring crops with UAV mounted cameras and spectrometers, problems such as weeds, pests, and nutrient stress on the plants can be observed, located, and treatment prescribed. These treatments can then be carried out by the same UAV's which performed the scouting, applying herbicides, pesticides, and fertilizers in the specific area of the detected issue; reducing resource usage and environmental impact. The effects of these treatments can then be monitored to gauge whether treatments should continue, new treatments prescribed, or if the issue has been resolved. This data can then be applied to make such treatments more efficient in the future, allowing the output from a single field to be increased while the resources required to implement this increase in production decrease, or at minimum, experience a slower rate of increase.

To this end, evaluation of tools which could be used develop a system to perform these functions proceeded in three sectors. Initially, the potential use of MATLAB and Olympe for development of the navigation subsystem was explored using UAV models within simulated environments. These simulations were then expanded to facilitate the use of image analysis with collected images. For this application, the MATLAB Image Processing Toolbox and OpenCV programs were evaluated for their ability to implement feature detection. To further increase the feature detection potential of these systems, machine learning algorithms were developed with Keras and Tensorflow. This facilitated the classification of items within an image set into both 2 and 3 categories. From this analysis, the Olympe system using OpenCV and Tensorflow for image processing and feature detection was selected for use in the development of future autonomous agricultural solutions.

These systems stand to be a potential revolution in agriculture, with an impact much like the Industrial Revolution. A 2013 study predicted "Every year that [UAV] integration is delayed, the United States loses more than $10 billion in potential economic impact." [18] Those willing to implement this technology, and improve upon it, could potentially increase their efficiency and output, potentially overtaking competitors who do not. As there are significant initial costs in implementing these systems, and management techniques increase in efficiency as more data is gathered on the field, the barrier to be competitive will only grow. An economic report published by the University of Kentucky stated, "The family farm has become a quaint ghost of Kentucky's past…. roughly one-third as many farms exist today as there were in 1950, while the average size of Kentucky's farms has doubled." [17] At the same time, the employment in farm work has dropped from 11 to 4 percent [17]. If we assume this is the common trend, successful farms today are large, industrialized operations, so to the farms in the future might be smart, automated ones. In order to remain competitive, this change needs to be embraced.

References

[1] L. A. Nicol and C. J. Nicol, "Adoption of Precision Agriculture to Reduce Inputs, Enhance Sustainability and Increase Food Production: A Study of Southern Alberta, Canada," *WIT Transactions on Ecology and the Environment*, vol. 217, pp. 327-336, 2019.

[2] E. R. Hunt and C. S. T. Daughtery, "What good are unmanned aircraft systems for agricultural remote sensing and precision agriculture?," *International Journal of Remote Sensing*, vol. 39, pp. 5345-5376, 2018.

[3] K. Liakos, P. Busato, D. Moshou, S. Pearson and D. Bochtis, "Machine Learning in Agriculture: A Review," *Sensors*, vol. 18, no. 8, pp. 1-29, 2018.

[4] P. Psirofonia, V. Samaritakis, P. Eliopoulos and I. Potamitis, "Use of Unmanned Aerial Vehicles for Agricultural Applications with Emphasis on Crop Protection: Three Novel Case-studies," *International Journal of Agricultural Science and Technology,* vol. 5, no. 1, pp. 30-39, 2017.

[5] X. Pantazi, D. Moshou, T. Alexandridis, R. Whetton and M. A.M., "Wheat yield prediction using machine learning and advanced sensing techniques," *Computers and Electronics in Agriculture,* vol. 121, pp. 57-65, 2016.

[6] C. Andrew, "Types of Drones: Multi-Rotor vs Fixed-Wing vs Single Rotor vs Hybrid VTOL," *DRONE,* June 2016.

[7] C. Zhang and K. J. M, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision Agriculture,* vol. 13, no. 6, pp. 693-712, 2012.

[8] D. I. Patricio and R. Reider, "Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review," *Computers and Electronic in Agriculture,* vol. 153, pp. 69-81, 2018.

[9] L. Deng, Z. Mao, X. Li, Z. Hu, F. Duan and Y. Yan, "UAV-based multispectral remote sensing for precision agriculture: A comaprison between different cameras," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 146, pp. 124-136, 2018.

[10] A. Nixon, "Best Drones For Agriculture 2019: The Ultimate Buyer's Guide," https://bestdroneforthejob.com/drone-buying-guides/agriculture-drone-buyers-guide/, 2019.

[11] Sensefly, "eBee SQ The Advanced Agriculture Drone," https://www.sensefly.com/drone/ebee-sq-agriculture-drone/, 2019.

[12] PrecisionHawk, "Drones for Agriculture," https://www.precisionhawk.com/agriculture/drones, 2019.

[13] DJI, "AGRAS MG-1," https://www.dji.com/mg-1/info#specs, 2019.

[14] L. Geiver, "Survey shows drone adoption rate by farmers," *UAS Magazine,* 31 July 2018.

[15] T. Luukkonen, "Modelling and control of quadcopter," Independent research project in applied mathematics, Espoo, 2011.

[16] J. Behmann, A.-K. Mahlein, T. Rumpf, C. Romer and L. Plumer, "A review of advanced machine learning methods for the detection of biotic stress in precision crop protection," *Precision Agric*, vol. 16, pp. 239-260, 2015.

[17] C. R. Bollinger, W. H. Hoyt, D. Blackwell and M. T. Childress, "Kentucky Annual Economic Report 2018," University of Kentucky, 2018.

[18] D. Jenkins and B. Vasigh, "The Economic Impact of Unmanned Aircraft Systems Integration in the United States," Association for Unmanned Vehicle Systems International, 2013.

Appendix I: Software Utilized within Development

Mathworks Products

MATLAB	Version 9.6
Simulink	Version 9.3
Aerospace Blockset	Version 4.1
Aerospace Toolbox	Version 3.1
Simulink 3D Animation	Version 8.2
Image Processing Toolbox	Version 10.4
Control System Toolbox	Version 10.6
Signal Processing Toolbox	Version 8.2
Optimization Toolbox	Version 8.3

Ubuntu System

Python 3
Gazebo 3
Olympe
Sphinx
OpenCV
Keras
Tensorflow

Appendix II: Select Code Utilized within the Project

Drone Movement

```
# 1 degree ~= 111km

# connect to drone
drone = olympe.Drone("10.202.0.1")
drone.connection()

# drone takeoff
drone(TakeOff()>> FlyingStateChanged(state="hovering", _timeout=5)).wait()

# send command to move to first coordinate
drone(  moveTo(36.985924,-86.455330, 10, 0, 0)).wait()

# keep moving until the drone arrives at the first waypoint
while (drone.get_state(FlyingStateChanged)["state"] is not FlyingStateChanged_State.hovering):
   drone.get_state(FlyingStateChanged)

# move to second waypoint
drone(  moveTo(36.984732,-86.459452, 10, 0, 0)  >> FlyingStateChanged(state="hovering",
_timeout=100)).wait()
while (drone.get_state(FlyingStateChanged)["state"] is not FlyingStateChanged_State.hovering):
    drone.get_state(FlyingStateChanged)

# Landing
drone(Landing()).wait()
print ('landing')
drone.disconnection()
```

Cow Detection from Images

```
# Read image and Apply bluring
im = cv2.imread("CowsNewGrass2.jpg", cv2.IMREAD_GRAYSCALE)

blur = cv2.medianBlur(im, 21)
cv2.imshow("Blur",blur)
cv2.waitKey(0)

# Apply Thresholding
retval,binar = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU);
cv2.imshow("Threshold",binar)
cv2.waitKey(0)

# Set detector parameters
params = cv2.SimpleBlobDetector_Params()

# Change thresholds
params.blobColor=255;

# Filter by Area.
params.filterByArea = True
params.minArea = 300

# Filter by Circularity
params.filterByCircularity = False
params.minCircularity = 0.1

# Filter by Convexity
params.filterByConvexity = False
params.minConvexity = 0.87

# Filter by Inertia
params.filterByInertia = False
params.minInertiaRatio = 0.01

# Create a detector with the parameters
detector = cv2.SimpleBlobDetector_create(params)

# Detect blobs.
keypoints = detector.detect(binar)

# Draw detected blobs as red circles.
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS ensures
# the size of the circle corresponds to the size of blob

im_with_keypoints = cv2.drawKeypoints(im, keypoints, np.array([]), (0,0,255),
cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

print(keypoints)

# Show blobs
cv2.imshow("Keypoints", im_with_keypoints)
```

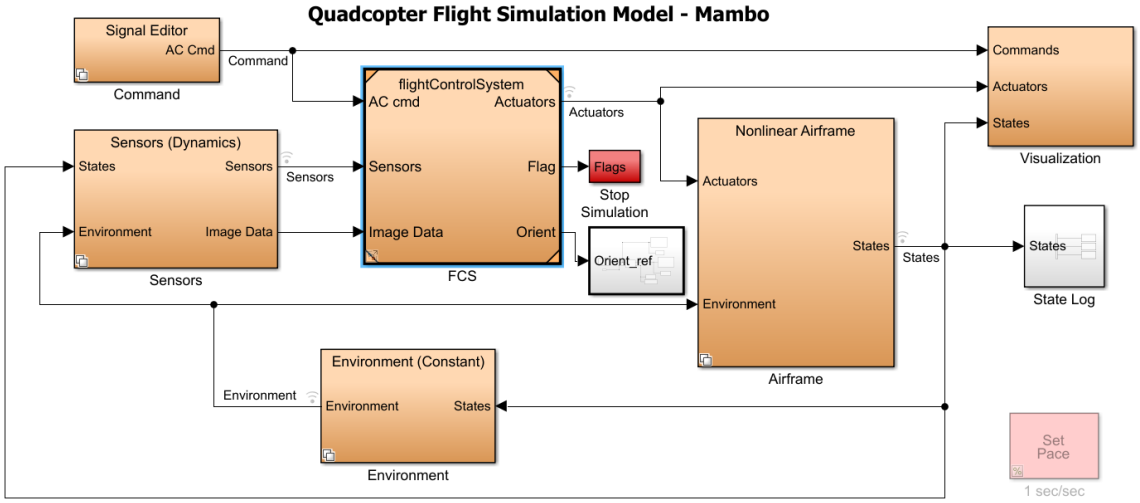
Machine Training

```
# define image input shape
img_width = 224
img_height = 224

if K.image_data_format() == 'channels_first':
        input_shape = (3, img_width, img_height)
else:
        input_shape = (img_width, img_height, 3)

# define layers of neural network
model = Sequential()
model.add(Conv2D(32, (2, 2), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(3))
model.add(Activation('softmax'))

print('**************************** Compilation ****************************')

# defing model parameters
model.compile(loss='sparse_categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

# define training image input variation
train_datagen = ImageDataGenerator(rescale=1. / 255, shear_range=0.2,
        zoom_range=0.2, horizontal_flip=True, vertical_flip=True)

# define test data input parameters
test_datagen = ImageDataGenerator(rescale=1. / 255)

# define training model
train_generator = train_datagen.flow_from_directory( train_data_dir,
        target_size=(img_width, img_height), batch_size=batch_size, class_mode='sparse')

# define validation model
validation_generator = test_datagen.flow_from_directory(validation_data_dir,
        target_size=(img_width, img_height), batch_size=batch_size, class_mode='sparse')

# train model
model.fit_generator(train_generator, steps_per_epoch=nb_train_samples // batch_size,
        epochs=epochs, validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size)

print('********************** Save Model ****************************')

# save trained model
model.save_weights("multi_model_weights_saved2.h5")
model.save("multi_model_saved_all2.h5")
modelN=load_model("multi_model_saved_all.h5")
```

Machine Learning Classification

```
# Necessary Imports
# Skipped for
# Readability

img_width = 224
img_height = 224

# load model
modelN=load_model("multi_model_saved_all2.h5")
modelN.compile(loss='sparse_categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

print('********************** Test First Case ***************************')
#load Test Image
TestCase='funnyCow.jpeg';

# input image and change to proper input shape
img = cv2.imread(TestCase);
img = cv2.resize(img,(img_width,img_height))
img = np.reshape(img,[1,img_width,img_height,3]);

# classify image and print prediction
classes = modelN.predict_classes(img);
print (classes);
value_est=classes[0]
print(value_est)

# display image with prediction
img = cv2.imread(TestCase);
font = cv2.FONT_HERSHEY_SIMPLEX;
if value_est==0:
        print("******************It is 0******************")
        cv2.putText(img,'   Cow   ',(0,130), font, 1, (250,255,155), 2, cv2.LINE_AA);
elif value_est==1:
        print("******************It is 1******************")
        cv2.putText(img,'   Car   ',(0,130), font, 1, (250,255,155), 2, cv2.LINE_AA);
else:
        print("******************It is 2**************")
        cv2.putText(img,'    Plane    ',(0,130), font, 1, (200,255,155), 2, cv2.LINE_AA);
cv2.imshow('image',img);
cv2.waitKey(0);
cap.release()
```

MATLAB UAV Control Model



MATLAB Bovine Detection

```
%% Load Image and Display
%Read image
im = 'New_Grass_Cows.png';
rgbImage = imread(im);
[rows, columns, numColorBands] = size(rgbImage);

%Display original image
subplot(2,2,1);
imshow(rgbImage, []);
title('Input Image');

%Place figure at fullscreen
set(gcf, 'units','normalized','outerposition',[0 0 1 1]);

% detect if grayscale image
if numColorBands > 1
    grayImage = rgb2gray(rgbImage);
else
    grayImage = rgbImage;
end

%display grayscale image
subplot(2,2,2);
imshow(grayImage, []);
title('Grayscale');

%Display Rough Binary Image
binary = grayImage > 145;
subplot(2,2,3);
imshow(binary,[]);
title('noisy Binary')

%Display Filtered Binary Image
cleanBinary = bwareaopen(binary, 300);
subplot(2,2,4);
imshow(cleanBinary, []);
%% Blob Detection

%find connected pixels and record geometric properties
[labeledImage, numberOfObjects] = bwlabel(cleanBinary);
blobMeasurements = regionprops(labeledImage, 'Perimeter', 'Area', 'FilledArea', 'Solidity', 'Centroid');

%fill holes in image
filledImage = imfill(cleanBinary, 'holes');
boundaries = bwboundaries(filledImage);

%record geometric properties
perimeters = [blobMeasurements.Perimeter];
areas = [blobMeasurements.Area];
filledAreas = [blobMeasurements.FilledArea];
solidities = [blobMeasurements.Solidity];
centroid_hold = [blobMeasurements.Centroid];
centroids = zeros(2,numberOfObjects);
centroids(1,:) = centroid_hold(1:2:2*numberOfObjects-1);
centroids(2,:) = centroid_hold(2:2:2*numberOfObjects);
circularities = perimeters.^2./(4*pi*filledAreas);

%print properties of each blob of significant size
fprintf('#,Perimeter,Area,FilledArea,Solidity,Circularity\n');
Num_cows = 0;
for blobNumber = 1 : numberOfObjects
    if areas(blobNumber) > 1200
        Num_cows = Num_cows + 1;
        fprintf('%d, %9.3f, %11.3f, %11.3f, %8.3f, %11.3f\n', Num_cows, perimeters(blobNumber),...
            areas(blobNumber), filledAreas(blobNumber), solidities(blobNumber), circularities(blobNumber));
    end
end

%print number of cows detected in the image
subplot(2,2,4);
title(strcat('Number of Cows detected: ', num2str(Num_cows)));
subplot(2,2,2);
hold on;
```

```
%Plot edges of the blobs on the grayscale image
for blobNumber = 1 : numberOfObjects
    if areas(blobNumber) > 1200
        % Plot boundaries and centroids
        thisBoundary = boundaries(blobNumber);
        thisBoundary = thisBoundary{1};
        plot(thisBoundary(:,2),thisBoundary(:,1), 'b', 'LineWidth', 3);
        thisBoundary = boundaries(blobNumber);
        thisBoundary = thisBoundary{1};
        plot(thisBoundary(:,2),thisBoundary(:,1), 'b', 'LineWidth', 3);
        plot(centroids(1,blobNumber),centroids(2,blobNumber),'r*');
    end
end
```